Part II

ECHONET Communication Middleware Specifications

Revision History

Note) Version numbers except Ver.3.20 indicate Japanese editions.
Unless otherwise stated, all versions are upward compatible.

- Version 1.0 March 18th 2000 Released Open to consortium members
  July 2000 Open to the public
- Version 1.01 May 23rd 2001 Open to the public
- Version 2.00 August 7th 2001 Open to consortium members

The following Table-of-Contents entries were revised:

	Revised entries	Revision/addition
1	1.2	The description of a registered trademark used in a figure was added.
2	4.2	A message format was added for the introduction of the secure communication function.
3	4.2.1	An EHD requirement was added for the introduction of the secure communication function.
4	4.2.6	Specifications for the AV-related device class group were added.
5	4.2.6, Table 4.3	Corrections were made to achieve agreement with the APPENDIX.
6	7.4	An explanation of the address conversion process was added.
7	7.4.1, 7.5.1, 9.11.5	The descriptions were changed because the power line A and power line B methods were integrated into a single method.
8	9.2.3	An explanation of the status change announcement was added.
9	9.3.5, 9.10.2, 9.12.2	Explanations of property maps were added.

- Version 2.01 December 19th 2001 Open to consortium members
  Typographical errors in Version 2.00 were corrected.
- Version 2.10 Preview December 28th 2001 Open to consortium members

The following Table-of-Contents entries were revised:

	Revised entries	Revision/addition
1	4.2	Specifications for frame formats for secure messages, compound messages, and arbitrary messages were added.
2	4.2.11	The section on the compound ECHONET service was added in conjunction with compound message format requirement.
3	4.2.12	The section on the processing target property counter was added in conjunction with compound message format requirement.
4	4.2.13	The section on the property data counter was added in conjunction with compound message format requirement.
5	9.11.1	The version data property was added to the node profile class.
6	Chapter 10	ECHONET secure communication specifications were added.

- Version 2.10 Draft February 15th 2002 Open to consortium members

The following Table-of-Contents entries were revised:

	Revised entries	Revision/addition
1	4.2.1	- Descriptions were corrected.
2	Chapter 8	- The ECHONET Communications Processing Block state transitions were revised. The descriptions of the Protocol Difference Absorption Processing Block state transitions were deleted.
3	4.2.11	- The description of the compound message service (CpESV) was corrected.
4	Chapter 5	- A router startup sequence was added to ensure that routing is partly achievable at a warm start.
5	4.2.6	- The secure communication access property setup class group was added. - The secure communication common key setup node was added to the 0x05 management/control-related device class group (0x05).
6	6.7.1	- A node startup process was added.
7	9.3.1	- The specification version data property was added to the device object super class.
8	9.9.1	- Detailed specifications for the secure communication common key setup node class were added.
9	9.11.1	- The version data property code was added. - The secure communication common key setup (User Key) property was added. - The secure communication common key setup (Service Provider Key) property was added. - The secure communication common key switchover (User Key) property was added. - The secure communication common key switchover (Service Provider Key) property was added.
10	9.11.2	- The router attribute was added as an item to the registration request router data property for the router profile class. - The master router data property was added.
11	9.13, 9.14, 9.15, 9.16	- Explanations of the communications definition object were added.
12	9.17	- The secure communication access property setup class group was stipulated.

・ Version 2.10 March 7th 2002 Open to consortium members

The following Table-of-Contents entries were revised:

	Revised entries	Revision/addition
1	1.1	- The descriptions of EHD b6 and b7 were revised (page 1-1).
2	3.4	- A description of trigger setup was added.
3	4.2.1	- The erroneous description of b7 in Fig. 4.2 was corrected.
4	4.2.2	- The description in Fig. 4.4 was revised.
5	4.2.5	- The b2-b5 descriptions were deleted from the note in Fig. 4.6.
6	4.2.6	- The “o” mark was added to the Remarks column for the secure communication common key setup node in Table 4.7.
7	4.2.7	- The erroneous description of b7 in Fig. 4.7 was corrected. - The contents of Note 1) for Table 4.9 were revised.

8	4.2.8	- The erroneous descriptions of b6 and b7 in Fig. 4.8-1 were corrected. - Descriptions were revised.
9	4.2.11	- The description of the compound message service (CpESV) was revised. - Descriptions were revised.
10	5.2.2	- Erroneous descriptions were corrected.
11	5.4.3	- Descriptions were revised.
12	5.4.3, 5.4.4	- The "ECHONET router" portion of the title was changed to "normal router".
13	7.1	- Descriptions were revised.
14	7.7	- Descriptions were revised.
15	8.2	- Descriptions were revised. - The descriptions in Fig. 8.1 were revised. - The descriptions in Table 8.1 were revised.
16	9.1	- Descriptions were revised.
17	9.3.3	- Descriptions were revised.

- Version 2.11 April 26th 2002 Open to consortium members
  The following Table-of-Contents entries were revised:

	Revised entry	Revision/addition
1	4.2, Fig. 4.1-2	- Explanation was added in relation to the EDT maximum value during secure communication.
2	Chapter 4	- Figure numbers for Fig. 4.7 and succeeding figures were corrected.
3	4.2.6 Tables 4.3,4.4,4.7,4.8	Revision of existing explanation.
4	4.2.8 (4) to (10)	- Revision of existing explanation and addition of explanation.
5	5.3, Figs. 5.7, 5.8-1 and 5.8-2	- Numbers in figures were corrected.
6	5.4, Figs. 5.9,5.10-1,5.10-2,5.13,5.14,5.16,5.17	- Numbers in figures were corrected.
7	5.4, Tables 5.1 and 5.2	- Table numbers were corrected.
8	5.4, Fig. 5.13	- Messages (5) and (6) were added.
9	5.4, Figs. 5.13, 5.14 and 5.17	- Registration router property was corrected.
10	8.2, Table 8.1	- ClcReset was corrected to ClcStart.
11	9.2,9.3	Revision of existing explanation.
12	9.3.3	Revision of existing explanation and addition of explanation.
13	9.16, page 9-49 Condition 4	Revision of existing explanation.
14	9.17, page 9-53 and page 9-55(5)	- Explanation was revised.
15	10.4.9, Fig. 10.8	- “Secure Key” was added.
16	10.4.11	- Explanation was revised.
17	10.9, Figs. 10.24	- Explanation was added and property codes were changed in

	through 10.26	figures.

Version 3.00 Draft June 12th 2002 Open to consortium members

The parts to which changes have been made are as follows:

	Changed part (section number in the Table of Contents)	Summary of additions/changes
1	1.2	* Addition; Fig. 1.1 New Transmission Media
2	7.4.6, 7.4.7	* Address conversion processing requirements for the IP/Bluetooth and IP/Ethernet/IEEE802.3 protocols added.
3	9.2.3	* Addition of explanation of the announcement (broadcasting) function to notify property value (status) changes.
4	9.3.1	* "Node identification number property" and "Manufacturer error code" added to Table 9.2.
5	9.3.13	* Node identification number property requirements added.
6	9.11.1	* Addition of "Node identification number property" ((25) and node profile class property table).

Version 3.00 August 29th 2002 Open to consortium members

The parts to which changes have been made are as follows:

	Changed part (section number in the Table of Contents)	Summary of additions/changes
1	4.2.6	* Class group code added.
2	4.2.8 (1) through (3)	* Additions to the explanation.
3	4.2.8	* Access rule column amended.
4	5.4.1	* The description of message (2) in Fig. 5.9 was amended.
5	5.4.2	* The description of message (2) in Fig. 5.10-1 was amended.
6	5.4.3	* The descriptions of messages (1), (12) and (13) in Fig. 5.13 were amended.
7	5.4.4	* The descriptions of messages (14) and (15) in Fig. 5.14 were amended.
8	5.5.3	* Additions to Fig. 5.17 (message (8)).
9	9.3.6	* Changes to the error description property table (Table 9.4).
10	9.3.12	* Additions to the explanation.
11	9.3.14	* Manufacturer error code requirements added.
12	9.9.1	* Explanation amended.
13	9.11.1	* Additions and amendments to the explanation.
14	9.17	* Additions to the explanation.
15	9.18	* Additions to the explanation.
16	9.19 (former 9.17)	* Explanation amended.

Version 3.10 Draft November 8th 2002 Open to consortium members

The parts to which changes have been made are as follows:

	Changed part (section number in the Table of Contents)	Summary of additions/changes
1	4.2	* Addition of the “Reserved for future use.” note to the parts describing “free-format messages” in Figs. 4.10-1 and 4.10-2 that can be set freely.
2	4.2.1	* Addition of the “Reserved for future use.” note to the explanation of free-format messages.
3	5.2	* Explanation on composite messages was added.
4	5.3.2	* Explanation of T3 added to Table 5.8-2.
5	5.4.1	* Number correction (Fig. 5.9).
6	5.4.2	* Number corrections (Figs. 5.10-1 and 5.10-2). * The explanation of T3 in Fig. 5.10.2 (former Fig. 5.10.1) was amended. * The explanation of message (3) in Fig. 5.10.3 (former Fig. 5.10.2) was amended.
7	5.4.3	* “*5” added to Fig. 5.13.
8	5.5.3	* The explanation of message (9) in Fig. 5.17 was amended. * Message number correction (Fig. 5.17 (instance change class announcement)).
9	9.3.1	* Addition of the ○ (“compulsory”) mark to “node identification number” in Table 9.2.
10	9.11.2	* Addition of notes. * Explanation on home router data added (3). * Explanation on registration request router data amended (6).
11	9.11.3	* “Transition” state EPC amended.
12	9.11.4	* “Transition” state EPC amended.
13	9.11.5	* “Transition” state EPC amended.
14	9.15	* Array element No. mask value size amended. * Masked array element No. size amended.

Version 3.10 December 18th 2002 Open to consortium members

The parts to which changes have been made are as follows:

	Changed part (section number in the Table of Contents)	Summary of additions/changes
	3.2	* The explanation was changed to make it clearer. * Explanation of the properties for which “Get” can be performed was added.
	4.2.6	* Entities deleted from Tables 4.2 through 4.6. The “Refer to Part VIII for explanation on service classes” note was added. * Table 4.9 added.
	4.2.7-	* Tables 4.9 through 4.13 were changed to Tables 4.10 through 4.14, respectively.
	4.2.11 (2)	* The explanation on response was amended.
	5.4.1	* Fig. 5.10-1 amended.
	5.4.2	* Fig. 5.10-2 amended.

	5.5.1	* The explanation on manual router setting was amended.
	9.3.1	* Table 9.2 corrected (error description, node identification number, version information)
	9.3.12	* Explanation amended.
	9.3.13	* Additions and amendments to the explanation.
	9.11.2	* Notes to the table amended.

Version 3.12 May 22nd 2003 Open to consortium members

The parts to which changes have been made are as follows:

	Changed part (section number in the Table of Contents)	Summary of additions/changes
1	4.2	* Explanation on the formats for messages exchanged between protocol difference absorption processing sections was added (2).
2	6.8	* Amendments to Table 6.1 (M4c "Remark" column).
3	9.11.3 through 9.11.5	* The EPC for the property "state shift" was changed from 0x8F to 0xAF.

Version 3.20 Draft October 17th 2003 Open to consortium members

The parts to which changes have been made are as follows:

	Changed part (section number in the Table of Contents)	Summary of additions/changes
1	2.3	* The "automatic assignment code area" was changed to "NetID assignment code area."
2	4.2	* The secure communication frame formats were changed.
3	5.1	* Definitions of timeouts and response waiting periods were added.
4	5.4	* The master router data property value in Fig. 5.9 was corrected.
5	5.4.1	* Additions, alterations and amendments to the explanations.
6	5.4.2	* Additions, alterations and amendments to the explanations.
7	5.4.3	* Additions, alterations and amendments to the explanations.
8	5.4.4	* Additions, alterations and amendments to the explanations.
9	5.5.3	* Alterations and amendments to the explanations.
10	5.6	* A new section was added.
11	5.7	* A new section was added.
12	9.2.2	* Additions to the explanation.
13	9.2.3	* Explanation amended.
14	9.3.3	* "Lavatories" was changed to "lavatories and locker rooms."
15	9.11.1	* "Secure communication common key setting (User Key)" and "Secure communication common key setting (Service Provider Key)" – changes were made in key size. * "Set" (default router data) was made compulsory.
16	9.11.2	* Property additions and deletions. * Explanation on properties added.

		* NetID “Get” made permissible only with regard to master routers.
17	9.11.3	* A new NetID server profile class was added.
18	9.11.4 through 9.11.6	* 9.11.3 through 9.11.5 were renumbered as 9.11.4 through 9.11.6, respectively.
19	9.16	* Explanation corrected.
20	Chapter 10	* The encryption method was changed from the DES method to the AES method.
21	10.6.1	* Explanations in Figs. 10.11 and 10.12 were corrected.
22	10.7.3	* Explanation in Fig. 10.15 was corrected.
23	10.7.4	* Explanation in Fig. 10.17 was corrected.
24	10.7.6	* Explanations in Figs. 10.19 and 10.20 were corrected.
25	Appendix 6	* A NetID basic sequence diagram was added.
26	9.16	* “ EA mask data” was amended as “ Instance code mask data.”
27	Chapter 10	* The MAS calculation range was changed. * The MAS position was changed to the end position in the frame format. * SHD composition changed to SKH and SNF. * SKH was extended by combining the existing SKH and AHD functions to a 2-byte parameter.
28	4.2.8	* EPC: The explanation of the processing for the case where no request is accepted (0x6C) was amended.
29	4.2.11	* “CpESV = 0x70” was added as an item “Reserved for future use.”
30	9.3.13 9.11.1	* The following was added to the explanation of node identification numbers: “If the hardware address is less than 8 bytes, the hardware address is stored in the unique number field from the lowest-order byte and the remaining bytes are padded with “0.”
31	9.11.2	* The NetID access rule “Get” shall be implemented in master nodes only.
32	9.11.1	* The default router data access rule “Set” was made compulsory. * Data composition changed for “Secure communication common key setting (User Key)” and “Secure communication common key setting (Service Provider Key).” * The secure communication common key (Serial Key) property was added.
33	9.15	* The action setting EPC was changed to an array type and linked startup condition values were added.
34	9.16	* The trigger setting EPC was changed to an array type and linked startup condition values were added.
35	10.3	* A new section was added.
36	9.3.12	* Explanation on the property map was added.

Version 3.20 December 12th 2006 Open to the public

The parts to which changes have been made are as follows:

	Changed part (section number in the Table of Contents)	Summary of additions/changes

1	6.2	* Explanation on the reception processing for ECHONET routers was added.
2	8.2	* Amendment to Table 8.1 (“instruction to the lower layers” in relation to “normal operation.”)
3	9	* The description relating to data type was amended so that it becomes consistent with that in Appendix.
4	9.3.13	* The method to store hardware addresses was changed to one that stores hardware addresses from the highest-order bytes.
5	9.11.1	* The method to store hardware addresses was changed to one that stores hardware addresses from the highest-order bytes.
6	9.11.2	* The access point function (EPC: 0xEA) was added.
7	9.19	* The service provider level property map access rule was amended.
8	10.5.4	* Transaction ID (TID) was added to SHD. The MAS generation range was amended.
9	10.8.3	* The encryption method was changed from Serial Key-based one to one that uses the secure communication common key setting (Serial Key) property.

Version 3.30 December 2, 2004 Open to consortium members

The parts to which changes have been made are as follows:

	Changed part (section number in the Table of Contents)	Summary of additions/changes
1	1.2	• The IEEE802.11/11b transmission media were added in Fig. 1.1.
2	4.2.2	• Explanations about group broadcasting within individual domains and within the home subnet were added.
3	5.4.3	• Explanations about NetID initial values were added.
4	5.4.3	• Explanations about the cold starting of automatic and manual setting routers were added.
5	5.4.4	• The explanations about warm starting were divided into explanations about the warm starting of automatic setting routers and those about the warm starting of manual setting routers.
6	5.6.1	• Explanations about NetID server processing were changed. 1) Router function checking was deleted.
7	5.6.1	• Detailed explanations about the router registration status property were added.
8	5.7	•Detailed explanations about the EHD of received data were added.
9	5.8	• The “Types of Messages that ECHONET Nodes Are Not Allowed to Send” section was added.
10	6.2	• Explanations about the processing to be performed after receiving an intra-domain group broadcast message and after receiving an intra-home subnet group broadcast message were added.
11	7.4	• The sentence stating that reference be made to Part 5 for details about destination address was deleted.
12	7.4.8	• The “Address Conversion Specifications for the IEEE802.11/11b Protocol” subsection (Subsection 7.4.8) was added.
13	7.5	• The bracketed part of the explanation about destination address was deleted.
14	7.5.1	• The explanation in the “Communication Type

		Conversion Specifications for the Power Line Communication Protocols" subsection was amended.
15	7.5.8	• The "Communication Type Conversion Specifications for the IEEE802.11/11b Protocol" subsection (Subsection 7.5.8) was added.
16	7.7	• Bluetooth®-, Ethernet- and IEEE802.11/11b-related functions were added in Table 7-1(1/2) and Table 7-1(2/2).
17	9.3.1	• The node identification numbers for IEEE802.11/11b (7.4) were added in Table 9.2.
18	9.11.1	• The node identification numbers for IEEE802.11/11b were added in the "Node Profile Class: Detailed Specifications" table.
19	9.11.1	• The "group broadcast number" property was added.
20	9.11.1	• The version information was changed as a result of the addition of the group broadcast function.
21	9.11.2	• Incorrect descriptions about router attributes were corrected.
22	9.11.6	• Bluetooth®, Ethernet and IEEE802.11/11b were added as lower-layer communication software types.
23	10.4	• The IEEE802.11/11b transmission media were added in Fig. 10.1.
24	Appendix 6	• Explanations about T7 were amended.
25	9.11.2	• Explanations were added about the case where the router profile class is implemented in a NetID server that does not perform received message routing processing.

Version 3.40 Draft December 28, 2004 Open to consortium members

The parts to which changes have been made are as follows:

	Changed part (section number in the Table of Contents)	Summary of additions/changes
1	1.2	• Power Line Communication Protocol D and C Systems were added in Fig. 1.1. The layout was changed.
2	4.2.6	Bit 7 of the class group code was rid of the restriction about its value. • The words "0: Fixed" for Bit 7 of the class group code in Fig. 4.7 was deleted. • The note to Fig. 4.7 was deleted. • The class group code value range (shown at the bottom of the page) was changed to "0x00 to 0sFF."
3	9.12	Communication definition class groups added. • Explanations were added about the overview of the property status referencing request compliability communication definition class group and property status notifiability communication definition class group.
4	9.20	• Specifications for the property status referencing request compliability communication definition class group were added.
5	9.21	• Specifications for the property status notifiability communication definition class group were added.
6	9.3	• "0x11 to 0x1F: Power line" in Table 9.2 was changed to "0x11 to 0x1F: Power Line Communication

		Protocol A and D Systems."
7	9.3	• "0xA1: Power Line Communication Protocol C System" was added in Table 9.2.
8	9.11.6	• Power Line Communication Protocol C and D Systems were added to the list under the heading "(5) lower-layer communication software type."

Version 3.40 February 3, 2005 Open to consortium members

	Changed part (section number in the Table of Contents)	Summary of additions/changes
1	Appendix 7	Explanations were added to the definition of the EDT size error.

Version 3.41 May 11, 2005 Open to consortium members

Version 3.2 October 13, 2005 Open to the public

Version 3.42 October 27, 2005 Open to consortium members

	Changed part (section number in the Table of Contents)	Summary of additions/changes
1	4.2.8	• Clear explanations were added about the "response not possible" processing for request messages that involve array elements.
2	9.3.9 9.3.10	• Requirements for storing product code property and serial number property data were added.
3	9.3.12	• Explanations about the property map property were added.
4	9.11.1	•Explanations about unique identifier data were added.
5	Appendices 2, 4 and 5	• Examples of values that can be used in the bit map were added.
6	Appendix 7	• Explanations were added about whether or not the array element number is necessary in relation to "response not possible" messages.

Version 3.50 Draft August 3, 2006 Open to consortium members

Version 3.50 September 20, 2006 Open to consortium members

Version 3.51 Draft February 2, 2007 Open to consortium members

Version 3.60 March 5, 2007 Open to consortium members

December 11, 2007 Open to the public

The specifications published by the ECHONET Consortium are established without regard to industrial property rights (e.g., patent and utility model rights). In no event will the ECHONET Consortium be responsible for industrial property rights to the contents of its specifications.

The publisher of this specification is not authorized to license and/or exempt any third party from responsibility for JAVA, IrDA, Bluetooth or HBS.
A party who intends to use JAVA, IrDA, Bluetooth or HBS should take action in being licensed for above-mentioned specifications.

In no event will the publisher of this specification be liable to you for any damages arising out of use of this specification.

The original language of The ECHONET Specification is Japanese. The English version of the Specification was translated the Japanese version. Queries in the English version should be refereed to the Japanese version.

Contents

Chapter 1 Overview

1.1 Basic Concept

The ECHONET Communication Middleware specifications indicated in this Part not only concern the communication protocol but also include processing for the portion found between the Application Software Block and the Lower-Layer Communications Software Block shown in the next section (Section 1.2 "Positioning on Communications Layers"). The communication protocol specifications are described in Chapters 2 through 5.

The ECHONET Communication Middleware specifications were designed primarily to enable concealment of differences in Lower-Layer Transmission Medium from the perspective of the application layer. Other issues relating to communication protocol specifications for the communication middleware block are listed below.

(1) Use of ET-2101[*1] resources

For EHD (ECHONET Header) specifications, the header codes (HD) specified in ET-2101 were used. In ET-2101, the b7 and b6 values are defined as fixed values for future expansion. In EHD, however, these values are defined differently, with the value b7 defined as a fixed value for future expansion.

(2) Use of JEM-1439[*2] resources

The specific command contents (device types, specific codes, etc.) of JEM-1439 were used for specific device object type and code specifications.

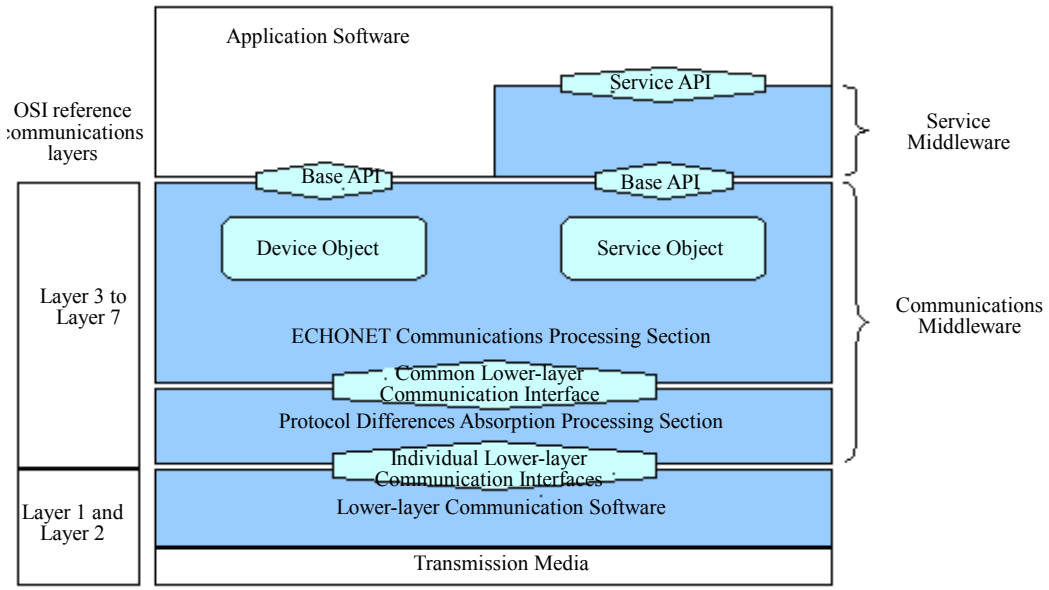
Notes:

1) A home network standard published in Sept. 1988 by the Electronic Industries Association of Japan. For detailed information on this standard, see References 1–3 in Appendix 1.

2) A home network (especially home equipment) standard published in Aug. 1988 by the Electronic Industries Association of Japan. For detailed information on this standard, see Reference 4 in Appendix 1.

1.2 Positioning on Communications Layers

Communication Middleware is positioned between Application Software and Lower-Layer Communications Software. Specifications are provided in this Part. In Fig. 1.1, the shaded area shows the Communication Middleware block to be specified.



Lower-layer Communication Software Supported by the Current Version

Simbol	Name of Lower-layer Communication Software	Transmission Medium
A	Power Line Communication Protocol A System Power Line Communication Protocol D System	Power distribution lines
B	Low-power Radio	Low-power radio
C	Extended HBS	Twisted-pair cables
D	IrDA Control	Infrared
E	LonTalk®	Low-power radio
F	Bluetooth® (UDP/IP)	Low-power radio (BT)
G	Ethernet IEEE802.3 (UDP/IP)	Ethernet
H	IEEE802.11 IEEE802.11b (UDP/IP)	Low-power radio (WLAN)
I	Power Line Communication Protocol C System	Power distribution lines

LonTalk® is a registered trademark of the Echelon Corporation that is used in the United States and other countries.

Bluetooth® is a registered trademark of Bluetooth SIG, Inc.

Ethernet is a registered trademark of the Xerox Corporation.

All other trademarks are properties of their respective owners.

Fig. 1.1 Positioning of the Communication Middleware

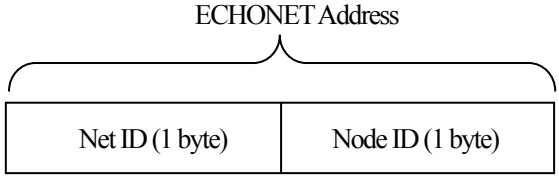
Chapter 2 ECHONET Address

2.1 Basic Concept

The ECHONET Address was introduced to conceal differences in Lower-Layer Transmission Medium from the ECHONET Communications Processing Block and the Application Software. The basic requirement for the address is that it uniquely identifies an ECHONET Node. The ECHONET Address is a logical address defined separately from the MAC address unique to each given transmission medium.

2.2 ECHONET Address Structure

An ECHONET Address consists of (1) an address (hereafter referred to as a Node ID) determined based on an address (hereafter referred to as a MAC address) that enables communication in Layer 2 of the transmission medium and (2) an address (hereafter referred to as a Net ID) that specifies the subnet. Specifically, it consists of a Net ID and a Node ID that uniquely correspond to the MAC address. Both are shown in the figure below. The Node ID is logically assigned so as to be unique within the subnet.



The procedure for deciding an ECHONET Address is specified in Chapter 5.
When the subnet changes due to a change in location of the ECHONET Node, its ECHONET address also changes. Specifying the ECHONET Node in an ECHONET Domain before and after movement can be performed using the device-unique data held in the Device Profile Object of each device (see Section 3.3 “Profile Objects”).

2.3 Net ID

The Net ID signifies a subnet identifier. The Net ID of each ECHONET node is set based on subnet information held in ECHONET Routers. Until the Net ID of an ECHONET node is set by an ECHONET Router, the Net ID is set to "0x00", indicating "Net ID not specified," and the node can communicate only within the subnet to which the node belongs. See Chapter 5 for the Net ID setting process.

Table 2.1 NetID Codes

	NetID (HEX)	Signification	Remarks
1	0x00	NetID not specified	
2	0x01 to 0x8F	NetID assigned codes	
3	0x90 to 0xFF	Codes available to user (manually assigned codes)	Used, for example, when a system manager for an apartment complex or building is present.

2.4 Node ID

The Node ID signifies an identifier used to uniquely identify an ECHONET node within a subnet. The Node ID is converted from a MAC address such that it is unique within the subnet. Each type of Lower-Layer Communications Software has its own conversion specifications (see Section 7.4 Address Conversion Processing).

Chapter 3 ECHONET Object

3.1 Basic Concept

The ECHONET Objects specified in this section were introduced with two objectives: first, compartmentalization of functions of devices connected to the ECHONET network; and second, modelization of communication between devices to enable application software developers to utilize ECHONET communication whenever possible without concern for detailed specifications. The ECHONET Objects are processed in the ECHONET Communications Processing Block. Control content exchanged in communications can be classified into four types: those relating to functions unique to each device; those relating to data profiling something other than the functions unique to each device; those relating to object communication operations; and those relating to service middleware functions. In ECHONET, all of these are specified as objects, and control and data exchange were achieved to enable their manipulation. The ECHONET specifications stipulate four ECHONET Objects:

(1) Device Objects
(2) Profile Objects
(3) Communications Definition Objects
(4) Service Objects

Each ECHONET Object has properties. The various unique functions possessed by an ECHONET node are represented as ECHONET Properties. Reading or writing the ECHONET Properties of the ECHONET Object in the relevant ECHONET node operates the device.

ECHONET Objects are defined as the following specifications: object type (codes are specified in the next section as EOJ); the properties possessed by each object (codes are specified in the next section as EPC); and the services for those properties (codes are specified in the next section as ESV). The following issues were taken into account when formulating the detailed specifications:

- It was assumed that each ECHONET node would have more than one Device Object of the same type (e.g., two Human Detection Sensor objects in the same node), and that identification could be performed by stipulating a specific code (see detailed specifications for EOJ in the following section).
- It was assumed that the various communications-related settings and status confirmations could be carried out by application software as ECHONET Object operations.

3.2 Device Objects

Device objects, which are "their working functions as devices" equipped with the devices, are specified in detail. Device objects aim easy control and status confirmation through communications between devices. Device Object data resides in the ECHONET Communication Middleware, but the device mechanical functions themselves reside in the Application Software Block. The ECHONET Communication Middleware manages instance property data and manages and processes operations related to communication for properties. In these Specifications, the term "Device Object" shall be used as a generic term for home air conditioners, refrigerators with freezers, etc. The object definitions for each Device Object are specified (see Appendix).

In a single ECHONET Device, one or more Device Objects is defined. Each Device Object defines the properties to be used in each class and the services corresponding to the properties. Figure 3.1 below demonstrates this relationship using a concrete example.

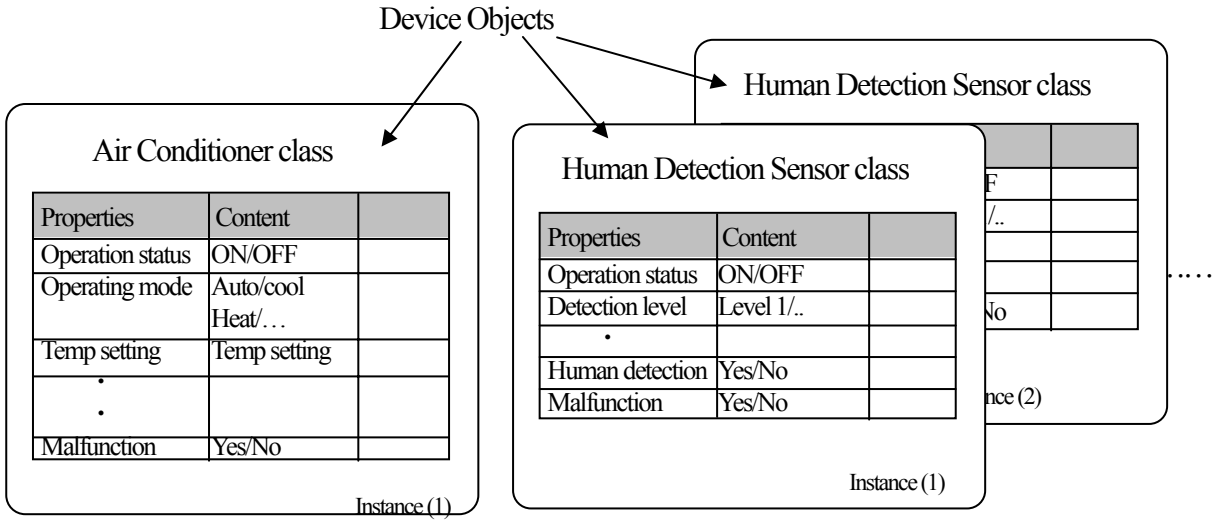


Fig. 3.1 Device Object

Class definitions for the Device Objects (Air Conditioner, etc.) (i.e., property configurations and other specific definitions and code specifications) are listed in Chapter 9 "ECHONET Objects: Detailed Specifications" and in the Appendix. Other ECHONET nodes seeking to control the functions and confirm the status of an ECHONET node via ECHONET do so by manipulating (i.e., reading/writing) these device objects.

When a value is written into a property, the value will be handed to the application software for processing. Whether processing is actually performed or not is determined by the value written into the property and the status of the application software.

With regard to Device Object property values, it must be possible to read the value currently held by the corresponding application software according to the class definitions given in the Detailed ECHONET Device Object Requirements Appendix, and, based on the functions of the application software, a change shall be generated by user operation of the equipment, automatic control through internal processing of the

equipment and/or ECHONET communication-based writing operation.

3.3 Profile Objects

ECHONET Node Profile data, such as ECHONET node operating state, manufacturer information, and implemented Device Objects list, are specified to enable manipulation (read/write) by application software and other ECHONET nodes. In these specifications, the term "Profile Objects" shall be used as a blanket term to refer to various ECHONET Profile Classes, such as Node Profile Object, Router Profile Object, and Protocol Difference Absorption Processing Block Profile Object, with detailed specifications to be provided individually (see Chapter 9). Similar to the Device Objects shown in Fig. 3.1 on the preceding page, Profile Objects define the properties to be used in each class and the services corresponding to the content and properties thereof (see Chapter 9 "ECHONET Objects: Detailed Specifications"). Operations on the various profiles of an ECHONET node are performed by manipulating (reading/writing) these profile objects.

3.4 Communications Definition Objects

Communications Definition Objects is the blanket term used to refer to all objects specified with the objective of manipulating the communications-based operations of Device Objects, Profile Objects, and Service Objects. Specifications will be provided for each class of Device Objects and Profile Objects, and Service Objects (e.g., Air Conditioner Communications Definition Object and Node Profile Communications Definition Object), which will be described later. It is possible to control communications operations when manipulating the properties of individual Device Objects, Profile Objects, and Service Objects by manipulating (i.e., by reading/writing) these Communications Definition Objects. Operations specified by the Communications Definition Objects are shown below. Detailed specifications are presented in Chapter 9.

3.5 Service Objects

Functions to be disclosed to the network based on Service Middleware functions are modeled, and the class specifications are defined as Service Objects. They are provided to enable operation of Service Middleware from other ECHONET devices via the ECHONET network. Detailed specifications are provided in Chapter 8.

3.6 ECHONET Objects as Viewed from Application Software

Control of ECHONET Objects from application software is performed using the Basic APIs specified in Part IV. Here, control from application software using Basic APIs is described for the three main cases listed below, with a focus on how the ECHONET Objects are perceived.

Case 1: Obtaining other node status
Case 2: Controlling other node functions

Case 3: Notifying other nodes of self-node status

This section shows only how ECHONET Objects are seen from the perspective of application software and does not provide API processing specifications. For this, refer to the detailed specifications in Part IV.

(1) ECHONET Objects when obtaining other node status

The ECHONET Specification provides two methods for obtaining the status of another node. These methods are shown in Figs. 3.2-1 and 3.2-2. In the method shown in Fig. 3.2-1, when a request is received from an application, an obtain status request is issued to objects in the specified other node (Node B), with the results notified to the application. With this method, object data for the other node need not be stored in the ECHONET Communication Middleware for the node (Node A in the figure) making the request. In the second method, shown in Fig. 3.2-2, even when no request is received from an application, the ECHONET Communication Middleware catches and holds the notified status of objects in other nodes in advance, and then returns them to an application when it receives a request.

In this method, objects copied to ECHONET Objects in other nodes actually exist within the ECHONET Communication Middleware. In the former method, because the access is performed from an application, a virtual copy of the ECHONET Objects in the other node exists in the ECHONET Communication Middleware. In both cases, in order to set the desired ECHONET Object instance via the Basic API, not only the ECHONET Object class code but also an instance code and data specifying the node (ECHONET address, etc.) are necessary. From the viewpoint of the application, therefore, ECHONET Objects are seen in the relationship shown in Fig. 3.2-3 within the ECHONET Communication Middleware.

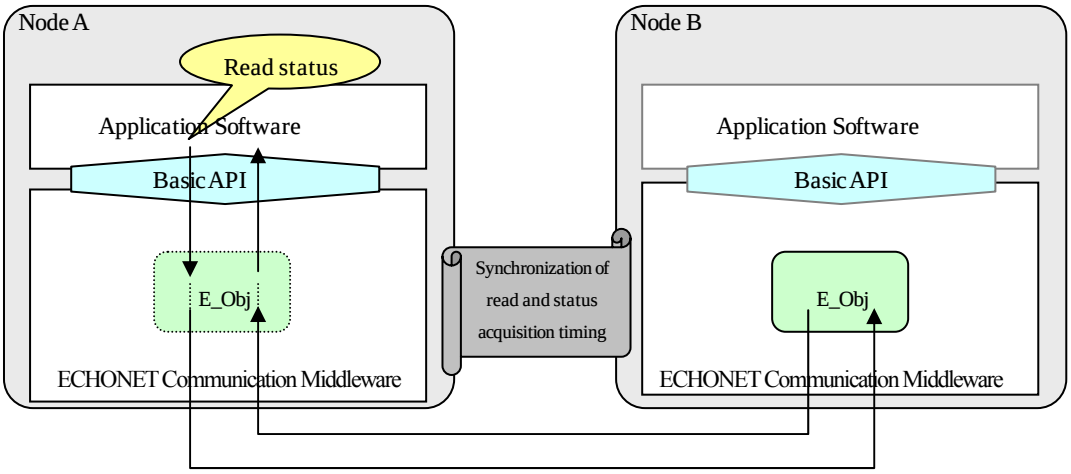


Fig. 3.2-1

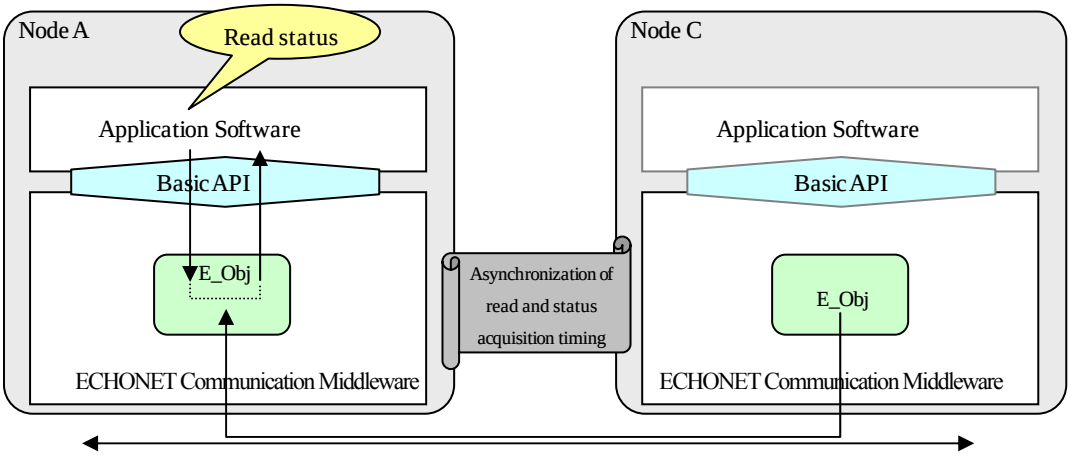


Fig. 3.2-2

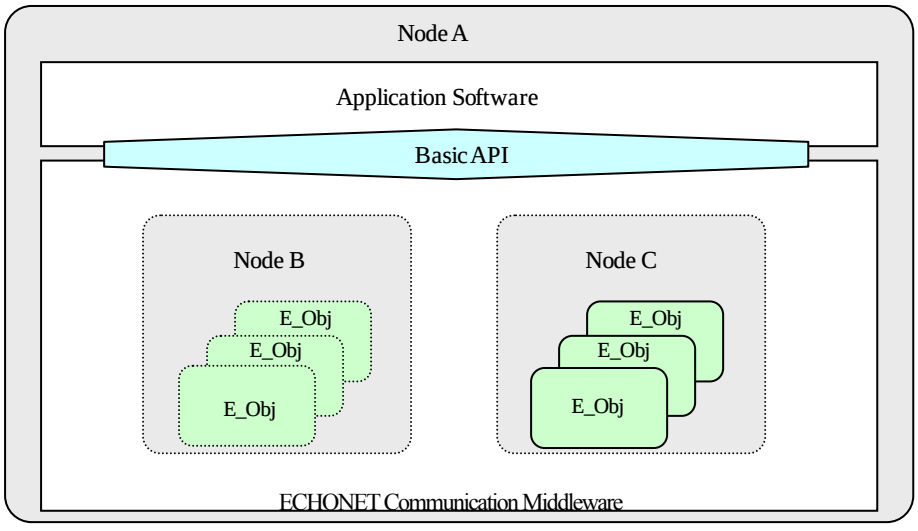


Fig. 3.2-3

(2) ECHONET Objects when controlling other node functions

ECHONET provides a method for controlling the functions of other nodes, as shown in Fig. 3.2-4. Just as in Fig. 3.2-1, however, a request for control (property value setting) is issued to objects in the specified other node (Node B), and the application is then notified of the results (although there are exceptions to this). Basically, therefore, property data for objects in the other node (Node B) need not be present in the ECHONET Communication Middleware for the node (Node A) making the request. To indicate the desired ECHONET Object instance via the Basic API, an ECHONET Address, an ECHONET Object class code, and its instance code are required. From the viewpoint of the application, ECHONET Objects are seen in the relationship shown by Node B in Fig. 3.2-5 within the ECHONET Communication Middleware.

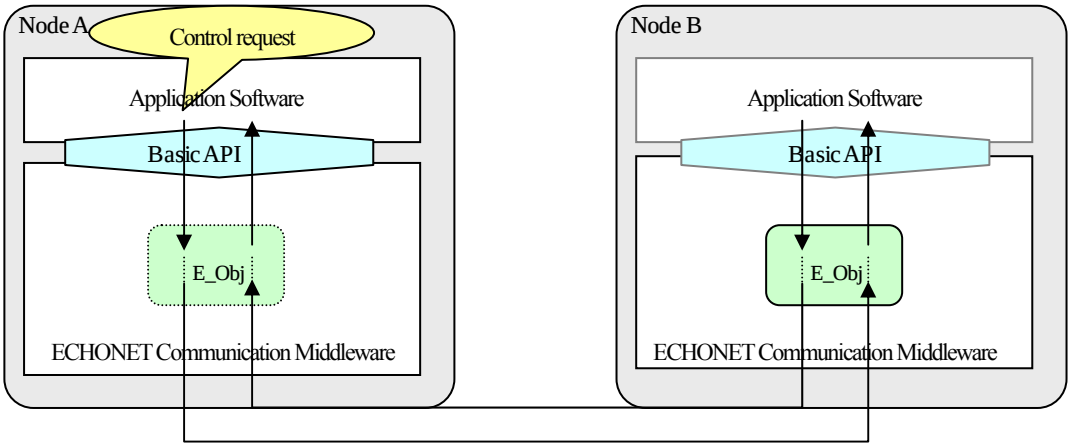


Fig. 3.2-4

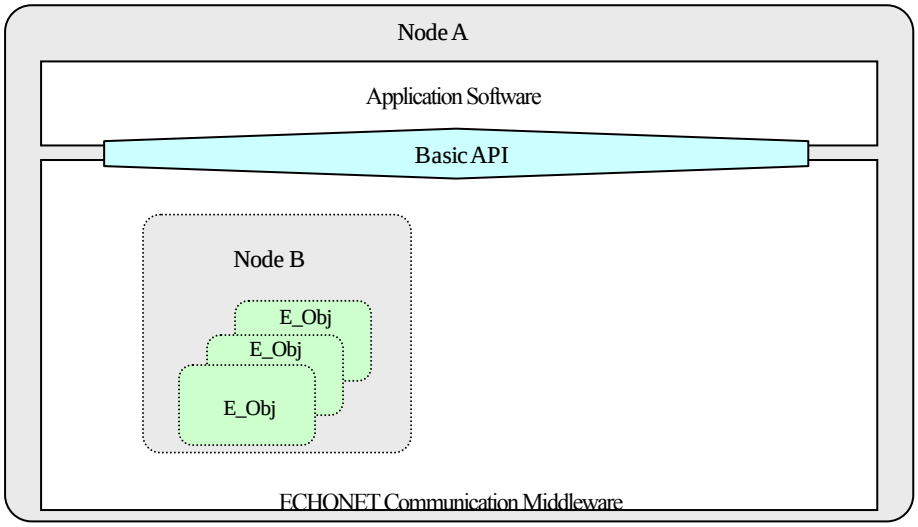


Fig. 3.2-5

(3) ECHONET Objects when notifying another node of self-node status

ECHONET provides two methods for notifying application software on another node of the status of the self-node. These methods are shown in Figs. 3.2-6 and 3.2-7. In the method shown in Fig. 3.2-6, when a request is received from an application, the specified other node (Node B) is immediately notified, and the device status need not be stored as an object in the ECHONET Communication Middleware for the node (Node A) announcing the status. In the second method, shown in Fig. 3.2-7, upon receiving a request from an application, the ECHONET Communication Middleware periodically sends notification of the property value to the other node using asynchronous timing that differs from the request from the application. Here, ECHONET Object data actually exists in the ECHONET Communication Middleware. In the former method (Fig. 3.2.6), however, because communication is stipulated by the application, a virtual copy of the ECHONET Objects exists in the ECHONET Communication Middleware. In either case, from the viewpoint of the application, the ECHONET objects of the self-node are seen as existing within the ECHONET Communication Middleware (see Fig. 3.2-8).

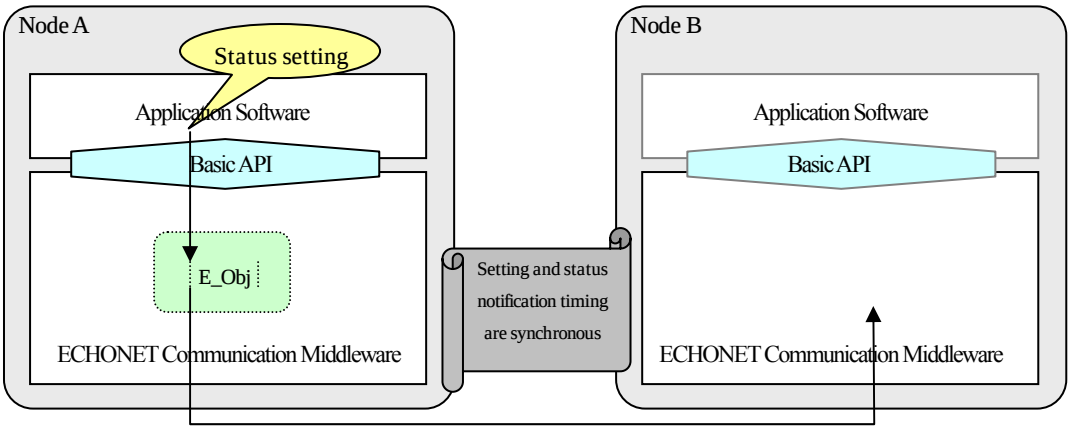


Fig. 3.2-6

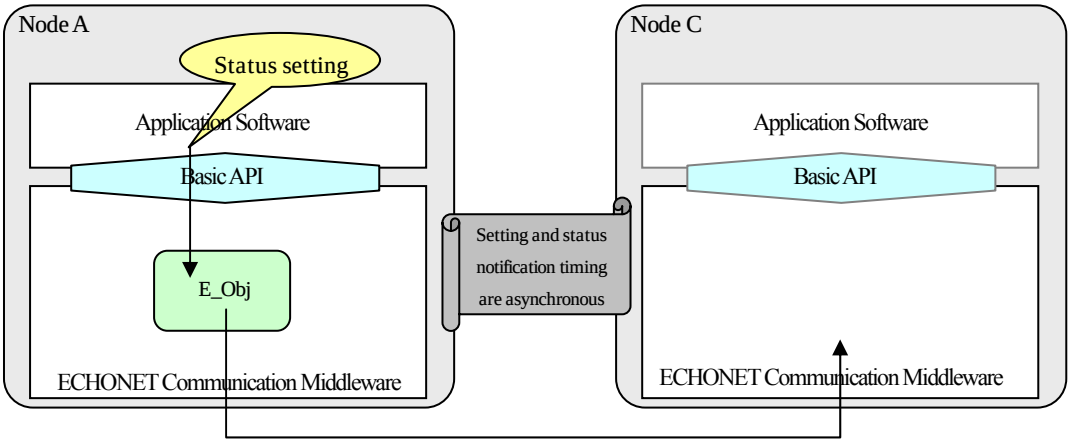


Fig. 3.2-7

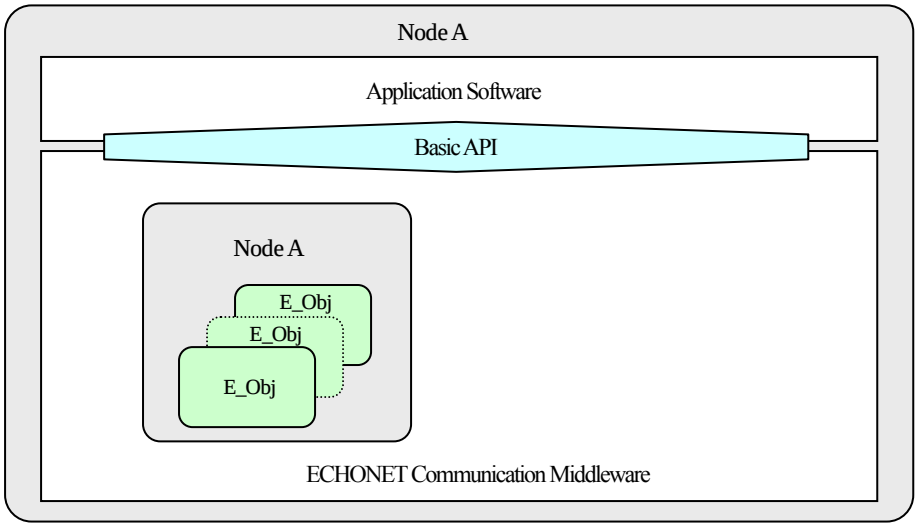


Fig. 3.2-8

As is clear from the three cases shown above, the ECHONET Communication Middleware is viewed by the application software as containing (and in some cases actually does contain) (1) a collection of ECHONET objects of the self-node whose role is to disclose the functions of the self-node to other nodes and to be controlled by other nodes; and (2) ECHONET objects at the node level whose role is to control and obtain the status of the functions of other nodes. Here, the "Self-device" shall be specified as the unit for a collection of ECHONET object instances showing the functions of the self-node. Only one such device exists in each piece of ECHONET Communication Middleware, but there may be as many other devices as there are other related nodes.

Based on the above, Fig. 3.2-9 shows a typical ECHONET Communication Middleware object configuration for a system in which an air conditioner, ventilation fan, and human detection sensor are connected as separate nodes via a network, seen from the perspective of the application software in the air conditioner.

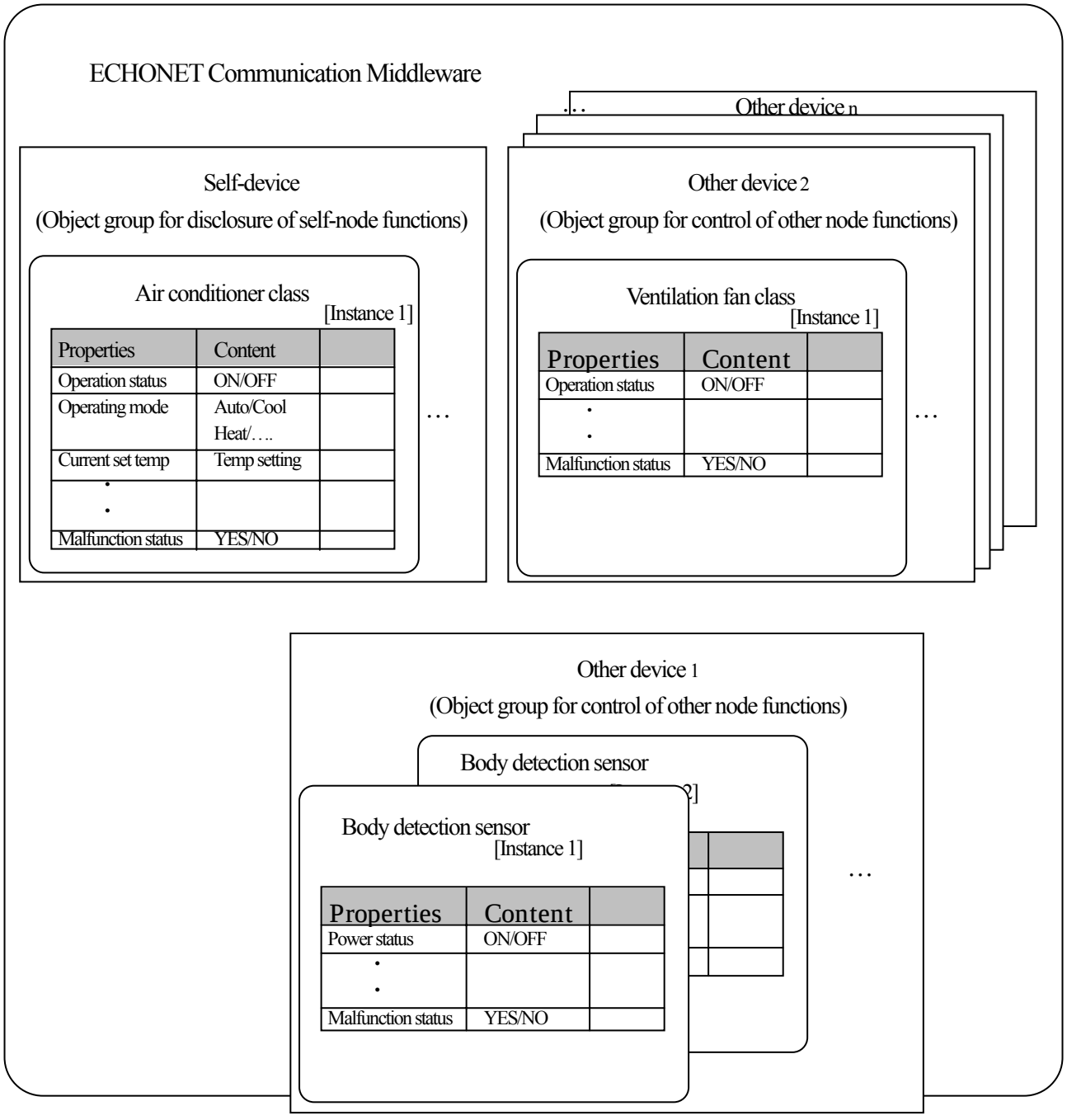


Fig. 3.2-9

Chapter 4 Message Structure (Frame Format)

4.1 Basic Concept

The ECHONET specifications were designed to enable the use of power line and wireless protocols as transmission media. Normally, noise and phasing have a major impact on power line and 400-MHz wireless used within the home. Moreover, slow transmission speeds discourage large data transfers, and it is desirable to reduce the mounting load on simple devices. In light of this situation, ECHONET specifies the frame format for the ECHONET Communication Middleware Block to minimize message size while fulfilling the requirements of the communications layer structure.

4.2 Frame Format

Figures 4.1-1 and 4.1-2 show the content of the ECHONET Communication Middleware frame format. Detailed specifications for each message component are provided on the following pages.

(1) Message configuration for exchange between ECHONET Communications Processing Blocks

In the ECHONET Communication Middleware Specifications, messages exchanged between ECHONET Communications Processing Blocks are called ECHONET frames. ECHONET frames are roughly divided into two types depending on the specified EHD: secure message format whose EDATA section is enciphered (see Chapter 10) and plain message format whose EDATA section is not enciphered. The secure message format and plain message format are subdivided into three formats depending on the specified EHD (see Section 4.2.1). Therefore, the following six different message formats are available for ECHONET frames:

① Plain basic message format
Insecure communication is performed so that one message is used to view or change the contents of one property.

② Plain compound message format
Insecure communication is performed so that one message is used to view or change the contents of two or more properties.

③ Plain arbitrary message format
Insecure communication is performed so as to exchange information that complies with vendor-unique specifications.

④ Secure basic message format
Secure communication is performed so that one message is used to view or change the contents of one property.

⑤ Secure compound message format
Secure communication is performed so that one message is used to view or change the contents of two

or more properties.

⑥ Secure arbitrary message format

Secure communication is performed so as to exchange information that complies with vendor-unique specifications.

Figure 4.1-1 shows the ECHONET frame structure for the plain message format. Figure 4.1-2 shows the ECHONET frame structure for the secure message format.

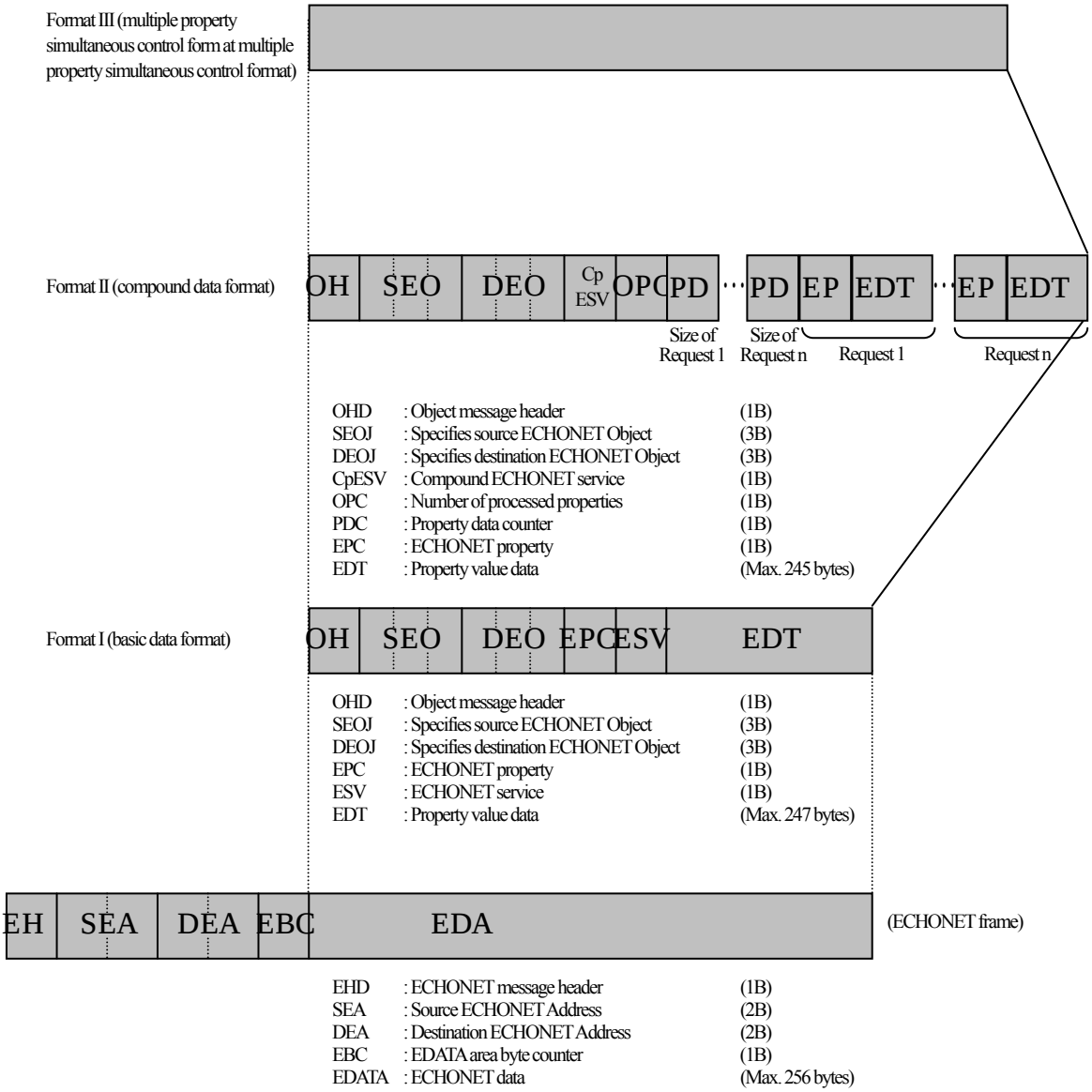
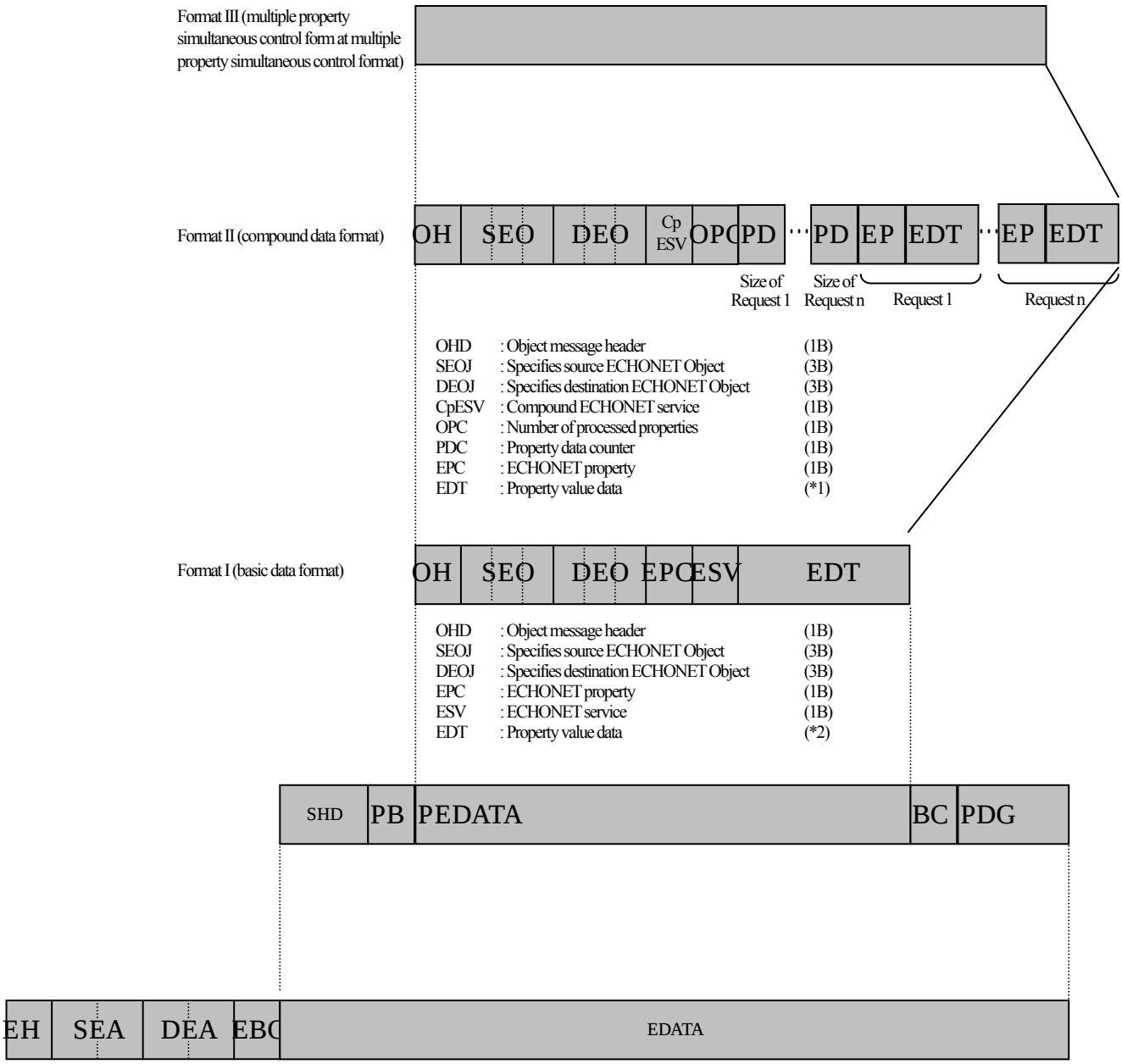


Fig. 4.1-1 ECHONET Frame for Plain Data Format



Note: Wavy-lined areas are to be enciphered (see Chapter 10).
(*1) The maximum value shall be 235 bytes when encryption is used and 223 bytes when encryption and authentication are used.
(*2) The maximum value shall be 237 bytes when encryption is used and 225 bytes when encryption and authentication are used.
(*3) This is included in the frame format when encryption and authentication are used or when encryption is used without authentication. This is not included when authentication is used without encryption.
(*4) This is included in the frame format when encryption and authentication are used or when authentication is used without encryption. This is not included when encryption is used without authentication.

Fig. 4.1-2 ECHONET Frame for Secure Message Format

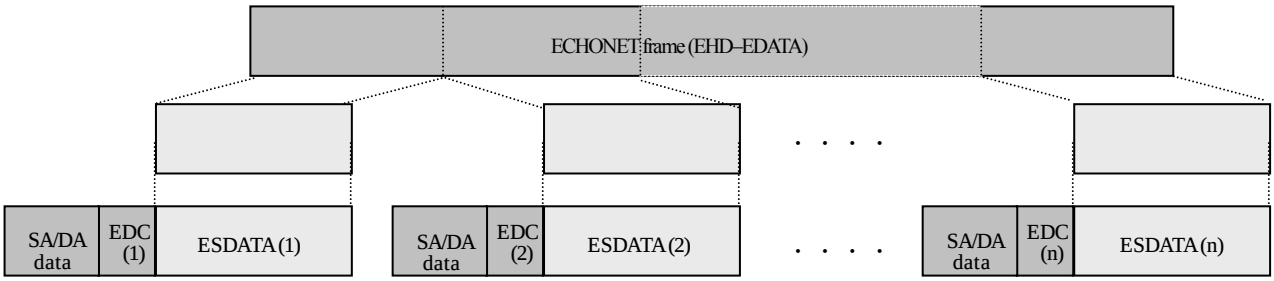
(2) Format for messages exchanged between Protocol Difference Absorption Processing Sections

In these Specifications, messages exchanged between Protocol Difference Absorption Processing Sections shall be called "ECHONET split frames." The composition of these messages allows differences in message size to be absorbed so that the processing at the ECHONET communication processing section is independent of the lower-layer communications software.

N : Number of split frames (max. 18)
EDC (n) : ECHONET message counter (1 byte)
ESDATA (1)-(n): Message from EHD–EDATA (ECHONET frame) split into n parts. Max. 262 bytes.
SA/DA data : DA (recipient's MAC Address data) when sending message, SA (source's MAC Address data) when receiving message
When sending, DA data is created from DEA value in ECHONET frame
Includes broadcast specification data

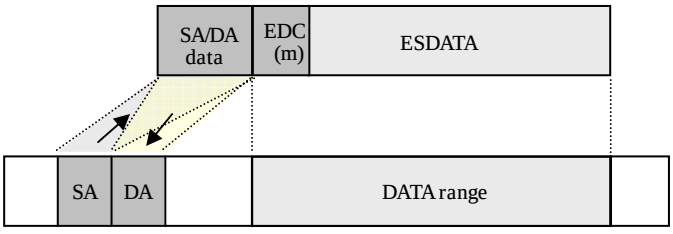
Relationship with upper-layer messages

The ECHONET split frame described above consists of an ECHONET frame that has been split into messages no larger than the maximum processable size for Lower-Layer Communications Software. Each one also contains header codes (EDC) for frame splitting, assembly, and routing as well as address data for the source and destination.



Relationship with lower-layer messages

ECHONET split frames



SA: MAC address of the source of messages between lower-layer communications software (dependent on lower-layer communications software)
DA: MAC address of destination of messages between lower-layer communications software (dependent on lower-layer communications software)
DATA area: Actual message to be exchanged between lower-layer communications software

Fig. 4.1-3 ECHONET Split Frames

ECHONET transmission frame

The "EDC (n)" and "ESDATA (n), which are components of the ECHONET split frame (n), are stored as the lower-layer communications software payload. In these Specifications, this "EDC (n) + ESDATA (n)" shall comprise an "ECHONET transmission frame." When the number of ECHONET split frames exchanged between Protocol Difference Absorption Processing Sections is 1 (i.e. no splitting), ESDATA becomes the ECHONET frame and "EDC (1) + ECHONET frame" becomes the ECHONET transmission frame.

n: The number of divided pieces of the ECHONET frame (max. = 8)

EDC (n): ECHONET message counter (1B)

ESDATA (1) to (n): The data of the message from EHD to EDATA (ECHONET frame) that was divided by n. The total shall be 262B.

* ECHONET transmission frames do not contain SA/DA data.

4.2.1 ECHONET Headers (EHD)

This section provides detailed specifications for the ECHONET Header (EHD) shown in Fig. 4.1-1 and Fig. 4.1-2.

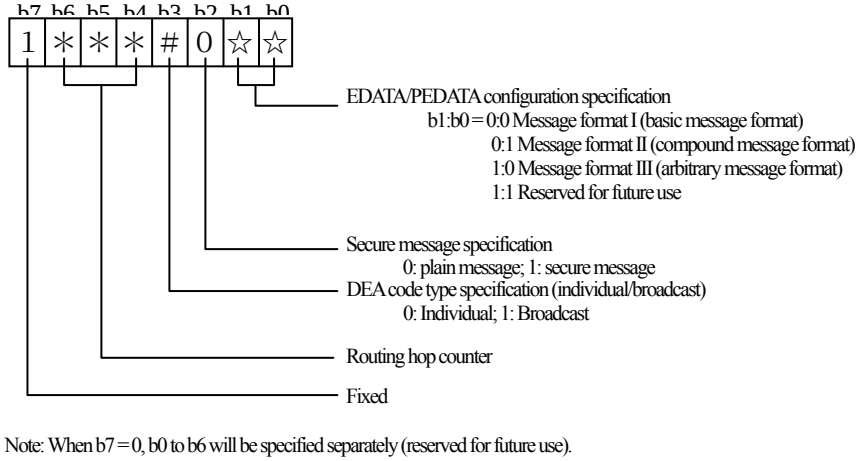


Fig. 4.2 EHD Detailed Specifications

The combination of b1 and b0 specifies the message format for the EDATA/PEDATA section. When b1:b0 = 0:0, it indicates Message Format I (basic message format), which allows one message to operate on one property of one object. When b1:b0 = 0:1, it indicates Message Format II (compound message format), which allows one message to operate on two or more properties of one object. When b1:b0 = 1:0, it indicates Message Format III (arbitrary message format), whose EDATA/PEDATA section is in an arbitrary format.

Bit b2 indicates whether or not the EDATA section is enciphered. When b2 = 1, it means that the EDATA section is enciphered. When b2 = 0, it means that the EDATA section is not enciphered. Detailed information about enciphered and other secure messages is set forth in Chapter 10.

Bit b3 specifies whether the DEA (destination ECHONET address) shown in Figs. 4.1-1 and 4.1-2 is a broadcast address or an individual address. When b3 = 1, it indicates that a broadcast address is stipulated by the DEA code. When b3 = 0, it indicates that an individual address is stipulated by the DEA code. Broadcast address codes are discussed in the next section.

Bits b4, b5, and b6 constitute a routing hop counter, which can be manipulated only by ECHONET Routers. When a message received at one subnet of an ECHONET Router is forwarded to another subnet, the counter is incremented. For every transmission from an ordinary node, a hop count of 0 is used. The relationship between b4, b5, and b6 and the hop count is shown in the table on the next page. The number of hops can be set to a value between 0 and 7.

b6	b5	b4	Hop count (router passes)
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

4.2.2 Source/Destination ECHONET Address (SEA/DEA)

This section provides detailed specifications for the source ECHONET address (SEA) and destination ECHONET address (DEA) shown in Figs. 4.1-1 and 4.1-2. Figure 4.3 shows the configuration of the source ECHONET address (SEA) and the destination ECHONET address (DEA) prevailing when an individual address is stipulated by setting b3 of EHD to 0 (see Chapter 2 for details).

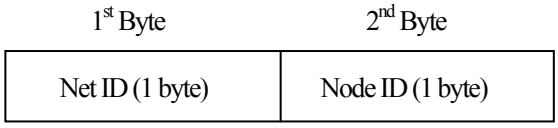
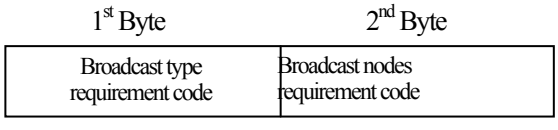


Fig. 4.3 Configuration of SEA and DEA When Individual Address Is Specified

When b3 of EHD is set to 1 to specify a broadcast, the destination ECHONET address (DEA) becomes a code indicating a broadcast message for specific ECHONET address groups (including a general broadcast). The DEA configuration in this case is shown in Fig. 4.4. The broadcast target requirement code is shown in Figs. 4.5-1 and 4.5-2.



Broadcast type requirement code	Broadcast target requirement code	Remarks
0x00	Specifies the node groups to be targeted for a broadcast within all subnets. For node group selection, see Fig. 4.5.	An intra-domain broadcast. In all subnets within a domain, a broadcast is sent to the nodes stipulated by the broadcast target requirement code.

0x01	Specifies the node groups to be targeted for a broadcast within own subnet. For node group selection, see Fig. 4.5.	An intra-own-subnet broadcast. Within own subnet, a broadcast is sent to the nodes stipulated by the broadcast target requirement code.
0x02	All nodes within the subnet having the NetID code stipulated by the "broadcast target requirement code" are targeted.	A general broadcast within a specified subnet. A broadcast is sent to all nodes within the subnet stipulated by the broadcast target requirement code.
0x03	Specifies the group broadcast numbers for the target groups in all subnets for a broadcast. For group broadcast number setup, refer to "9.11.1 (27) Group broadcast number."	Intra-domain group broadcasting. A broadcast message is sent in all subnets within the domain to the nodes specified by the broadcast target code. This function is optional. In the case where this function is not used, the group broadcast number property described in "9.11.1 (27) Group broadcast number" need not be implemented. However, in the case where the intra-domain group broadcast function is implemented, the intra-home subnet group broadcast function must also be implemented.
0x04	Specifies the group broadcast numbers for the target groups in the home subnet for a broadcast. For group broadcast number setup, refer to "9.11.1 (27) Group broadcast number."	Intra-home subnet group broadcasting. A broadcast message is sent in the home subnet to the nodes specified by the broadcast target code. This function is optional. In the case where this function is not used, the group broadcast number property described in "9.11.1 (27) Group broadcast number" need not be implemented. However, in the case where the intra-home subnet group broadcast function is implemented, the intra-domain group broadcast function must also be implemented.
0x05–0x7F	for future reserved	
0x80–0xFF	Open to user	Used when a system manager will manage the system in a collective housing unit or small office building.

Fig. 4.4 DEA (Broadcast-Stipulated) Address Configuration

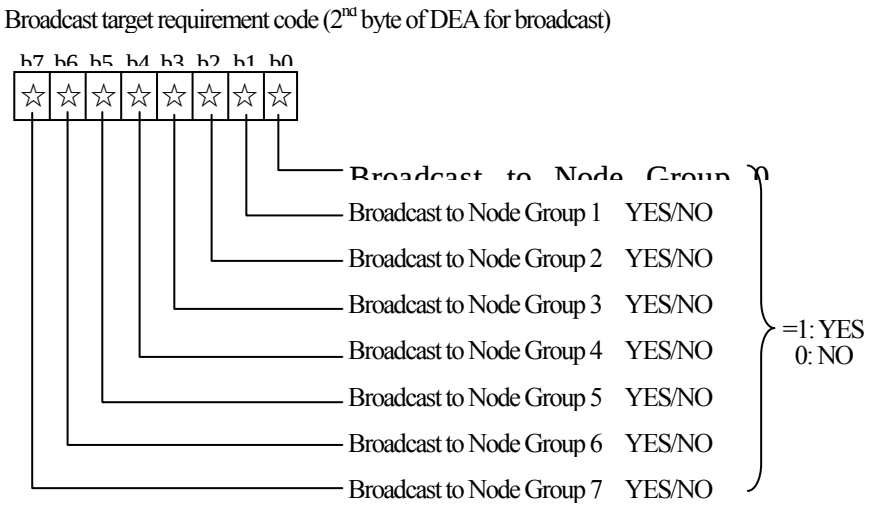


Fig. 4.5-1 Broadcast Target Requirement Code

Note: The node IDs of the nodes belonging to Node Groups 0 to 7 are as indicated below. For example, a node whose ID is 0xA2 belongs to Group 2.

Upper \ Lower	0	8	4	C	2	A	6	E	1	9	5	D	3	B	7	F	
0																	Group 0
8																	
4																	Group 1
C																	
2																	Group 2
A																	
6																	Group 3
E																	
1																	Group 4
9																	
5																	Group 5
D																	
3																	Group 6
B																	
7																	Group 7
F																	

Fig. 4.5-2 Node Group Requirement Bit Specifications

4.2.3 ECHONET Byte Counter (EBC)

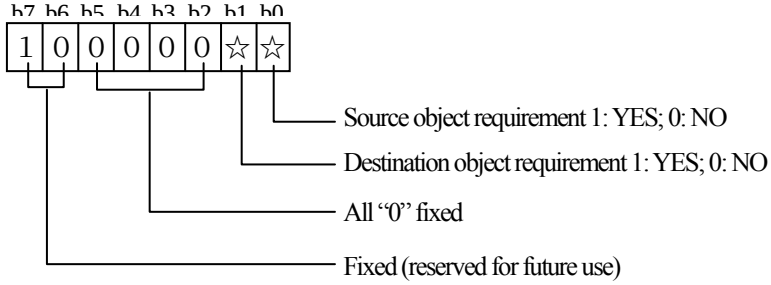
Indicates the size of the ECHONET data region (EDATA region) shown in Figs. 4.1-1 and 4.1-2. The size is variable in 1-byte increments. The acceptable EDATA region size ranges from 6 to 256 bytes (0x06 to 0xFF; 0x00 = 256). The lower limit is 6 bytes, which indicates that a message consists of at least 6 bytes. The reason for this is that either the SEOJ or DEOJ needs to be specified with the EPC to ESV options specified for a plain message. A 6-byte message can be a message requesting an ESV with the DEOJ specified or a message carrying a response of "processing impossible" for ESV with the SEOJ specified.

4.2.4 ECHONET Data (EDATA)

The DATA region for messages exchanged by ECHONET Communication Middleware. Maximum size: 256 bytes.

4.2.5 Object Message Header (OHD)

This section provides detailed specifications for the Object Message Header (OHD) shown in Figs. 4.1-1 and 4.1-2. The state in which b1 and b0 are both 0 will never occur.

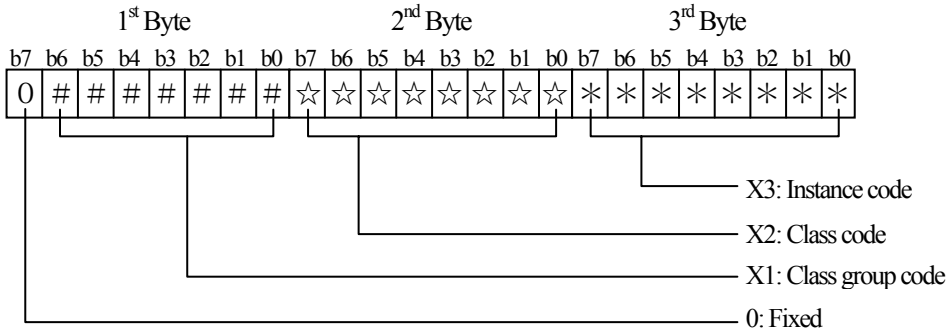


Notes: When b6 and b7 have values other than b6 = 0 and b7 = 1, b0–b5 will have different meanings.
The meanings of bits b0 to b5 when b6 and b7 have values other than b6 = 0 and b7 =1 are to be stipulated in the future (reserved for future use).

Fig. 4.6 OHD Detailed Specifications

4.2.6 ECHONET Objects (EOJ)

This section provides detailed specifications for the source ECHONET object (SEOJ) code and destination ECHONET object (DEOJ) code shown in Figs. 4.1-1 and 4.1-2.



Note: The meanings of the bits when b7 of the 1st byte is 1 are to be stipulated in the future (reserved for future use).

Fig. 4.7 EOJ Detailed Specifications

ECHONET objects are described using the formats [X1.X2] and [X3], to be specified as shown below. (However, "." is used only for descriptive purposes and does not mean a specific code.) The object class is designated by the combination of X1 and X2, while X3 shows the class instance. A single ECHONET node may contain more than one instance of the same class, in which case X3 is used to identify each one.

The specific items in Tables 4.1–4.8 were specified based on JEM-1439. Detailed specifications for the objects shown here will be developed over time, and during this phase specifications for the objects themselves (i.e., present/not present) will be further reviewed. Objects for which detailed specifications (including property configurations) have already been formulated will be indicated with a ○ in the Remarks column, with the detailed specifications to be provided in the Appendix.

The instance code 0x00 is regarded as a special code (code for specifying all instances). When a DEOJ having this specified code is received, it is handled as a code specifying a broadcast to all instances of a specified class.

・X1	: Class group code	0x00–0x7F. For details, refer to Table 4.1.
・X2	: Class code	0x00–0xFF. For detailed examples, refer to Tables 4.2–4.8.
・X3	: Instance code	0x00–0xFF. Identifier code used when more than one of the same class specified by [X1.X2] exists within the same node. However, 0x00 is used for general broadcast to all instances of class specified with [X1.X2].

Table 4.1 List of Class Group Codes

class group CODE	GROUP NAME	Remarks
0x00	Sensor-related device class group	
0x01	Air conditioner-related device class group	
0x02	Housing/facility-related device class group	Includes lighting
0x03	Cooking/housework-related device class group	
0x04	Health-related device class group	
0x05	Management/control-related device class group	
0x06	AV-related device class group	
0x07–0x0C	Reserved for future use	
0x0D	Service class group	
0x0E	Profile class group	
0x0F	User definition class group	
0x10–0x1F	Communications definition class group for requirement of status notification method	
0x20–0x2F	Communications definition class group for requirement of setting control reception method	
0x30–0x3F	Communications definition class group for linked settings (action settings)	
0x40–0x4F	Communications definition class group for linked settings (trigger settings)	
0x50–0x5F	Secure communication access property setup class	
0x60–0x7F	Reserved for future use	
0x80–0x8F	Property status referencing request compliability communication definition class group	
0x90–0x9F	Property status notifiability communication definition class group	

Table 4.2 Class Code List (Class Group Code X1 = 0x00)
For details, refer to Table 1.1 in the Appendix.

Table 4.3 Class Code List (Class Group Code X1 = 0x01)
For details, refer to Table F.7 in Appendix 1.2.

Table 4.4 Class Code List (Class Group Code X1 = 0x02)
For details, refer to Table 1.3 in the Appendix.

Table 4.5 Class Code List (Class Group Code X1 = 0x03)
For details, refer to Table 1.4 in the Appendix.

Table 4.6 Class Code List (Class Group Code X1 = 0x04)
For details, refer to Table 1.5 in the Appendix.

Table 4.7 List of Class Codes for Class Group Code (X1 = 0x05)

class code	CLASS NAME	DetaILED Specifications	REMARKS
0x00–0xFC	Reserved for future use		
0xFC	Secure communication common key setup node	●	Detailed specifications for this class are given in Part 2, Paragraph 9.11.1.
0xFD	Switch		
0xFE	Portable terminal		
0xFF	Controller		

(Note) ●: Details, including the property configuration, are specified in Part 2.

Table 4.8 List of Class Codes for Class Group Code (X1 = 0x0E)

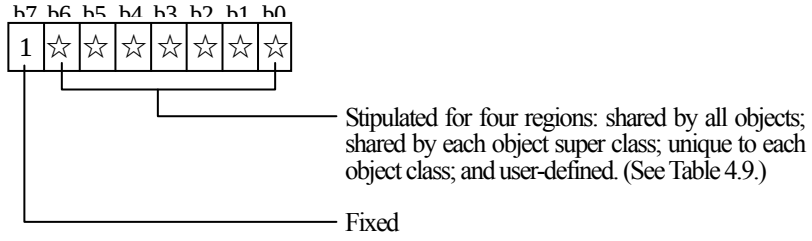
class code	CLASS NAME	DetaILED Specifications	REMARKS
0x00–0xEF	Reserved for future use		
0xF0	Node profile	●	Detailed specifications for this class are given in Part 2, Paragraph 9.11.1
0xF1	Router profile	●	Detailed specifications for this class are given in Part 2, Paragraph 9.11.2
0xF2	ECHONET Communications Processing Block profile	●	Detailed specifications for this class are given in Part 2, Paragraph 9.11.3
0xF3	Protocol Difference Absorption Processing Block profile	●	Detailed specifications for this class are given in Part 2, Paragraph 9.11.4
0xF4	Lower-layer media profile	●	Detailed specifications for this class are given in Part 2, Paragraph 9.11.5
0xF5–0xFF	Reserved for future use		

(Note) ●: Details, including the property configuration, are specified in Part 2.

Table 4.9 Class Code List (Class Group Code X1 = 0x0D)
For details, refer to Table 1.1 in Part VIII, Section 1.3.

4.2.7 ECHONET Properties (EPC)

This section provides detailed specifications for the ECHONET property (EPC) code shown in Figs. 4.1-1 and 4.1-2. The EPC specifies a service target function. Each object stipulated by X1 (class group code) and X2 (class code), described in the previous section, is specified here. (When a specified object changes, the target function also changes even when the code remains unchanged. However, the detailed specifications are designed to ensure that, whenever possible, the same functions will have the same code.) Specific code values for each object are stipulated in Chapter 9 and in the Appendix. These codes correspond to the object property identifiers in the object definitions.



Note: When b7 = 0, the other bits will be defined differently.

Fig. 4.8 EPC Detailed Specifications

Table 4.10 EPC Code Allocation Table

	8	9	A	B	C	D	E	F
0								
1								
2								
3								
4								
5								
6								
7	Region shared by all object classes		Region shared by each class group*2		Region unique to each class*2			User-defined*1
8								
9								
A								
B								
C								
D								
E								
F								

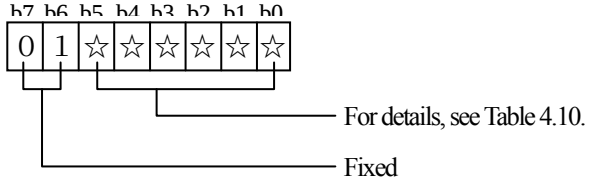
←b7–b4 values (hex)

↑ b3–b0 values (hex)

Notes: 1) Stipulated for each user. In the case of a user-defined object class, 0xA to 0xF in the four high-order bits (b7 to b4) are user-defined.
2) As a rule these two regions are used, but in practice the boundary line will change for each class group. Individual regions will be specified in the object class detailed specifications in the Appendix and in Chapter 9.

4.2.8 ECHONET Service (ESV)

This section provides detailed specifications for the ECHONET service (ESV) code shown in Figs. 4.1-1 and 4.1-2.



Note: In cases other than when b7:b6 = 0:1, the meaning of values b0–b5 will be specified separately.

Fig. 4.9-1 ESV Detailed Specifications

This code stipulates manipulation of the properties stipulated by the EOJ. The three main kinds of operations are shown below. There are also two kinds of responses: the "response," which is given when the stipulated properties exist; and the "response not possible," which is given when the requested properties (including array elements) do not exist or when the stipulated service cannot be processed.
"Request" / "Response" (response/response not possible) / "Notification"

A "response" is considered a reply to a "request" that requires a response; when the object stipulated in the DEOJ exists, it is generally either "response" or "response not possible" (stipulated processing cannot be accepted, or the stipulated object exists but the property does not). When the request requires no response and the stipulated object does not exist, no response is made.
There are two types of "notification": one for transmitting own property information autonomously and the other for sending a response to a notification request. However, these two types have the same code.
Three specific operations are provided: write (response required/no response required), read, and notification (notification/notification with response required). The 12 operations shown below are set in consideration of whether or not the content of the given property is an array.

① Property value write (response required/no response required)
② Property value read
③ Property value notification
④ Property value array-element-stipulated write (response required/no response required)
⑤ Property value array-element-stipulated read
⑥ Property value array-element-stipulated notification
⑦ Property value array-element-stipulated addition (response required/no response required)
⑧ Property value array-element-stipulated deletion (response required/no response required)
⑨ Property value array-element-stipulated existence confirmation

⑩ Property value array element addition (response required/no response required)

⑪ Property value notification (response required)

⑫ Property value array-element-stipulated notification (response required)

The relationship between message configuration (presence or absence of SEOJ and DEOJ) and EPC and ESV is described below.

[1] The EPC in an ECHONET message stipulating only SEOJ indicates the properties of the sender object specified in the SEOJ. Here, the ESV contains an autonomous "notification" or "notification" or "response" in response to a request for properties specified in the SEOJ and EPC. If the ESV is a "request" in such a case, the received message is treated as an illegal message.

[2] The EPC in an ECHONET message stipulating only DEOJ indicates the properties of the destination object specified in the DEOJ. Here, the ESV contains a "request" regarding the properties specified in the DEOJ and EPC. If the ESV is a "response" or a "notification" in such a case, the received message is treated as an illegal message.

[3] For ECHONET messages stipulating both SEOJ and DEOJ, the ESV value is used to determine whether the EPC is stipulated by SEOJ or DEOJ. When the ESV is a "response" or a "notification", the EPC is considered to be a component of the object specified by the SEOJ and is viewed as a "response" or "notification" directed towards the object stipulated in the DEOJ. When the ESV is a "request," the EPC is considered to be a component of the DEOJ and is viewed as a "request" from the object stipulated in the SEOJ.

Tables 4.11-1 through 4.11-3 show specific ESV code assignments based on the content described above. Specific descriptions of ① through ⑫ above are provided in (1) through (12). (The related number is indicated in the Remarks column of the table.) In the diagrams in (1) through (12), the DEOJ values used in relation to "requests" are individually specified codes. However, although a service request is made to two or more nonspecific object instances using a single message when the DEOJ value indicates broadcasting to all instances of the specified class (i.e. X3 = 0x00), the processing in such a case shall assume that a request message was sent individually to each instance. That is, when it is necessary to send response messages, they shall be generated in such a manner that the number of instances equals the number of response messages, and messages with contents that match the individual instances shall be sent after storing such contents.

Note that in the table, the "array elements" described above are presented as "elements." Figure 4.9-2 provides a sequence diagram of the relationship between individual ESVs.

Table 4.11-1 List of ESV Codes for Requests

Service Code (ESV)	ECHONET Service Content	Symbol	Remarks
0x60	Property value write request (no response required)	SetI	(1); broadcasting permitted
0x61	Property value write request (response required)	SetC	
0x62	Property value read request	Get	(2); broadcasting permitted
0x63	Property value notify request	INF_REQ	(3); broadcasting permitted
0x64	Property value element-stipulated write request (no response required)	SetMI	(4); broadcasting permitted
0x65	Property value element-stipulated write request (response required)	SetMC	
0x66	Property value element-stipulated read request	GetM	(5); broadcasting permitted
0x67	Property value element-stipulated notify request	INFM_REQ	(6); broadcasting permitted
0x68	Property value element-stipulated add request (no response required)	AddMI	(7); broadcasting permitted
0x69	Property value element-stipulated add request (response required)	AddMC	
0x6A	Property value element-stipulated delete request (no response required)	DellMI	(8); broadcasting permitted
0x6B	Property value element-stipulated delete request (response required)	DellMC	
0x6C	Property value element existence confirm request	CheckM	(9); broadcasting permitted
0x6D	Property value element add request (no response required)	AddMSI	(10); broadcasting permitted
0x6E	Property value element add request (response required)	AddMSC	
0x6F	Reserved for future use		

Table 4.11-2 List of ESV Codes for Response/Notification

Service Code (ESV)	ECHONET Service Content	Symbol	Remarks
0x71	Property value write response	Set_Res	ESV = 0x61 response (1); individual response
0x72	Property value read response	Get_Res	ESV = 0x62 response (2); individual response
0x73	Property value notification	INF	*1 (3); individual notification or broadcast notification
0x74	Property value notification (response required)	INFC	(11) individual notification
0x75	Property value element-stipulated write response	SetM_Res	ESV = 0x65 response (4); individual response
0x76	Property value element-stipulated read response	GetM_Res	ESV = 0x66 response (5); individual response
0x77	Property value element-stipulated notify	INFM	*2 (6); individual notification or broadcast notification
0x78	Property value element-stipulated notify (response required)	INFMC	(12); individual notification
0x79	Property value element-stipulated add response	AddM_Res	ESV = 0x69 response (7); individual response
0x7A	Property value notify response	INFC_Res	ESV = 0x74 response (11); individual response
0x7B	Property value element-stipulated delete response	DelM_Res	ESV = 0x6B response (8); individual response
0x7C	Property value element-stipulated existence confirm response	CheckM_Res	ESV = 0x6C response (9); individual response
0x7D	Property value element-stipulated notify response	INFMC_Res	ESV = 0x78 response (12); individual response
0x7E	Property value element add response	AddMS_Res	ESV = 0x6E response (10); individual response
0x70, 0x7F	Reserved for future use		

Notes: *1 Used for autonomous property value notification and for 0x63 response.

*2 Used for autonomous property value notification and for 0x67 response.

Table 4.11-3 List of ESV Codes for "Response Not Possible" Responses

Service Code (ESV)	ECHONET Service Content	Symbol	Remarks
0x50	Property value write "process not possible" response	SetI_SNA	ESV = 0x60 response not possible (1); individual response
0x51	Property value write "process not possible" response	SetC_SNA	ESV = 0x61 response not possible (1); individual response
0x52	Property value read "process not possible" response	Get_SNA	ESV = 0x62 response not possible (2); individual response
0x53	Property value notify "process not possible" response	INF_SNA	ESV = 0x63 response not possible (3); individual response
0x54	Property value element-stipulated write request "process not possible" response	SetMI_SNA	ESV = 0x64 response not possible (4); individual response
0x55	Property value element-stipulated write request "process not possible" response	SetMC_SNA	ESV = 0x65 response not possible (4); individual response
0x56	Property value element-stipulated read request "process not possible" response	GetM_SNA	ESV = 0x66 response not possible (5); individual response
0x57	Property value element-stipulated notify request "process not possible" response	INFM_SNA	ESV = 0x67 response not possible (6); individual response
0x58	Property value element-stipulated add request "process not possible" response	AddMI_SNA	ESV = 0x68 response not possible (7); individual response
0x59	Property value element-stipulated add request "process not possible" response	AddMC_SNA	ESV = 0x69 response not possible (7); individual response
0x5A	Property value element-stipulated delete request "process not possible" response	DelMI_SNA	ESV = 0x6A response not possible (8); individual response
0x5B	Property value element-stipulated delete request "process not possible" response	DelMC_SNA	ESV = 0x6A response not possible (8); individual response
0x5C	Property value element-stipulated existence confirm request "process not possible" response	CheckM_SNA	ESV = 0x6C response not possible (9); individual response
0x5D	Property value element add request "process not possible" response	AddMSI_SNA	ESV = 0x6D response not possible (10); individual response
0x5E	Property value element add request "process not possible" response	AddMSC_SNA	ESV = 0x6E response not possible (10); individual response
0x5F	Reserved for future use		

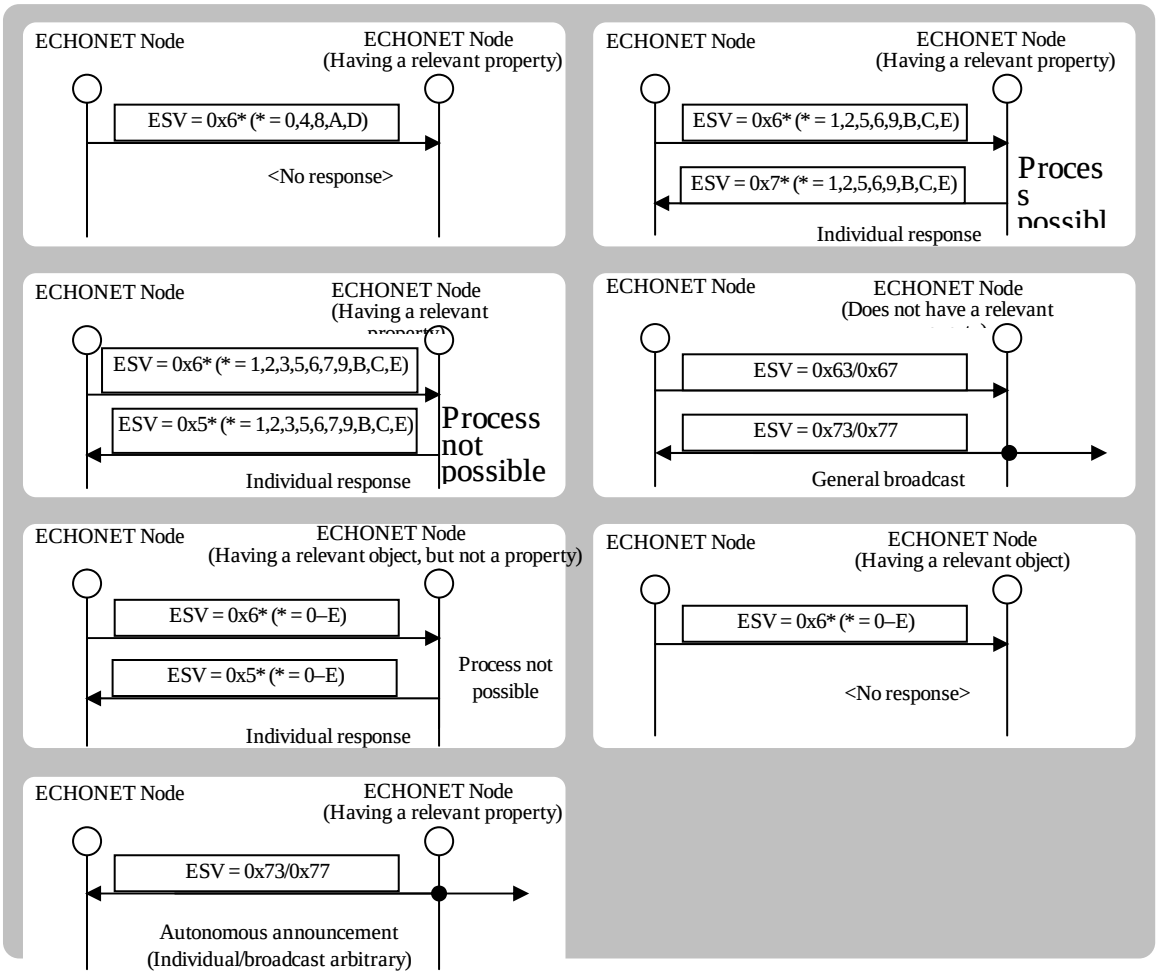
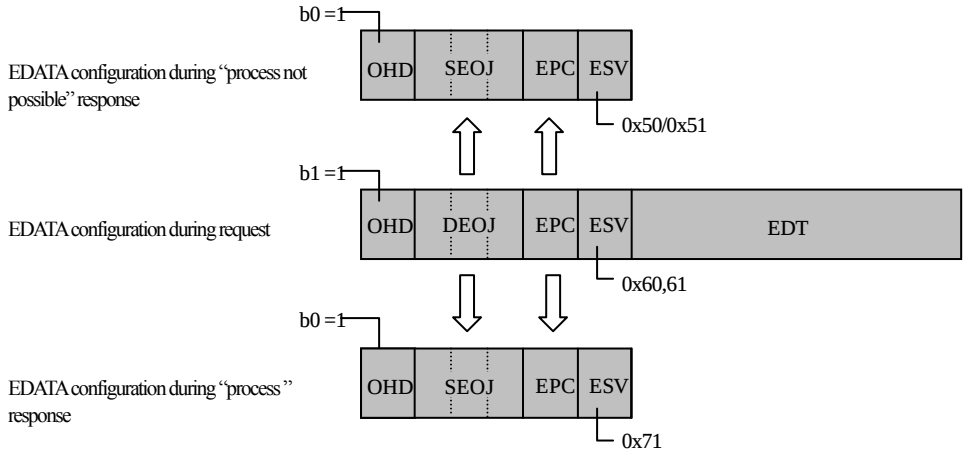


Fig. 4.8-2 Service-related Basic Sequence Diagram

(1) Property value write service [0x60, 0x61, 0x71, 0x50, 0x51]

In the case of a "request" (0x60, 0x61), this indicates a request to write the content shown in the EDT to the property stipulated in the EPC of the DEOJ-stipulated object. In response to this "request," when a value indicating a response is stipulated (0x61) and the request is to be (or has already been) accepted, a "response" (0x71) is returned. This "response" is not a processing implementation response. When the request is not to be accepted, or when the stipulated DEOJ exists but the stipulated EPC does not exist, "response not possible" (0x50, 0x51) is returned. In the response frame format, SEOJ represents the value of the object stipulated by the request, and the relevant property is set in the EPC. When the relevant object itself does not exist, neither "response" nor "response not possible" is returned. (See Fig. 4.8-2 for the exchange procedure.) Also, the "response" message DEA is defined as the requesting entity (i.e., the request message SEA).



When EDATA stipulates SEOJ during a "request," the EOJ stipulated by SEOJ in EDATA during the "request" is allocated as a DEOJ (b1 of the OHD is also set to 1), in the case of both "response not possible" and "response."

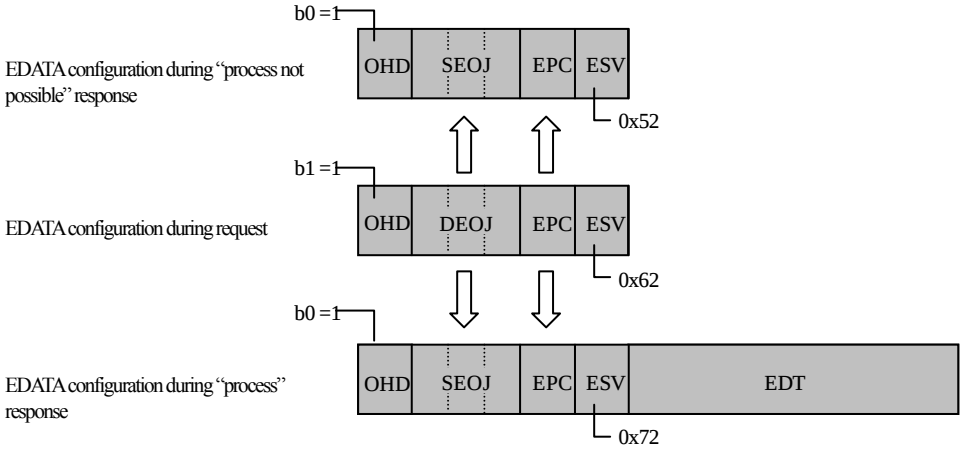
(2) Property value read service [0x62, 0x72, 0x52]

In the case of a "read" (0x62), this indicates a request to read the content of the property stipulated in the EPC of the DEOJ-stipulated object.

If the property specified by the EPC is an array property, the requested node reads the array property content, from the top position to the end, in units of the array element size and arrays them in the same order (starting with Element 0) to use as the EDT for the response message. If the total data size of the array property elements exceeds the maximum EDT size, an EDT shall be generated after discarding the excess portion. The parts that correspond to areas with no array element, if any, shall not be left vacant but shall instead be used for the succeeding array elements.

In response to this "read," when the request is to be (or has already been) accepted, a "response" (0x72) is returned. When the request is not to be accepted, or when the stipulated DEOJ exists but the stipulated EPC

does not exist, "response not possible" (0x52) is returned. In the response frame format, the value of the object stipulated by the request is set in the SEOJ, the requested property is set in the EPC, and the value of the requested property (i.e., the read content) is set in the EDT. When "response not possible" is returned, nothing is written to the EDT. When the relevant object itself does not exist, neither "response" nor "response not possible" is returned. (See Fig. 4.8-2 for the exchange procedure.) Also, the "response" message DEA is defined as the requesting entity (i.e., the request message SEA).



When EDATA stipulates SEOJ during a "request," the EOJ stipulated by SEOJ in EDATA during the "request" is allocated as a DEOJ (b1 of the OHD is also set to 1), in the case of both "response not possible" and "response."

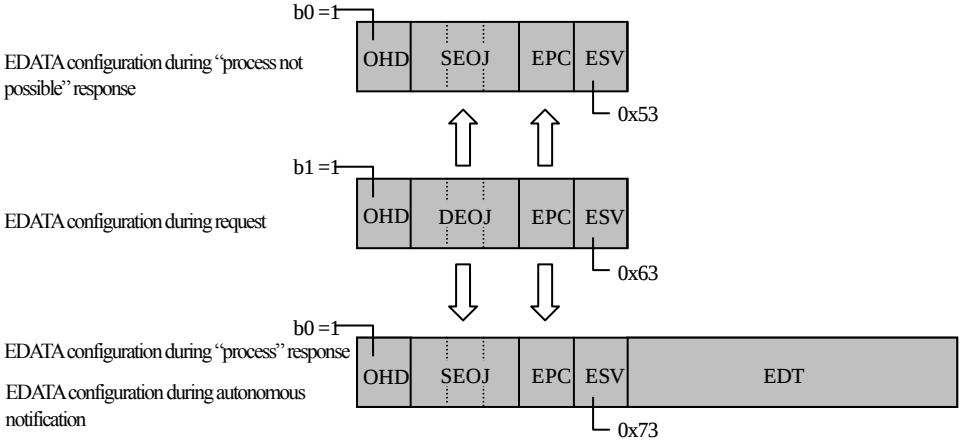
(3) Property value notification service [0x63, 0x73, 0x53]

There are two types of "notification": the notification sent as a response to a "notify request" (0x63) and the autonomous notification, which is unrelated to notify requests. The codes for the two types are identical. (Here, notification in response to a "notify request" signifies an announcement that does not specify the property value [content], while an autonomous notification is a voluntary announcement that was not made in response to a request.) In the case of a "notify request" (0x63), this indicates a request to notify (by general broadcast; hereafter "announce" will signify a general broadcast to the entire domain) the content of the property stipulated in the EPC of the DEOJ-stipulated object .

If the property specified by the EPC is an array property, the requested node reads the array property content, from the top position to the end, in units of the array element size and arrays them in the same order (starting with Element 0) to be used as the EDT for the property notification service. If the total data size of the array property elements exceeds the maximum EDT size, an EDT shall be generated after discarding the excess portion. The parts that correspond to areas with no array element, if any, shall not be left vacant but shall instead be used for the succeeding array elements.

In response to this "notify request," when the request is to be accepted, a "response" (0x73) value is notified; when the request is not to be accepted, a "response not possible" response (0x53) value is returned. In the response frame format, the value of the object stipulated by the request is set in the SEOJ, the requested property is set in the EPC, and the value of the requested property (i.e., the notification content) is

set in the EDT. Here, the DEA is set to general broadcast, but when "response not possible" is returned, nothing is written to the EDT, and the DEA sets the EA value of the requesting entity. When the relevant object itself does not exist, neither "response" nor "response not possible" is returned. (See Fig. 4.8-2 for the exchange procedure.) In the case of an autonomous "notification", the DEA is set to a general broadcast for a required status change notification. In the other cases, however, the DEA can be set as desired regardless of whether "broadcast" or "individual" is selected.



When EDATA stipulates SEOJ during a request, the EOJ stipulated by SEOJ in EDATA during the "request" is allocated as a DEOJ. In the case of both "response not possible" and "process," the EOJ stipulated in the SEOJ in the EDATA during the "request" is allocated as a DEOJ within the EDATA (b1 of the OHD is also set to 1). In the case of autonomous notification, the required notification of status change does not add a DEOJ; in all other cases, the addition of a DEOJ is optional.

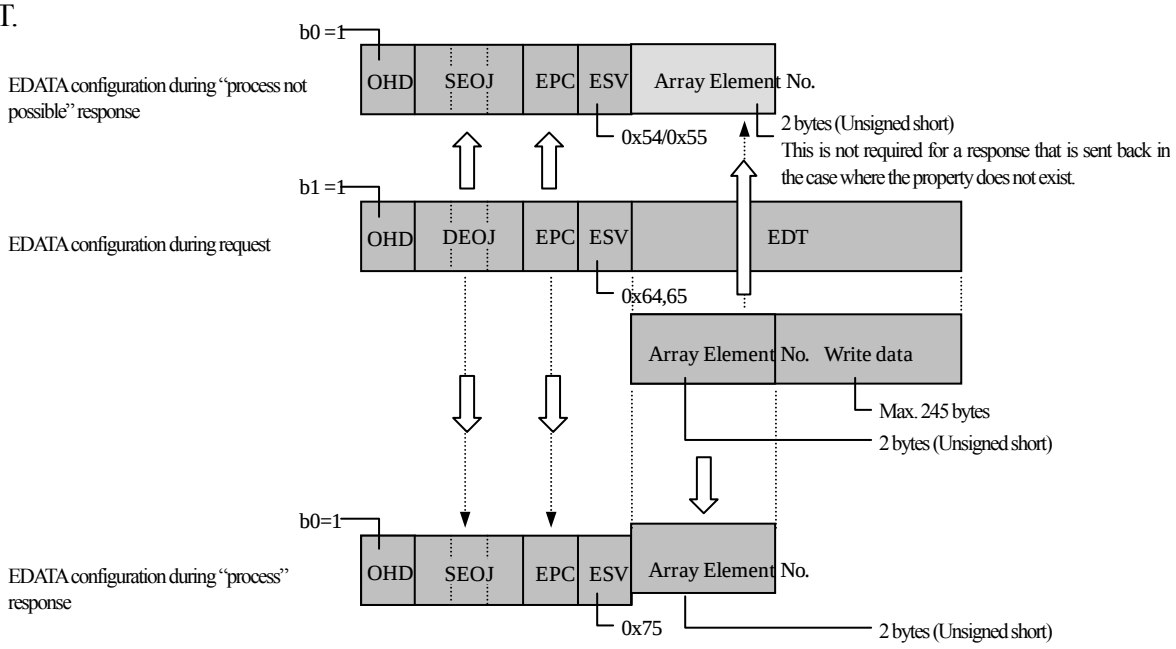
(4) Property value element-stipulated write service [0x64, 0x65, 0x75, 0x54, 0x55]

In the case of a "request" (0x64, 0x65), this indicates a request to write the value stipulated in the EDT (includes array element number and write request value data) of the property stipulated in the EPC of the DEOJ-stipulated object. In response to this "request," when a value to process the response is stipulated, and when the request is to be (or has already been) accepted, a "response" (0x75) is returned. However, this "response" is not a processing implementation response. When the request is not to be accepted, or when the stipulated DEOJ exists but the stipulated EPC does not exist, and when the stipulated DEOJ and EPC exist but the array element does not, "response not possible" (0x54, 0x55) is returned.

In the frame format for response, the value of the object stipulated by the request is set in the SEOJ, and the relevant property is set in the EPC. When the relevant object itself does not exist, neither "response" nor "response not possible" is returned. (See Fig. 4.9-2 for the exchange procedure.) Also, the "response" message DEA is defined as the requesting entity (i.e., the request message SEA).

The EDT of the "response not possible" message in the case where the request cannot be accepted or where the specified DEOJ and EPC exist but the specified array element does not exist shall be the array element number of the request. In the case where the specified DEOJ exists but the specified EPC does not exist, the "response not possible" message shall have no EDT or shall use the array element number as the

EDT.



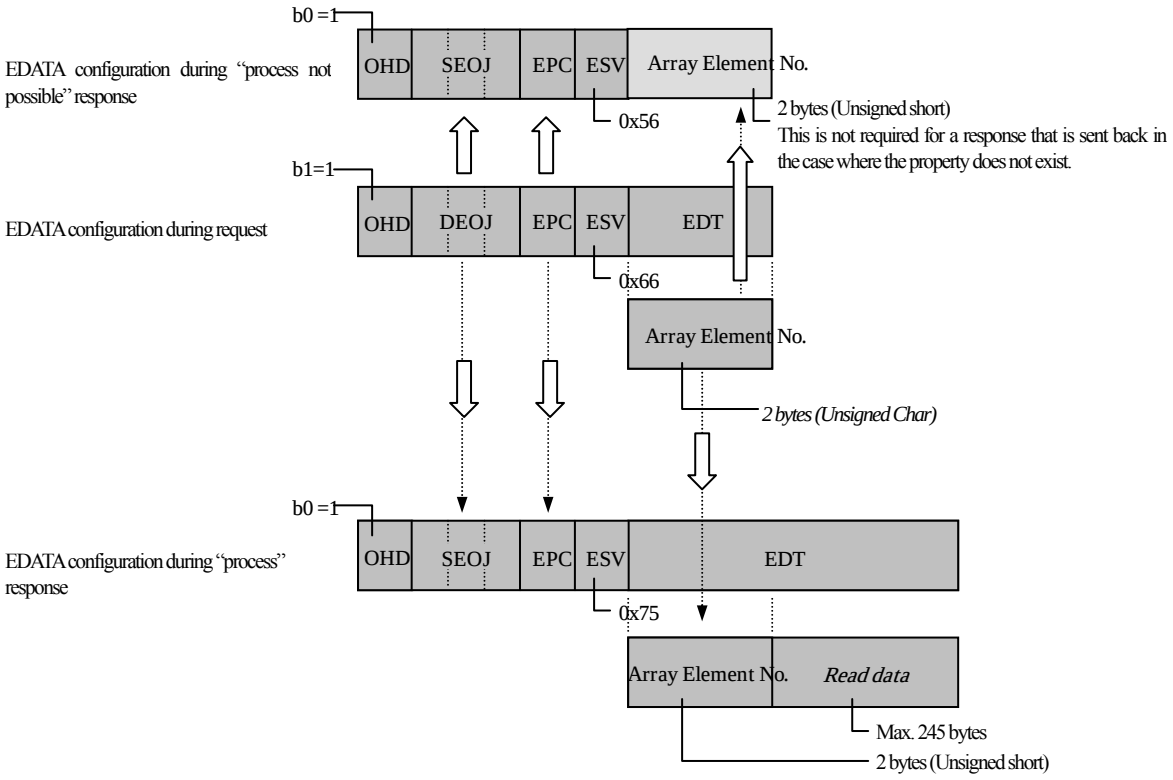
The content of each array element number in an array format property is defined separately for each property. When the stipulated (array) element does not exist, "response not possible" is returned. Also, when EDATA stipulates SEOJ during a "request," the EOJ stipulated in SEOJ by EDATA during the "request" is allocated as a DEOJ within EDATA (b1 of the OHD is also set to 1) in the case of both "response not possible" and "response."

(5) Property value element-stipulated read service [0x66, 0x76, 0x56]

In the case of a "read" (0x66), this indicates a request to read the content stipulated in the array element indicated in the EDT (includes array element number data to be read) of the property stipulated in the EPC of the DEOJ-stipulated object. In response to this "read," when the request is to be (or has already been) accepted, a "response" (0x76) is returned. When the request is not to be accepted, or when the stipulated DEOJ exists but the stipulated EPC does not, and when the stipulated DEOJ and EPC exist but the array element does not, "response not possible" (0x56) is returned.

The SEOJ, EPC and EDT of the response message shall be set to the value of the specified object, the requested property and the value of the requested property (the content of the data that has been read), respectively.

The EDT of the "response not possible" message in the case where the request cannot be accepted or where the specified DEOJ and EPC exist but the specified array element does not exist shall be the array element number of the request. In the case where the specified DEOJ exists but the specified EPC does not exist, the "response not possible" message shall have no EDT or shall use the array element number as the EDT. In the case where the target object itself does not exist, neither "response" message or "response not possible" message shall be returned (For the message exchange procedure, refer to Fig. 4.9-2). The DEA of a "response" message shall be the sender of the request (the SEA of the request message).



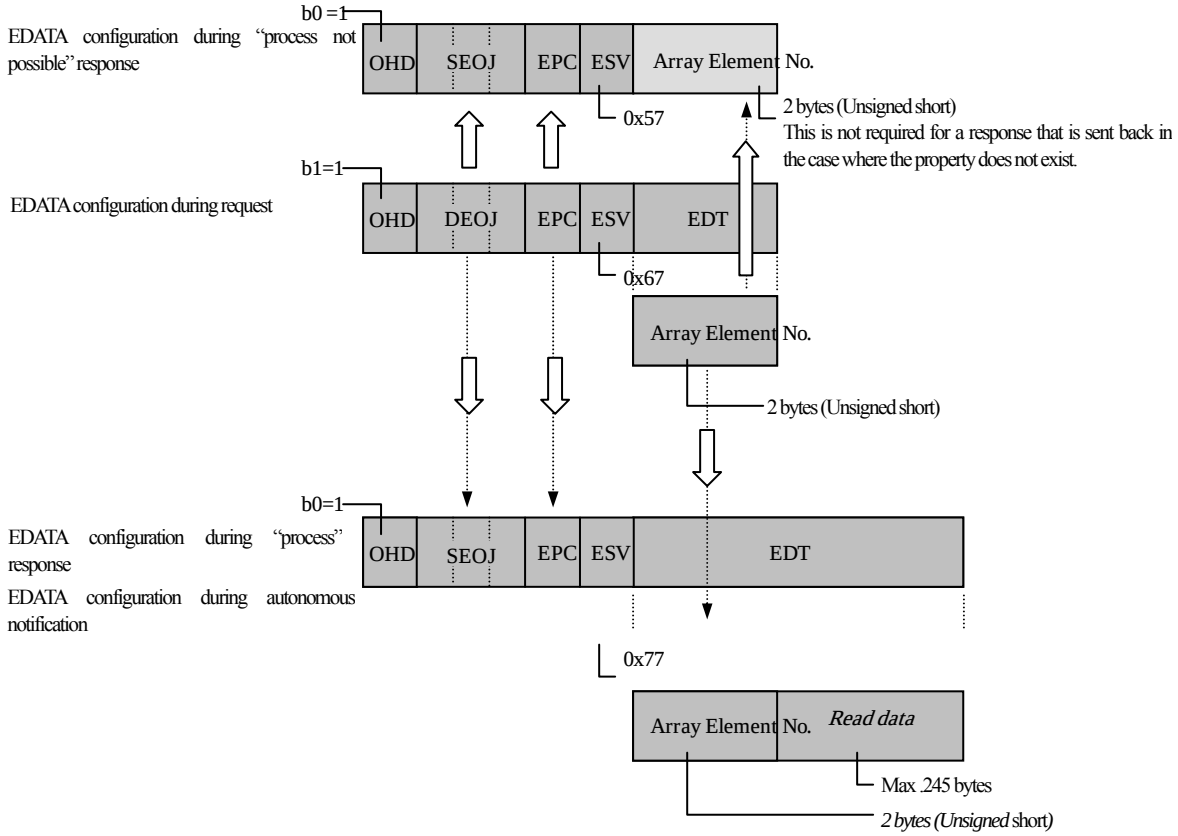
The content of each array element number in an array format property is defined separately for each property. When the stipulated array element (element) does not exist, "response not possible" is returned. If the EDATA of the request contains SEOJ designation, the EOJ specified by the SEOJ in the EDATA of the request shall be used as the DEOJ of the EDATA of the "response" or "response not possible" message (b1 of the OHD shall be set to 1).

(6) Property value element-stipulated notification service [0x67, 0x77, 0x57]

There are two types of "notification": notification sent in response to a "notify request" (0x67) and autonomous notification, which is unrelated to notify requests. The codes for the two types are identical. (Here, notification in response to a "notify request" signifies an announcement that does not specify the property value [content], while an autonomous notification is a voluntary announcement that was not made in response to a request.) In the case of a "notify request" (0x67), this indicates a request to notify (announce) the content of the array element number stipulated in the EDT of the property stipulated in the EPC of the DEOJ-stipulated object. In response to this "notify request," when the request is to be accepted, an array element value (content) is announced as a "response" (0x77). When the request is not to be accepted, or when the stipulated DEOJ exists but the stipulated EPC does not, and when the stipulated DEOJ and EPC exist but the array element does not, "response not possible" (0x57) is returned. In the frame format for response, the value of the object stipulated by the request is set in the SEOJ, the requested property is set in the EPC, and the value of the requested array element number and its array element value (i.e., the notification content) is set in the EDT. Here, the DEA is set to general broadcast, but when

"response not possible" is returned, the DEA sets the EA value of the requesting entity. When the relevant object itself does not exist, neither "response" nor "response not possible" is returned. (See Fig. 4.9-2 for the exchange procedure.)

The EDT of the "response not possible" message in the case where the request cannot be accepted or where the specified DEOJ and EPC exist but the specified array element does not exist shall be the array element number of the request. In the case where the specified DEOJ exists but the specified EPC does not exist, the "response not possible" message shall have no EDT or shall use the array element number as the EDT.



The content of each array element number is defined separately for each property. When the stipulated (array) element does not exist, "response not possible" is returned. Also, when EDATA stipulates SEOJ during a "request," the EOJ stipulated in the SEOJ by EDATA during the request is allocated as a DEOJ within the EDATA (b1 of the OHD is also set to 1) in the case of both "response not possible" and "response." In the case of autonomous notification, the required notification of status change does not add a DEOJ; in all other cases, the addition of a DEOJ is optional.

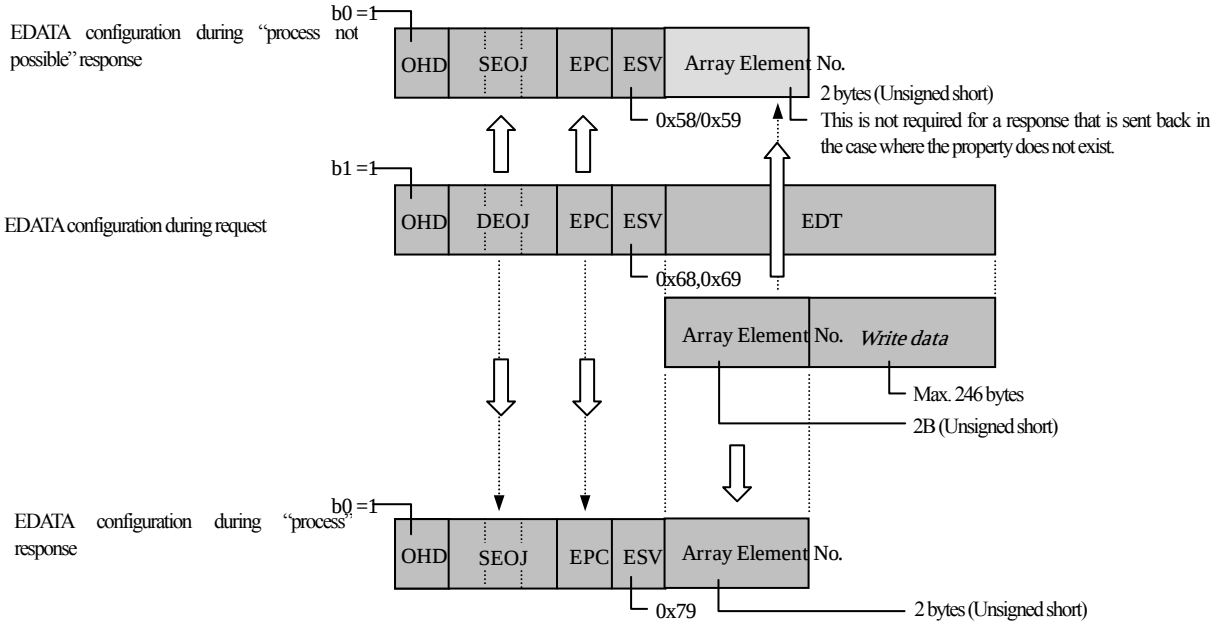
(7) Property value element-stipulated addition [0x68, 0x69, 0x58, 0x59, 0x79]

In the case of a "request" (0x68, 0x69), this indicates a request to add the array element indicated in the EDT (includes array element number and write request value) of the property stipulated in the EPC of the DEOJ-stipulated object, and to write the value stipulated therein. In response to this "request," when a value indicating implementation of the response (0x68) is stipulated, and when the request is to be (or has already been) accepted, "response" (0x78) is returned. However, this "response" is not a processing

implementation response. When the request is not to be accepted, or when the stipulated DEOJ exists but the stipulated EPC does not, and when the stipulated DEOJ and EPC exist but the array element does not, "response not possible" (0x58, 0x59) is returned.

In the frame format for response, the value of the object stipulated by the request is set in the SEOJ, and the requested property is set in the EPC. When the relevant object itself does not exist, neither "response" nor "response not possible" is returned. (See Fig. 4.9-2 for the exchange procedure.) Also, the "response" message DEA is defined as the requesting entity (i.e., the request message SEA).

The EDT of the "response not possible" message in the case where the request cannot be accepted or where the specified DEOJ and EPC exist and the specified array element exists shall be the array element number of the request. In the case where the specified DEOJ exists but the specified EPC does not exist, the "response not possible" message shall have no EDT or shall use the array element number as the EDT.



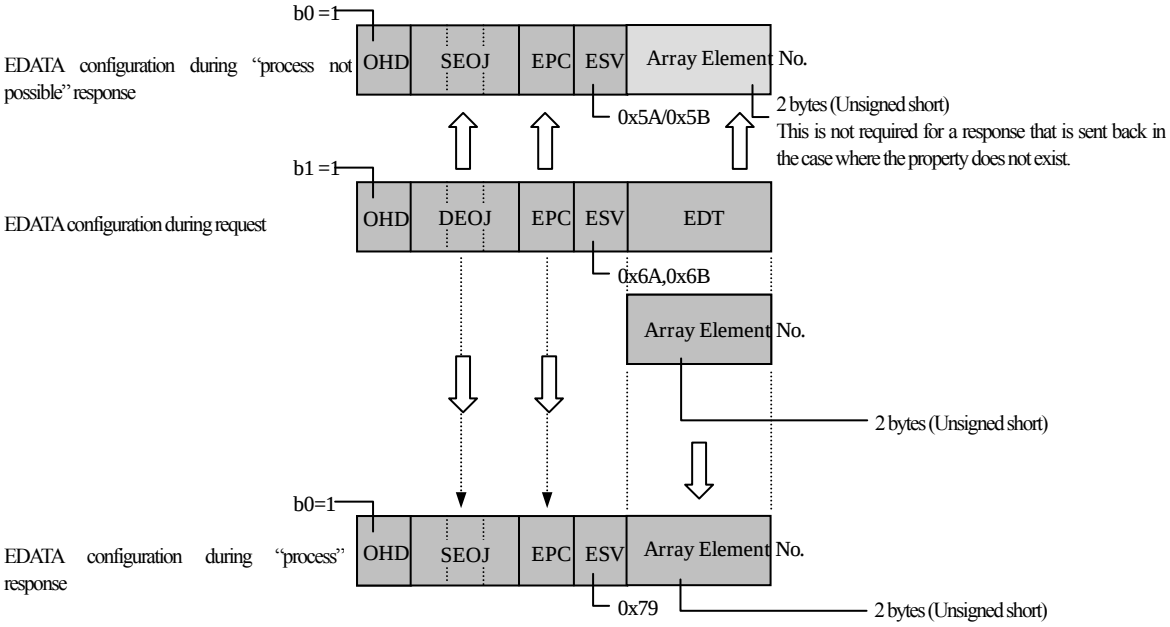
The content of each array element number in an array format property is defined separately for each property. When the stipulated array (element) does not exist, "response not possible" is returned. Also, when EDATA stipulates SEOJ during a "request," the EOJ stipulated in the SEOJ by EDATA during the request is allocated as a DEOJ within the EDATA (b1 of the OHD is also set to 1) in the case of both "response not possible" and "response."

(8) Property value element-stipulated deletion [0x6A, 0x6B, 0x5A, 0x5B, 0x7B]

In the case of a "request" (0x6A, 0x6B), this indicates a request to delete the array element indicated in the EDT (array element number) from the property stipulated in the EPC of the DEOJ-stipulated object. In response to this "request," when a value indicating implementation of the response (0x6B) is stipulated, and when the request is to be (or has already been) accepted, a "response" (0x7B) is returned. However, this "response" is not a processing implementation response. When the request is not to be accepted (including cases in which the deletion is not to be implemented), or when the stipulated DEOJ exists but the stipulated

EPC does not, "response not possible" (0x5A, 0x5B) is returned. In the frame format for response, the value of the object stipulated by the request is set in the SEOJ, and the relevant property is set in the EPC. When the relevant object itself does not exist, neither "response" nor "response not possible" is returned. (See Fig. 4.9-2 for the exchange procedure.) Also, the "response" message DEA is defined as the requesting entity (i.e., the request message SEA).

When the request is not to be accepted, or when the stipulated DEOJ and EPC exist but the array element does not, the "response not possible" EDT is the array element number of the "request." When the stipulated DEOJ exists but the stipulated EPC does not, the "response not possible" is without EDT.



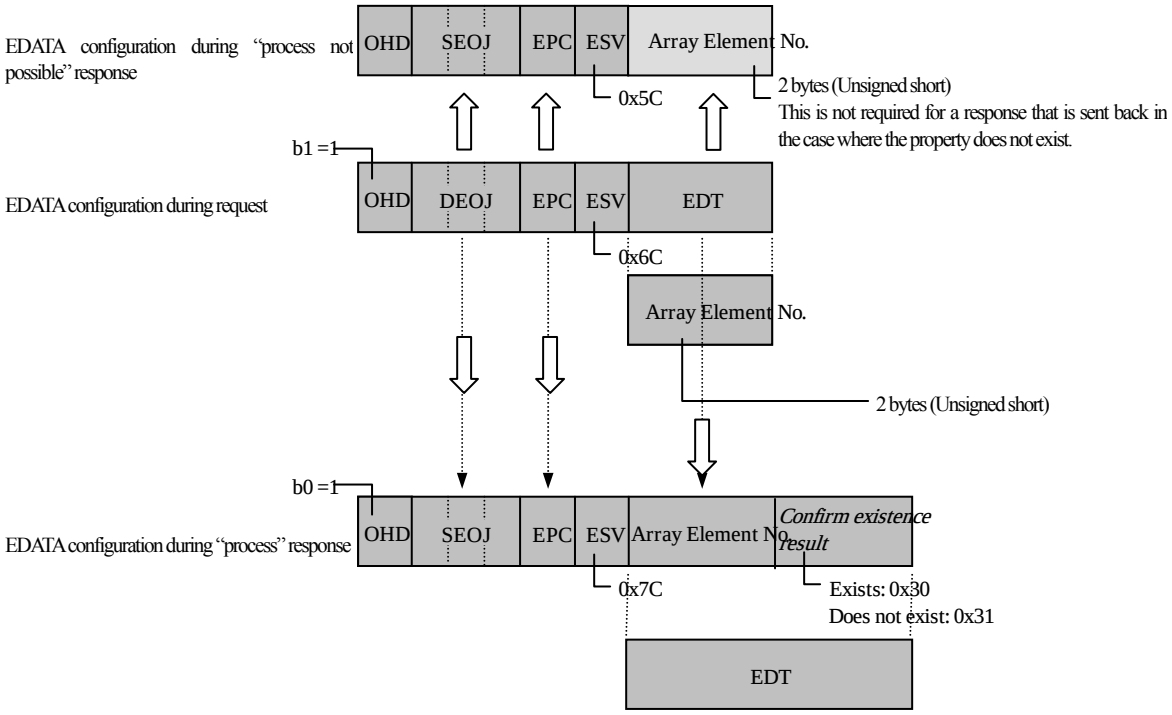
The content of each array element number in an array format property is defined separately for each property. When the stipulated array element (element) does not exist, "response not possible" is returned. Also, when EDATA stipulates SEOJ during a "request," the EOJ stipulated in the SEOJ by EDATA during the request is allocated as a DEOJ within the EDATA (b1 of the OHD is also set to 1) in the case of both "response not possible" and "response."

(9) Property value element-stipulated existence confirmation [0x6C, 0x5C, 0x7C]

In the case of a "request" (0x6C), this indicates a request to confirm the existence of the array element indicated in the EDT (includes array element number value information) in the property stipulated in the EPC of the DEOJ-stipulated object. When the request is to be (or has already been) accepted, a "response" (0x7C) is returned. When the request is to be rejected (cannot be processed by the ESV) or when the specified DEOJ exists but the specified EPC does not exist, "process not possible" (0x5C) is returned. In the frame format for response, the value of the object stipulated by the request is set in the SEOJ, and the relevant property is set in the EPC. When the relevant object itself does not exist, neither "response" nor "response not possible" is returned. (See Fig. 4.9-2 for the exchange procedure.) Also, the "response" message DEA is defined as the requesting entity (i.e., the request message SEA).

The EDT of the "response not possible" message in the case where the request cannot be accepted or

where the specified DEOJ and EPC exist but the specified array element cannot be added shall be the array element number of the request. In the case where the specified DEOJ exists but the specified EPC does not exist, the "response not possible" message shall have no EDT or shall use the array element number as the EDT.



The content of each array element number in an array format property is defined separately for each property. Also, when EDATA stipulates SEOJ during a "request," the EOJ stipulated in the SEOJ by EDATA during the "request" is allocated as a DEOJ within the EDATA (b1 of the OHD is also set to 1) in the case of both "response not possible" and "response."

(10) Property value element addition [0x6D, 0x6E, 0x5D, 0x5E, 0x7E]

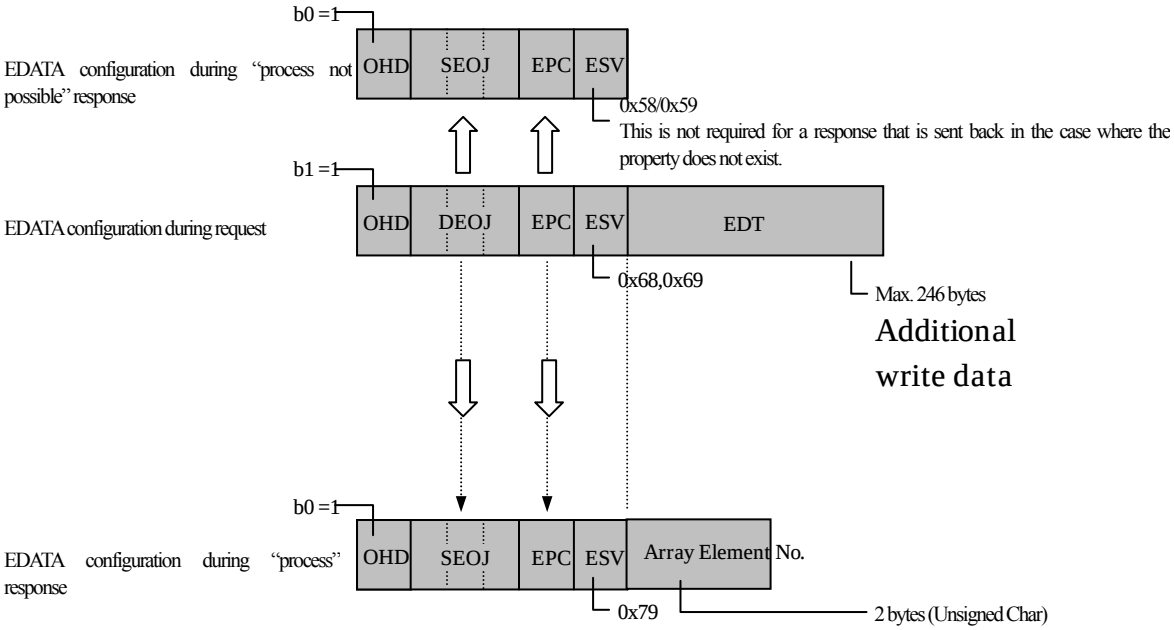
In the case of a "request" (0x6D, 0x6E), this indicates a request to newly add an array element to the property stipulated in the EPC of the DEOJ-stipulated object, and to write to the newly added array element the value data stipulated in the EDT. In response to this "request," when a value indicating implementation of the response (0x6E) is stipulated, and when the request is to be (or has already been) accepted, a "response" (0x7F) is returned. However, this "response" is a processing implementation response, and the added array element number is returned as an EDT. When the request is not to be accepted, or when the stipulated DEOJ exists but the stipulated EPC does not, "response not possible" (0x5D, 0x5E) is returned.

In the frame format for response, the value of the object stipulated by the request is set in the SEOJ, and the relevant property is set in the EPC. When the relevant object itself does not exist, neither "response" nor "response not possible" is returned. (See Fig. 4.9-2 for the exchange procedure.) Also, the "response" message DEA is defined as the requesting entity (i.e., the request message SEA).

For "response not possible", EDT does not exist.

The EDT of the "response not possible" message in the case where the request cannot be accepted or

where the specified DEOJ and EPC exist but the specified array element cannot be added shall be the array element number of the request. In the case where the specified DEOJ exists but the specified EPC does not exist, the "response not possible" message shall have no EDT or shall use the array element number as the EDT.



The content of each array element number in an array format property is defined separately for each property. Also, when EDATA stipulates SEOJ during a "request," the EOJ stipulated in the SEOJ by EDATA during the "request" is allocated as a DEOJ within the EDATA (b1 of the OHD is also set to 1) in the case of both "response not possible" and "response."

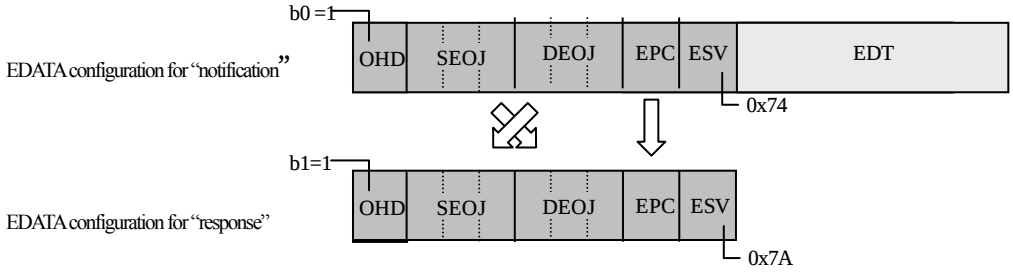
(11) Property value notification (response required) [0x74, 0x7A]

The "notification (response required)" (0x74) autonomously notifies a specific node of the property value stipulated by the EPC of the SEOJ-stipulated object and requests a response. The response process for this "notification (response required)" varies depending on whether or not the DEOJ is specified.

When the DEOJ is not specified, a "response" (0x7A) for autonomous notification reception is returned at all times.

When the DEOJ is specified, the subsequent process varies depending on whether or not the specified DEOJ exists. If the specified DEOJ exists, a "response" (0x7A) for autonomous notification reception is returned. If the specified DEOJ does not exist, the message is discarded.

If a node receives a "notification (response required)" for which a broadcast is specified, the node discards the message.



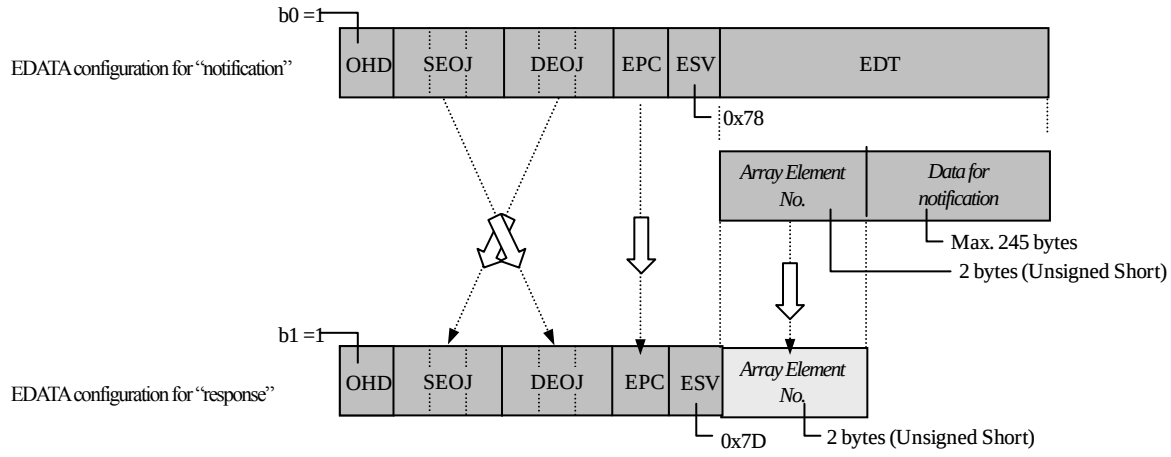
(12) Property value element-stipulated notification (response required) [0x78, 0x7D]

The "notification (response required)" (0x78) autonomously notifies a specific node of the array element value stipulated by the EDT (array element number) of the property stipulated by the EPC of the SEOJ-stipulated object, and requests an acknowledgment. The response message format and response process for this "notification (response required)" varies depending on whether or not the DEOJ is specified.

When the DEOJ is not specified, a "response" (0x7D) for notification reception is returned at all times.

When the DEOJ is specified, the subsequent process varies depending on whether or not the specified DEOJ exists. If the specified DEOJ exists, a "response" (0x7D) for notification reception is returned. If the specified DEOJ does not exist, the message is discarded.

If a node receives a "notification (response required)" for which a broadcast is specified, the node discards the message.



The services shown in Tables 4.10-1 through 4.10-3 above are specified for each property. Regarding those stipulated as services that must be incorporated in each property, if they have the functions of that property and disclose via communications (read/write/notification, etc.), this indicates that they must be processed. Processing of services for each property is specified in Part II, Chapter 9 and in the ECHONET Objects Detailed Specifications Appendix of Part II in the Access Rules column of the object class detailed specification tables. Access rules indicate all services that can be implemented. In this Specification, the following nine access rules are specified:

SetProcesses services related to write requests for non-array property values (Performs processing indicated in (1))
Get Processes services related to read requests for non-array property values (Performs processing indicated in (2) (3) and (11))
SetM Processes services related to write requests for array property values (Performs processing indicated in (4))
GetM Processes services related to read requests for array property values (Performs processing indicated in (5) (6) and (12))
AddM Processes services related to element-stipulated add requests for array property values ((Performs processing indicated in (7))
DellM Processes services related to delete requests for array property values (Performs processing indicated in (8))
CheckM Processes services related to existence confirmation requests for array property value elements (Performs processing indicated in (9))
AddMS Processes services related to non-array-element-stipulated add requests for array property values (Performs processing indicated in (10))
Anno Processes non-array property value notification services (Performs processing indicated in (3) and (11))
AnnoM Processes array property value notification services (Performs processing indicated in (6) and (12))

The above processing is specified for each property; there are no requirements for mixing Set and SetM or Get and GetM.

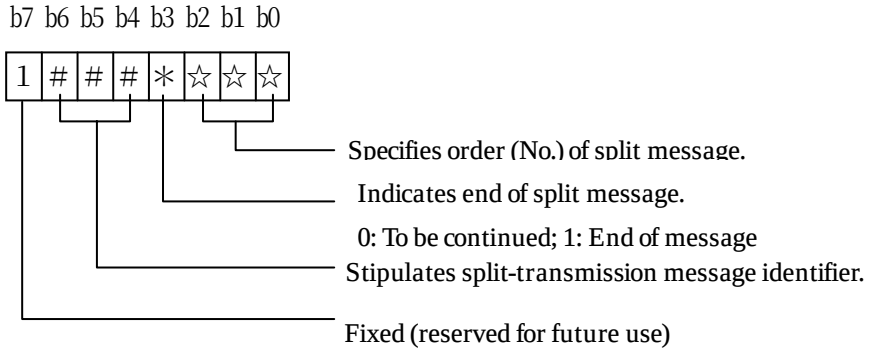
4.2.9 ECHONET Property Value Data (EDT)

This section presents detailed code specifications for the ECHONET property value data (EDT) range shown in Figs. 4.1-1 and 4.1-2. EDT consists of data for the relevant ECHONET property (EPC), such as status notification or specific setting and control by an ECHONET service (ESV). Detailed specifications are provided for the size, code value, etc. of the EDT for each EPC (see Chapter 9.)

4.2.10 ECHONET Data Counter (EDC)

This section provides detailed specifications for the ECHONET data counter (EDC) code that is a component of ECHONET split frames shown in Fig. 4.1-3.

Messages may be split into a maximum of eight components, and b0–b2 shows the order of the split messages (starts at b0 = b1 = b2 = 0; ends at a maximum of b0 = b1 = b2 = 1). The split message identifier requirement bit (b4, b5, b6) is also specified for cases in which a message from the ECHONET Communications Processing Block is sent repeatedly to the same node and in which all of the repeated messages require splitting. However, the method for setting this value will not be specified here. Therefore, in the receiving-side Protocol Difference Absorption Processing Block, messages with the same MAC address for the source and the same values for b4–b6 are assembled based on the data contained in the b0–b2 split counter.



Note: The meanings of the bits when b7 = 0 are to be stipulated in the future (reserved for future use).

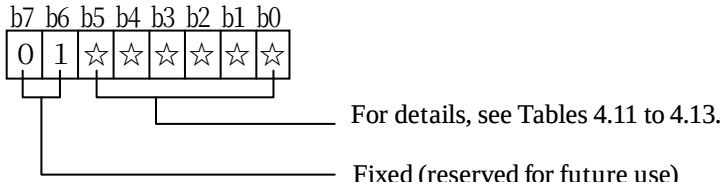
Fig. 4.10 EDC Detailed Specifications

When a message is not split, values shall be set as follows: b2:b1:b0 = 0:0:0 and b3 = 1.

When the intended destination is the same, it is recommended that split messages having the same identifier be sent without any intervening messages.

4.2.11 Compound ECHONET Service (CpESV)

This section provides detailed specifications for the compound ECHONET service (CpESV) code shown in Figs. 4.1-1 and 4.1-2.



Note: When bits b7 and b6 are 0 and 1, respectively, the meanings of bits b0 to b5 are stipulated separately.

Fig. 4.11 EpESV Configuration

The service provided by this code is used when the compound message format is used. It specifies a simultaneous action for two or more properties stipulated by the EPC. However, it does not stipulate the order of operations. The order of property operations is an implementation issue.

Three types of operations are provided: request, response, and notification. The response is subdivided into two types: "accepted" response and "process not possible" request. The "accepted" response is used when the service request in relation to all the EPC-stipulated properties is accepted. The "process not possible" request is used when one or more specified properties do not exist or when the specified service cannot be processed for one or more properties.

- Request
- Response ("accepted" response/"process not possible" response)
- Notification

The "response" is a response to a "request" that requires a response. It must be returned when a DEOJ-stipulated object exists. When the service processing request related to all the EPC-stipulated properties is accepted, the "accepted" response must be returned. If the processing request related to one or more specified properties cannot be accepted or if the object exists but one or more properties do not exist, "process not possible" must be returned. When the "request" does not require any response or when the specified object does not exist, no "response" will be returned.

Furthermore, "write" (response-required write/no-response-required write), "read", and "notification" (autonomous notification/response-required notification) are regarded as specific operations. Therefore, the following five types are set. Regarding the OpESV for compound messages, array element properties are not targeted.

1) Property value write request (no response required)
2) Property value write request (response required)

3) Property value read request
4) Property value notification
5) Property value notification (response required)

The CpESV and message configuration (presence of SEOJ and DEOJ) and their relationship to EPC and ESV are described below.

[1] The EPC of an ECHONET message in which only the SEOJ is specified indicates the property of the SEOJ-stipulated source object. In this case, the "response", "notification", or autonomous "notification" concerning the "request" related to two or more SEOJ/EPC-stipulated properties is positioned in the CpESV. When the CpESV is a "request" while this configuration is employed, the associated message must be handled as an erroneous message.

[2] The EPC of an ECHONET message in which only the DEOJ is specified indicates the property of the DEOJ-stipulated destination object. In this case, the "request" related to two or more DEOJ/EPC-stipulated properties is positioned in the CpESV. When the CpESV is a "response" or "notification" while this configuration is employed, the associated message must be handled as an erroneous message.

[3] The EPC of an ECHONET message in which the SEOJ and DEOJ are both specified is such that the CpESV value determines whether the target object is stipulated by the SEOJ or DEOJ. When the CpESV is a "response" or "notification", it is concluded that the EPC forms a SEOJ-stipulated object and that the "response" or "notification" is addressed to a DEOJ-stipulated object. On the other hand, when the CpESV is a "request", it is concluded that the EPC forms a DEOJ and that the "request" is issued from an SEOJ-stipulated object.

Tables 4.11 through 4.13 show specific CpESV code assignments. The details of ① through ⑤ above are given in (1) through (5) (the related numbers are indicated in the Remarks column of the tables). The figures in (1) through (5) presume that the DEOJ for a "request" is an individually specified code. However, when the DEOJ indicates an instance general broadcast, a response is transmitted with both "process not possible" and "response" configured for each target instance. Figure 4.12 shows a sequence diagram that indicates the relationship between individual CpESVs. The codes marked "reserved for future use" in the tables are to be stipulated in the future and must not be used with Version 2.11.

Table 4.11 List of CpESV Codes for Request/Notification

Service Code	ECHONET Service Content	Symbol	Remarks
0x60	Property value write request (no response required)	CpSetI	(1)
0x61	Property value write request (response required)	CpSetC	(2)
0x62	Property value read request	CpGet	(3)
0x63–0x6F	Reserved for future use		

Table 4.12 List of CpESV Codes for "Accepted" Response

Service Code	ECHONET Service Content	Symbol	Remarks
0x71	Property value write "accepted" response	CpSet_Res	CpESV = 61 response (2)
0x72	Property value read "accepted" response	CpGet_Res	CpESV = 62 response (3)
0x73	Property value notification	CpINF_Res	(4)
0x74	Property value notification (response required)	CpINFC	(5)
0x7A	Property value notification response	CpINFC_Res	CpESV = 74 response (5)
0x75–0x79, 0x7B–0x7F	Reserved for future use		

Table 4.13 List of CpESV Codes for "Process Not Possible" Response

Service Code	ECHONET Service Content	Symbol	Remarks
0x50	Property value write "process not possible" response (1)	CpSetI_SNA	CpESV = 60 "process not possible" response (1)
0x51	Property value write "process not possible" response (2)	CpsSetC_SNA	CpESV = 61 "process not possible" response (2)
0x52	Property value read "process not possible" response	CpGet_SNA	CpESV = 62 "process not possible" response (3)
0x5F	Message length excessive	CpOverFlow	Response to be returned when the response message is too long
0x53–0x5E	Reserved for future use		

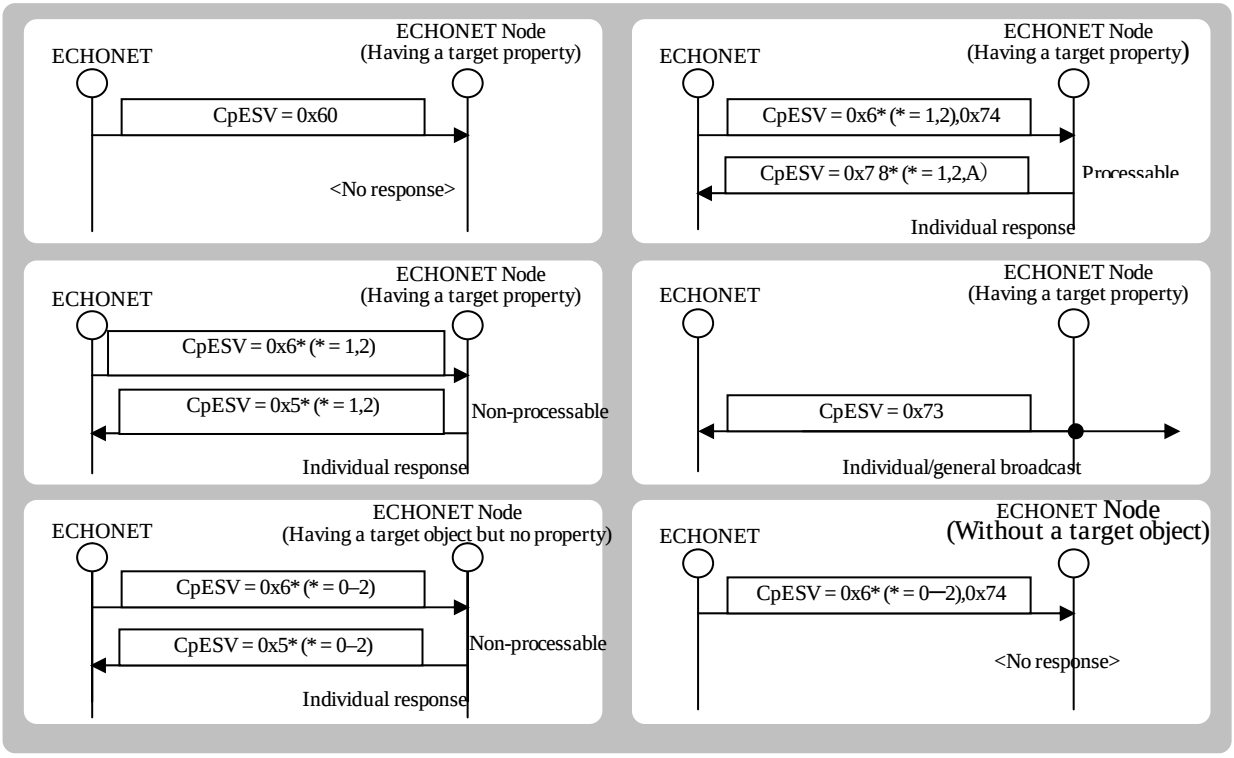


Fig. 4.12 Basic Sequence

(1) Property value write request (requiring no response) service [0x60, 0x50]

The write request requiring no response (CpESV = 0x60) requests that the EDT-stipulated contents be written into the EPC-stipulated properties of the DEOJ-stipulated object. The order of write operations is not stipulated. The response from a request-processing node is as indicated below:

(a) When a processing request for all properties is accepted

No response will be made.

(b) When one or more properties relevant to the request do not exist, a processing request to one or more properties cannot be accepted, or an array property is targeted

A write "process not possible" response (1) (CpESV = 0x50) will be returned.

(c) When the object relevant to the request does not exist

No response will be made.

(d) When two or more identical properties exist in the request message

Individual processes will be performed on the presumption that differing requests are issued.

A response will be made in accordance with the processing results.

Note: The order of processes depends on the implementation. Therefore, the resulting final property status and value also depend on the implementation.

The message structure of a write "process not possible" response to a property value write request (requiring no response) is such that the object code of the request destination becomes the SEOJ and the object code of the request source becomes the DEOJ. The OPC takes the same value as in the request message.

For requests (1 to n) that relate to nonexistent properties and process requests that are rejected, both the PDC and EDT use the same values as those used in the write request. For requests related to properties for which processing requests are accepted, the PDC value is 0x01 and the EDT value is omitted. As for the EPC, the EPC in the request message is used as is. If the target object does not exist, neither the "response" nor the "process not possible" response is returned.

An appropriate value for the OHD must be specified in accordance with the SEOJ/DEOJ configuration in the message. Figure 4.13 shows the relationship between a write request requiring no response and write addition response for situations where Request m cannot be accepted. The EPC sequence in the request message must be equal to the EPC sequence in the write "process not possible" response message.

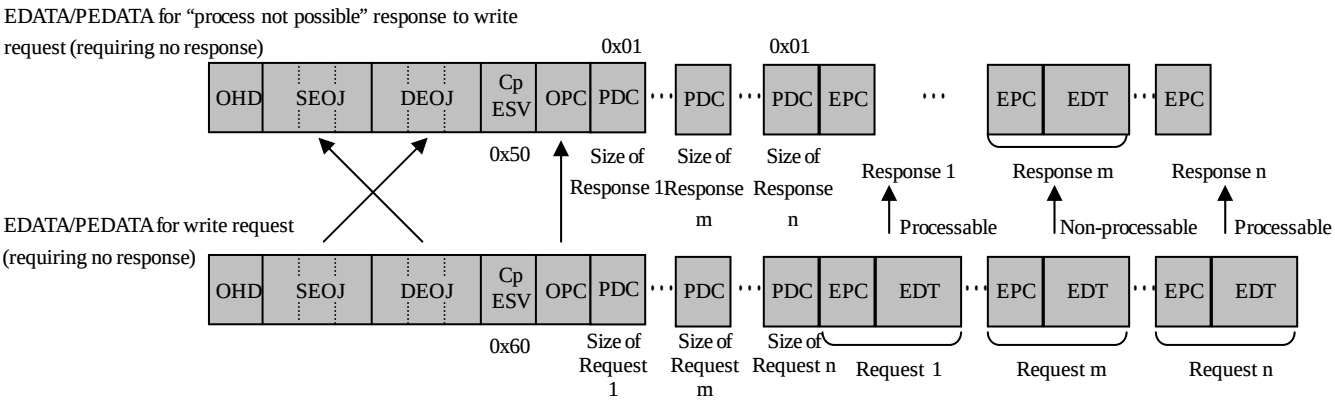


Fig. 4.13 Relationship Between Write Request (Requiring No Response) and Write "Process Not Possible" Response

(2) Property value write request (requiring a response) service [0x61, 0x71, 0x51]

The write request requiring a response (CpESV = 0x61) requests that the EDT-stipulated contents be written into the EPC-stipulated properties of the DEOJ-stipulated object. The order of write operations is not stipulated. The response from a request-processing node is as indicated below:

(a) When a processing request for all properties is accepted
A write "accepted" response (CpESV = 0x71) will be returned.

(b) When one or more properties relevant to the request do not exist, a processing request to one or more properties cannot be accepted, or an array property is targeted
A write "process not possible" response (CpESV = 0x51) will be returned.

(c) When the object relevant to the request does not exist
No response will be made.

(d) When two or more identical properties exist in the request message
Individual processes will be performed on the presumption that differing requests are issued. A response will be made in accordance with the processing results.
Note: The order of processes depends on the implementation. Therefore, the resulting final property status and value also depend on the implementation.

The message structure of a write "process not possible" response to a property value write request (requiring a response) is such that the object code of the request destination becomes the SEOJ and the object code of the request source becomes the DEOJ. The OPC takes the same value as in the request message.

For requests (1 to n) that relate to nonexistent properties and process requests that are rejected, both the PDC and EDT use the same values as those used in the write request. For requests related to properties for which processing requests are accepted, the PDC value is 0x01 and the EDT value is omitted. As for the EPC, the EPC in the request message is used as is. If the target object does not exist, neither the "response" nor the "process not possible" response is returned.

The message structure of a write "accepted" response is such that the object code of the request destination becomes the SEOJ and the object code of the request source becomes the DEOJ. The OPC and subsequent values are omitted.

An appropriate value for the OHD must be specified in accordance with the SEOJ/DEOJ configuration in the message. Figure 4.14 shows the relationships among a write request requiring a response, a write "accepted" response, and a write "process not possible" response for situations where Request m cannot be accepted. The EPC sequence in the request message must be equal to the EPC sequence in the write "process not possible" response message.

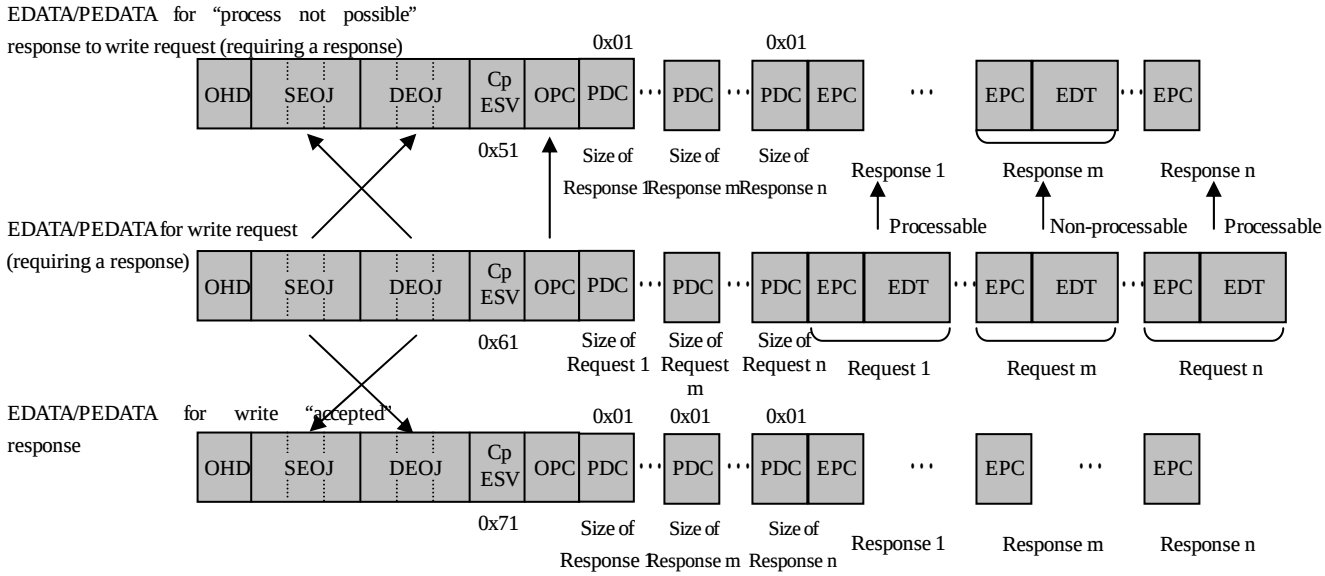


Fig. 4.14 Relationship Among Write Request (Requiring a Response), Write "Accepted" Response, and Write "Process Not Possible" Response

(3) Property value read request service [0x62, 0x72, 0x52, 0x5F]

The property value read request (CpESV = 0x62) requests that the contents of EPC-stipulated properties of the DEOJ-stipulated object be read. The order of read operations is not stipulated. The response from a request-processing node is as indicated below:

(a) When a processing request for all properties is accepted

A read "accepted" response (CpESV = 0x72) shall be used to return all the read values.

(b) When one or more properties relevant to the request do not exist, a processing request to one or more properties cannot be accepted, or an array property is targeted

A write "process not possible" response (CpESV = 0x52) shall be used to return the values of the read properties.

(c) When the object relevant to the request does not exist

No response will be made.

(d) When two or more identical properties exist in the request message

Individual processes will be performed on the presumption that differing requests are issued. A response will be made in accordance with the processing results.

Note: The order of processes depends on the implementation. Therefore, if two or more property states are read, the resulting final status depends on the implementation.

The message structure of a read "process not possible" response is such that the object code of the request destination becomes the SEOJ and the object code of the request source becomes the DEOJ. The OPC takes the same value as in the request message.

For requests (1 to n) that relate to nonexistent properties and process requests that are rejected, the PDC value is 0x01 and the EDT value is omitted. For requests related to properties for which processing requests are accepted, the read value is set in the EDT and the total number of EPC and EDT bytes is regarded as the PDC. If the target object does not exist, neither the "response" nor the "process not possible" response is returned.

The message structure of a read "accepted" response is such that the object code of the request destination becomes the SEOJ and the object code of the request source becomes the DEOJ. The read value is set in the EDT, and the total number of EPC and EDT bytes is regarded as the PDC.

An appropriate value for the OHD must be specified in accordance with the SEOJ/DEOJ configuration in the message. Figure 4.15 shows the relationships among a read request, a read "accepted" response, and a read "process not possible" response for situations where Request m cannot be accepted. The EPC sequence in the request message must be equal to the EPC sequence in the read "accepted" response and read "process not possible" response messages.

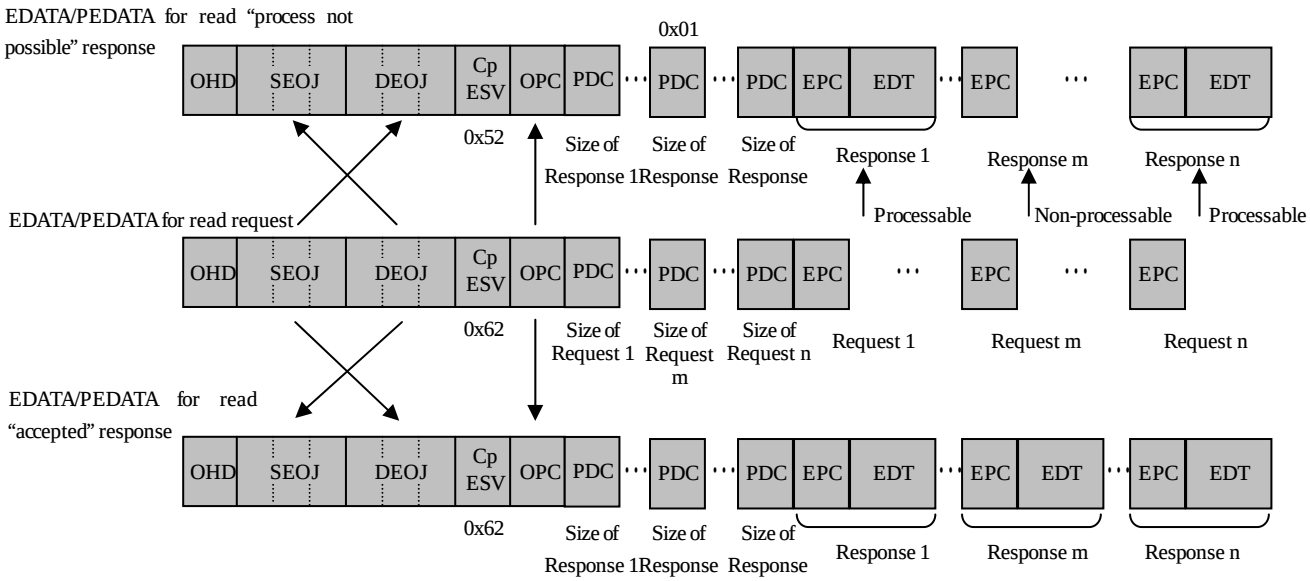


Fig. 4.15 Relationship Among Read Request (Requiring a Response), Read "Accepted" Response, and Read "Process Not Possible" Response

As is obvious from Fig. 4.15, the read "accepted" response message is longer than the read response message. Therefore, the maximum permissible message length may be exceeded when an attempt is made to return all the property values that are read in compliance with the request. In such a situation, a response will be made using the message length overflow service code (CpESV = 0x5F). In this case, the responding side can determine the number of property values to be returned; however, the sequence of such properties must be the same as in the request message.

(4) Property value notification service [0x73]

The property value notification (CpESV = 0x73) reads the contents of EPC-stipulated properties and reports them to the DEOJ-stipulated object. When the DEOJ is not contained in the message, it is a notification to nodes. Either "individual" or "broadcast" can be selected for addressing purposes. The order of property value notifications is not stipulated. Nodes receiving this message will not return a response.

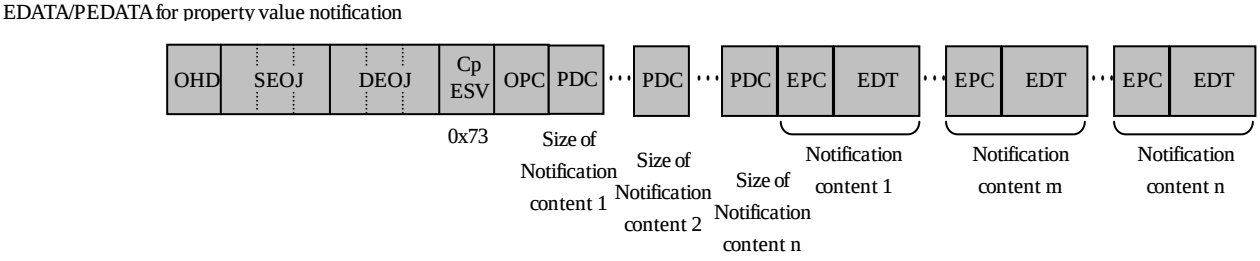


Fig. 4.16 Relationship Between Notification Request and Notification Response

(5) Property value notification (requiring a response) service [0x74, 0x7A]

The property value notification requiring a response (CpESV = 0x74) reads the contents of EPC-stipulated properties and reports them to the DEOJ-stipulated object. When the DEOJ is not contained in the message, it is a notification to a node. Only "individual" is available for addressing purposes. The order of property value notifications is not stipulated. The response from a node receiving this message is as indicated below:

(a) When a notification is accepted
A property value notification response (CpESV = 0x7A) will be returned.

(b) When the DEOJ-stipulated object does not exist
No response will be made.

The message structure of the notification response is such that the object code of the request destination becomes the SEOJ and the object code of the request source becomes the DEOJ. The OPC takes the same value as in the request message.
An appropriate value for the OHD must be specified in accordance with the SEOJ/DEOJ configuration in the message. Figure 4.17 shows the relationship between the property value notification (requiring a response) service and property value notification response service. The EPC sequence in the property value notification request service message must be equal to the EPC sequence in the property value notification response service message.

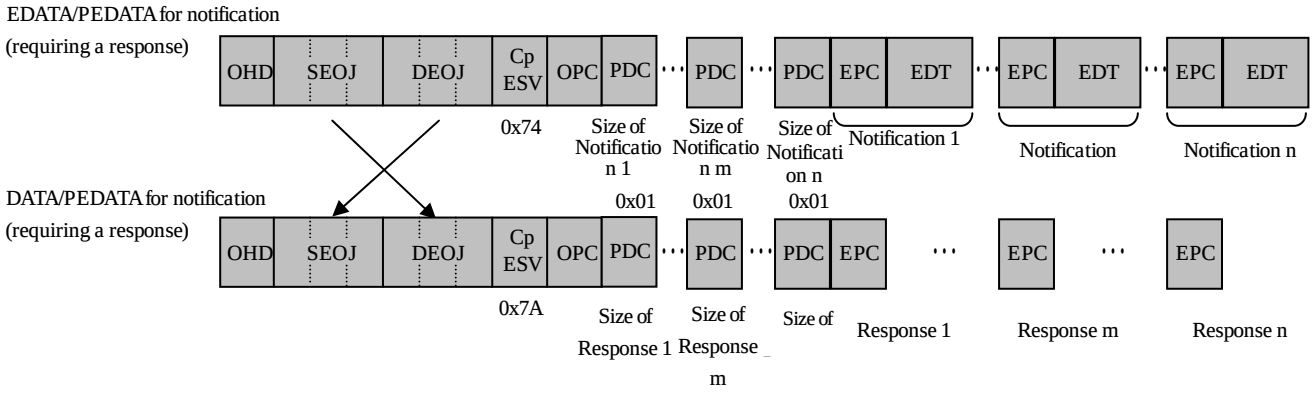


Fig. 4.17 Relationship Between Property Value Notification (Requiring a Response) and Property Value Notification Response

4.2.12 Processing Target Property Counter (OPC)

The processing target property counter is used in the compound message format only. It consists of one byte. In a compound message, the processing target property counter retains the number of properties targeted for a write or read operation. This counter can retain the value 1 or greater. Therefore, a compound message is allowed to exist even when the number of simultaneously operable properties is only one. The maximum number of simultaneously operable properties is limited by the maximum permissible message length.

If, for instance, there are three requests as shown in Fig. 4.18, the processing target property counter is 0x03.

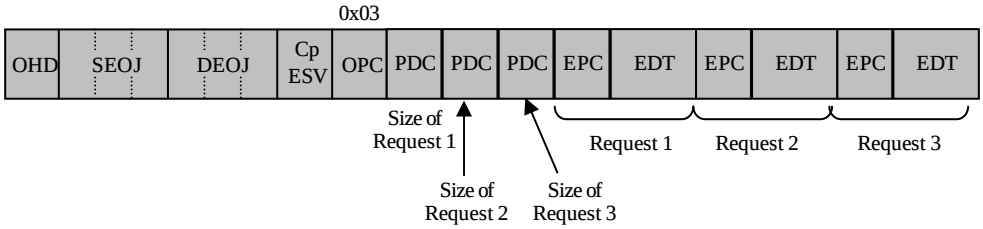


Fig. 4.18 Processing Target Property Counter for Three Requests

4.2.13 Property Data Counter (PDC)

The processing data counter is used in the compound message format only. It retains the number of bytes in the ECHONET property code (EPC) and ECHONET data (EDT), which follow the proper data counter. If, for instance, the ECHONET data sizes for Requests 1, 2, and 3 are 2 bytes, 1 byte, and 5 bytes, respectively, the values placed in the first, second, and third property data counters are 0x03, 0x02, and 0x06, respectively, as shown in Fig. 4.19.

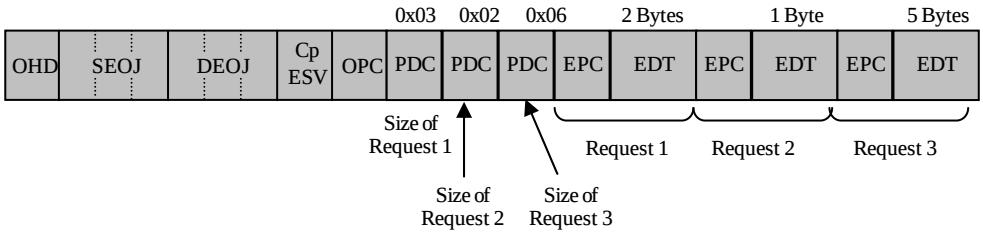


Fig. 4.19 Property Data Counter

Chapter 5 Basic Sequences

5.1 Basic Concept

Of the sequences exchanged between the ECHONET Communication Middleware (or more precisely, between ECHONET Communications Processing Blocks) for nodes connected to the ECHONET network, those that must be implemented are called "basic sequences." This section divides these basic sequences into four main categories for specification:

1) Basic sequences for object control
2) Basic sequences for node startup (1)
3) Basic sequences for node startup (2)
4) Basic sequences for node normal operation

ECHONET nodes are divided into devices that have the ECHONET Router functions and those without such functions. "2) Basic sequences for node startup (1)" shows the basic sequence for startup of ECHONET nodes in general, while "3) Basic sequences for node startup (2)" shows the basic sequence for startup of ECHONET devices that have the ECHONET Router or NetID Server functions. Table 5.1 defines the timeout and response waiting periods used in the basic sequences for startup.

Depending on the type of device, some of the basic sequences specified in this section, all of which are required, involve complex exchanges and thus entail much heavier communications processing than application processing. Therefore, the specifications were formulated to make the sequences as simple as possible.

The ECHONET Communications Processing Block's internal processing sequence that is performed at node startup is described in Section 6.7 "Startup Processing".

Table 5.1 Timeout and Response Waiting Periods

Name of Period	For Communication with → Nodes/Routers	Type of Communication → within/beyond subnet	Individual/Broadcasting	Classification
T1	Nodes	Within subnet	Individual	Timeout period
T2	Nodes	Beyond subnet	Individual	Timeout period
T3	Routers/NetID servers	Within subnet	Individual	Timeout period
T4	Routers/NetID servers	Beyond subnet	Individual	Timeout period
T5	Nodes	Within subnet	Broadcasting	Response waiting period
T6	Nodes	Beyond subnet	Broadcasting	Response waiting period
T7	Routers/NetID servers	Within subnet	Broadcasting	Response waiting period
T8	Routers/NetID servers	Beyond subnet	Broadcasting	Response waiting period

5.2 Basic Sequences for Object Control

ECHONET Communication Middleware exchanges are performed by stipulating the service (ESV:ECHONET service) with respect to the object property specified in the previous section. Basic sequences for objects can be broadly divided into basic sequences for object control in general and basic sequences for service content (see below). These two types are described below.

1) Basic sequences for object control in general
2) Basic sequences for service content

5.2.1 Basic Sequences for Object Control in General

The ECHONET Communication Middleware performs the following five processes as basic processing when it receives a service (specified in Table 4.11, Table 4.12 to 4.14) for an object property. The first three processes are described here. The fifth process (E) is described in the next section under Basic Sequences for Service Content.

A) Processing when the controlled object does not exist
B) Processing when the controlled object exists but the controlled property does not exist or the control content cannot be interpreted
C) Processing when both the controlled object and controlled property exist but the stipulated array element does not exist or the control content cannot be interpreted
D) Processing when the controlled property exists but the stipulated service processing functions are not available
E) Processing when the controlled property exists and the stipulated service processing functions are available

(A) The received ECHONET message is discarded, and no response is required.

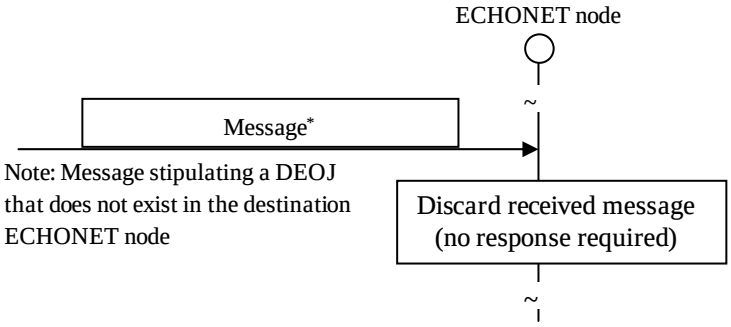


Fig. 5.1 Basic Sequence When Object to Be Controlled Does Not Exist

(B) Processing performed when the object to be controlled exists but the property to be controlled does not exist or the control content cannot be interpreted

The received ECHONET message is discarded, and the associated "process not possible" response (ESV = 0x50 to 0x5E, CpESV = 0x50 to 0x52) is returned. The figure below shows the basic sequence that is performed when a received request ESV = 0x6# (#: 0 to E), CpESV = 0x6# (#: 0 to 2) relates to an existing DEOJ and nonexistent EPC:

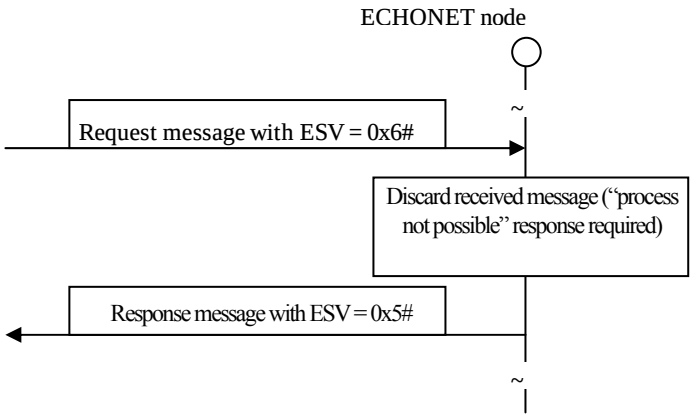


Fig. 5.2 Basic Sequence Performed When Object to Be Controlled Exists
But Property to Be Controlled Does Not Exist or Control Content Cannot Be Interpreted

(C) Processing performed when the object and property to be controlled exist but the specified array element does not exist or the control content cannot be interpreted

The received ECHONET message is discarded. However, the "process not possible" response (ESV = 0x54 to 0x5E) associated with the specified service (ESV = 0x64 to 0x6E) is returned. The figure below

shows the basic sequence:

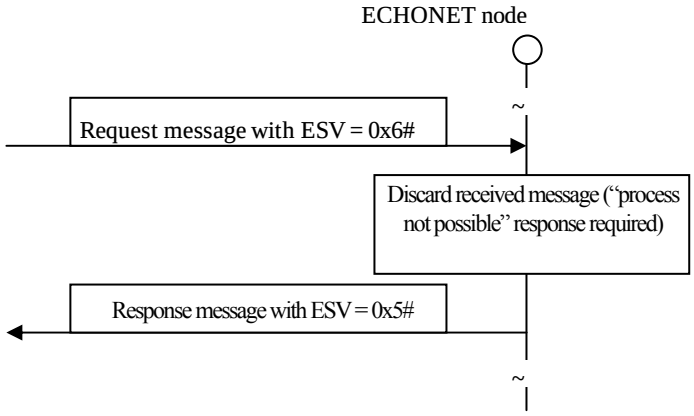


Fig. 5.3 Basic Sequence Performed When Object and Property to Be Controlled Exist But Specified Array Element Does Not Exist or Control Content Cannot Be Interpreted

(D) Processing when the controlled property exists but the stipulated service processing functions are not available

Same as for (B) above.

5.2.2 Basic Sequences for Service Content

The ECHONET Communication Middleware has three basic processing sequences for the reception of object property-related services (specified in Table 4.10), assuming the stipulated property exists and has service functions:

A) Basic sequence for receiving a request (no response required)
B) Basic sequence for receiving a request (response required)
C) Basic sequence for property value notification (autonomous notification)

(A) Basic sequence performed when a request requiring no result-indicating response is received

There are some operations (ESV = 0x60 to 0x6E, CpESV = 0x60 to 0x62) that an ECHONET node performs in relation to properties. The figure below shows the ECHONET node's basic sequence that is performed upon receipt of ESV = 0x60, 0x64, 0x68, 0x6A, or 0x6D, CpESV = 0x60 (no response required):

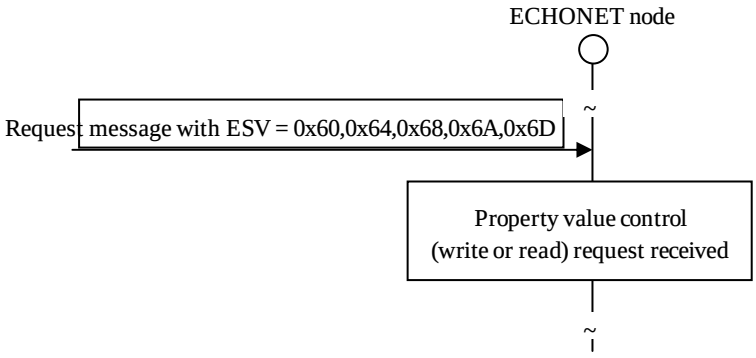


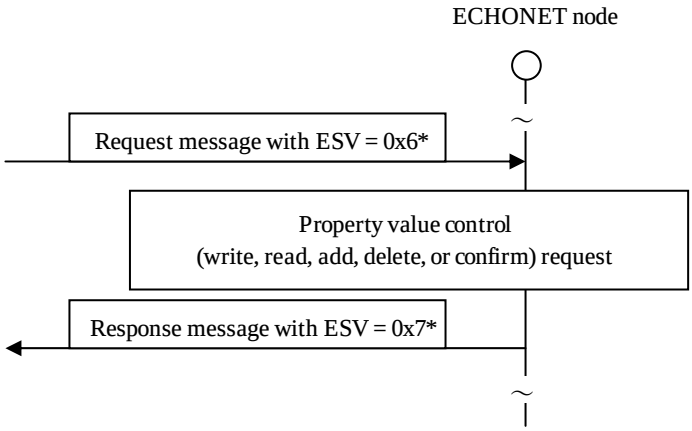
Fig. 5.4 Basic Request Receiving Sequence
for ESV = 0x60,0x64,0x68,0x6A,0x6D,CoESV = 0x60

(B) Basic sequence for receiving a request (response required)

Figure 5.5 shows the basic sequence, for each ESV, for an ECHONET node that has received a property value-related manipulation from another ECHONET node (ESV = 0x60 to 0x6E, CpESV = 0x60 to 0x62), where ESV = 0x61 to 0x63, 0x65 to 0x67, 0x69, 0x6B, 0x6C, 0x6E (response required).

・Basic request receiving sequence for ESV = 0x6* (*: 1,2,5,6,9,B,C,E), CpESV = 0x6* (*: 1, 2)

(Response is returned to request message source)



・Basic request receiving sequence for ESV = 0x6# (#: 3,7)

(Response returned using general broadcast)

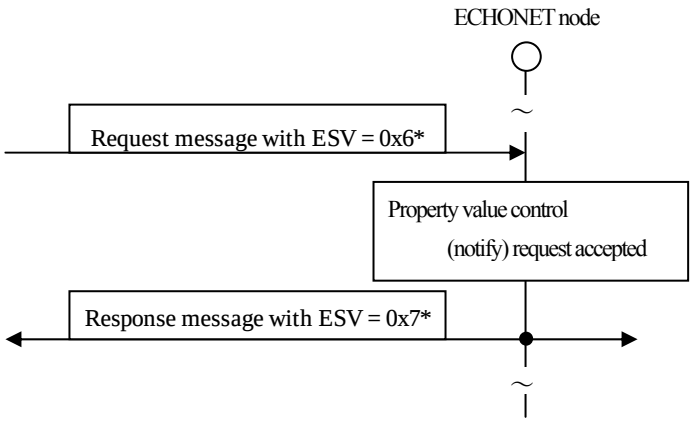


Fig. 5.5 Basic Request Receiving Sequence
for ESV = 0x6α (α: 1 to 3,5 to 7,9,B,C,E), CpESV = 0x6α (α: 1, 2)

(C) Basic sequence for property value notification

The figure below shows the basic sequence for properties that are required to notify their status when the object property value changes (i.e., when there is a change in the status setting from the application software).

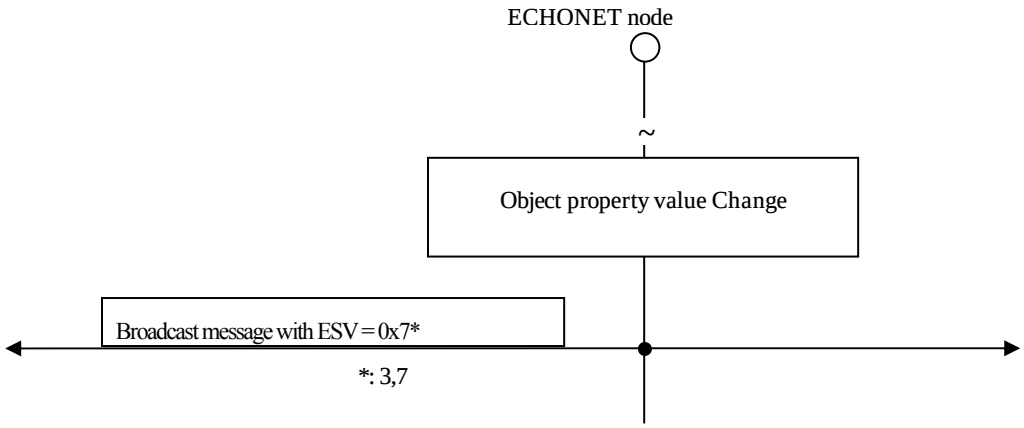


Fig. 5.6 Basic Sequence for Property Value Change

5.3 Basic Sequence for ECHONET Node Startup

For the ECHONET nodes described in this section, startup begins with the acquisition of an ECHONET address for self-recognition and specification. As was noted in Chapter 2 "ECHONET Addresses," an ECHONET address consists of a NodeID and a NetID.

This section specifies the following two NetID acquisition processing sequences, assuming that the NodeID has already been obtained when the ECHONET Communication Middleware begins operation:

(1) Basic sequence for cold start[*1]
(2) Basic sequence for warm start[*2]

This section describes the default router that appears in the basic sequence. The ECHONET Communication Middleware performs exchange without recognition of the relevant subnets by using ECHONET addresses that identify ECHONET nodes. Nodes having ECHONET addresses with the same NetID are part of the same subnet, and in the lower-layer communications software it is possible to send messages directly to another ECHONET node by specifying the MAC address. Meanwhile, nodes whose ECHONET addresses have different NetIDs belong to other subnets that are connected by routers. The concept of "default routers" was introduced to reduce the processing load on individual (non-router) ECHONET nodes when source ECHONET messages are sent to ECHONET nodes in other subnets. Individual (non-router) ECHONET nodes contain default router data, which consists of the ECHONET

address for one of the routers connected to the same subnet, when their NetIDs are set. This "default router data" comprises the lower-order bytes of the EDT and SEA at the time of receiving the NetID read response, which gives the NetID and NodeID of the default router, respectively. ECHONET messages to ECHONET nodes in other subnets can be sent to the default router irrespective of the subnet in which they are located (See "6.3.2 Routing Processing Specifications for Outgoing Messages") The default router is selected as follows:

(1) When the NetID is read by means of broadcasting to the routers in the home subnet during a cold start:
One of the responding ECHONET routers is selected as the default router and its EA is set in the default router data property. No selection criterion is specified (Section 5.3.1).

(2) When a NetID write request is received:
The requesting ECHONET router is selected as the default router and its EA is set in the default router data property (Section 5.5.3).

(3) When there is no response from the default router during a warm start and the NetID is read by means of broadcasting to the routers in the home subnet.
One of the responding ECHONET routers is selected as the default router and its EA is set in the default router data property. No selection criterion is specified (Section 5.3.2).

The sequences specified in this section are those for nodes whose NetIDs are within the NetID assignment code area. No requirements are specified for ECHONET nodes whose NetIDs are within the area open to users (0x90 to 0xFF).

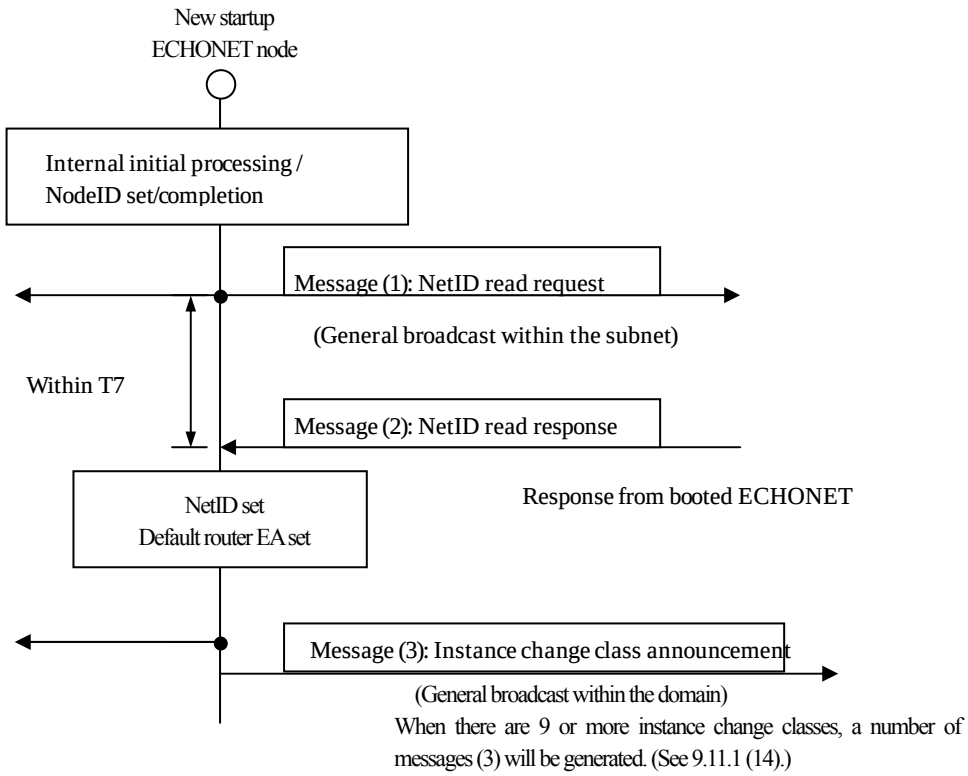
Notes: *1 <Cold start>
Start by resetting the ECHONET Communications Middleware and Lower-Layer Communications Software, which resets the ECHONET address.

*2 <Warm start>
Start with the NetID settings preserved.
Starts with the ECHONET address already set.

5.3.1 Basic Sequence for ECHONET Node Cold Start

During a cold start, the ECHONET node Communications Middleware obtains a NodeID from the lower-layer transmission medium or from the application software settings, and then obtains a NetID via ECHONET. Shown below is the basic sequence by which an ECHONET node acquires a NetID during a cold start.

The figure below shows the basic processing sequence after the NodeID has been set (i.e., after communication within the subnet via lower-layer transmission media becomes possible):



Message (1)	・Sets default value (x'00') not stipulated in SEA NetID. ・Uses the DEA to stipulate a general broadcast to all nodes within the home subnet (0x01FF). ・Stipulates (with DEOJ) router profile objects (0x0EF101)・No SEOJ requirement. ・Stipulates NetID properties (0xE1) with EPC.・Stipulates read request (0x62) with ESV.
Message (2)	・Individual response message. (The DEA is the SEA of Message (1). The SEA is the router EA whose NetID is 0x00.) ・Stipulates read request by Message (1) (SEOJ = 0x0EF101, No SEOJ requirement; EPC = 0xE1, ESV = 0x72, EDT = NetID data).
Message (3)	・Stipulates broadcast to all nodes within a domain (0x01FF) with DEA. ・Uses the SEA to stipulate own FA in accordance with the assigned NetID. ・Stipulates node profile objects (0x0EF001) with SEOJ.・No DEOJ requirement. ・Stipulates instance change class announcement property (0xD5) with EPC. ・Stipulates notification announcement (0x73) with ESV.
T7	Message (2) reception wait timeout. When Message (2) is not received by time T2 *(design guideline: 60 sec)*, 0x00 is assigned as the NetID.

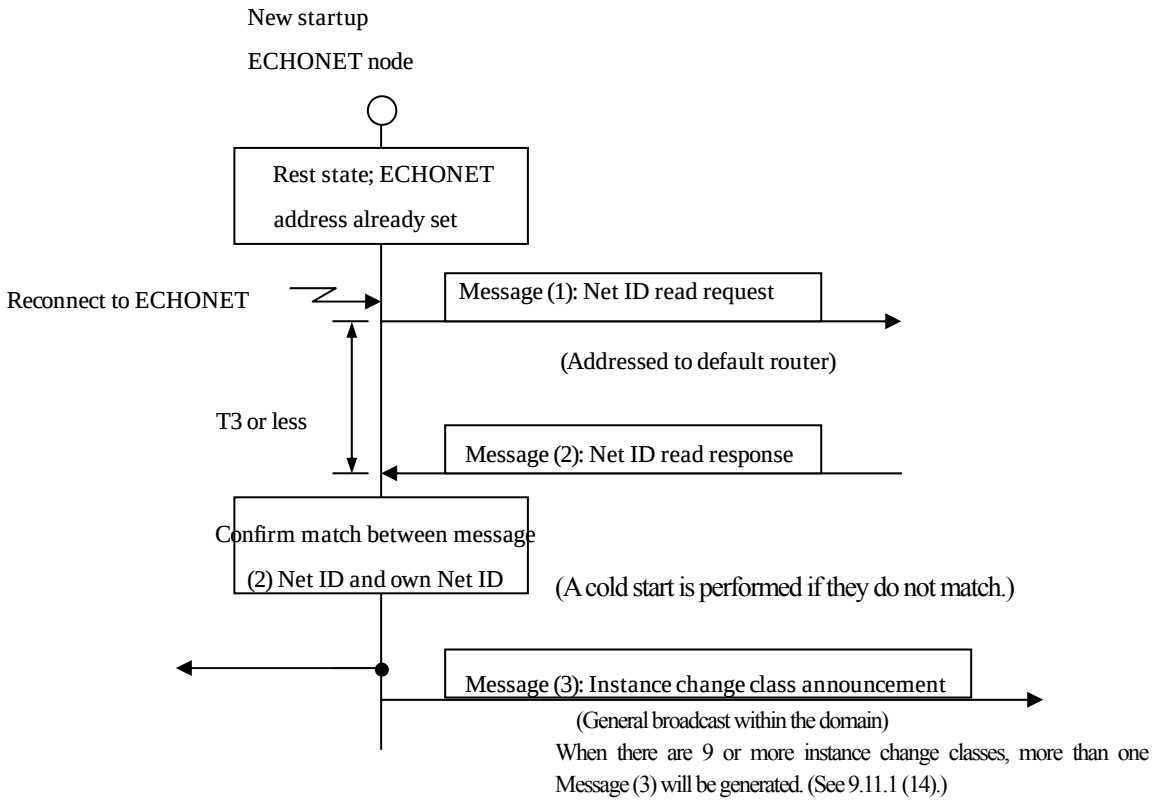
Fig. 5.7 Basic Sequence for ECHONET Node Startup (1)

5.3.2 Basic Sequence for ECHONET Node Warm Start

The figure below shows the basic sequence for ECHONET node warm starts. A warm start is a restart whereby the power is switched off once and then switched back on with the ECHONET address data (NetID, NodeID and default router data) retained.

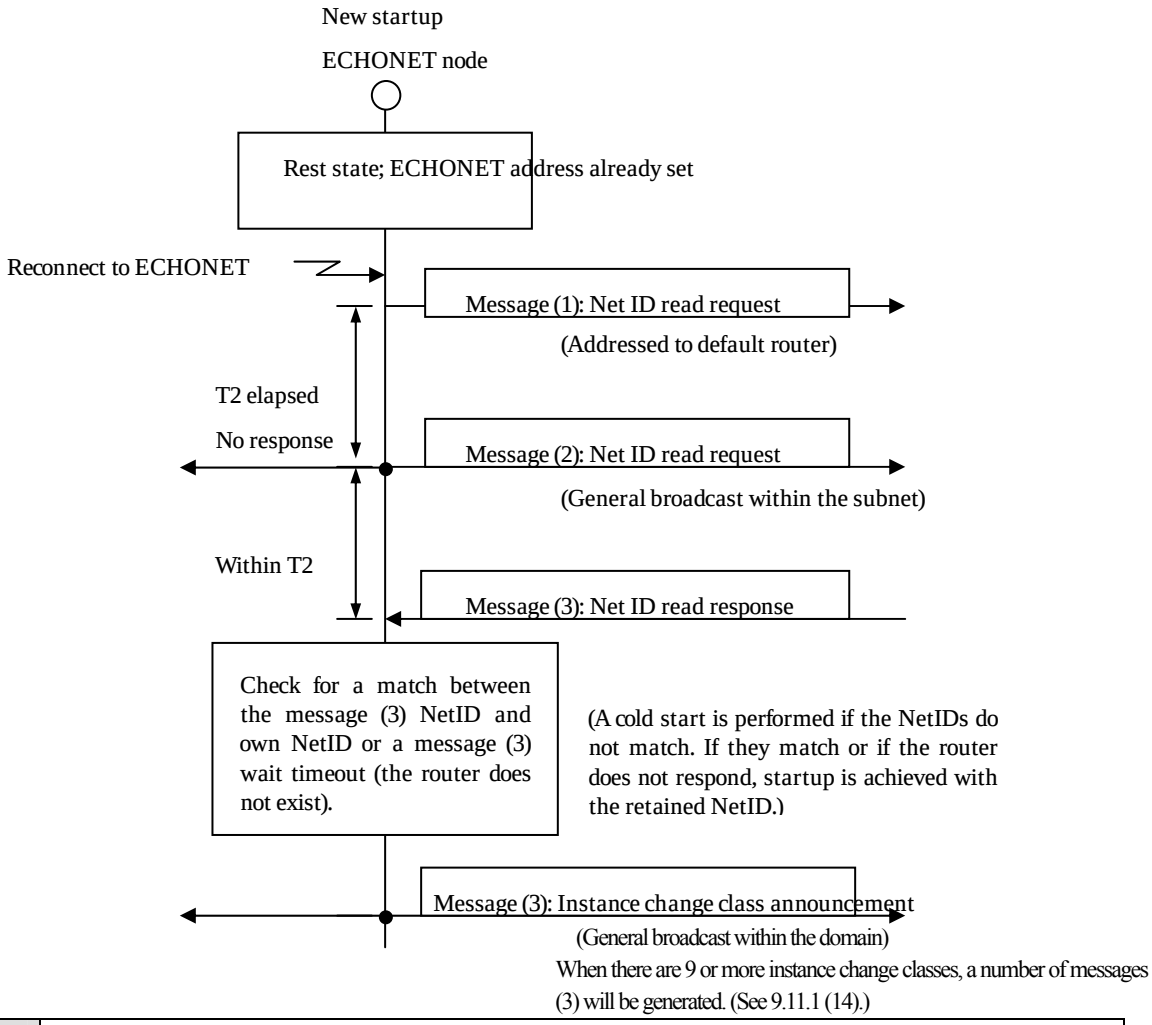
A warm start uses the existing NodeID and NetID during the startup process. However, a check shall be made before using the NetID to confirm that the NetID value did not change during a power-down period, etc. First, a NetID read request is sent to the default router based on the existing default router data. Then, a check is made to see whether or not the returned NetID matches the current NetID. Figure 5.8-1 shows the sequence for a case where the returned NetID matches the current NetID, and Fig. 5.8-2 shows the sequence for a case where no response to Message (2) is returned. If the NetID from Message (2) in Fig. 5.8-1 or Message (3) in Fig. 5.8-2 does not match the existing data, the processing for ECHONET node cold starts described in the previous section is performed.

When the current NetID value is "0x00" (i.e. a system without a router), the transmission of Message (1) may be omitted. Similarly, the transmission of Message (2) and the receiving of Message (3) and the NetID collation processing described in Fig. 5.8-2 may be omitted when the current NetID value is "0x00." However, it is recommended that the transmission of Message (2) and the receiving of Message (3) and the NetID collation processing be performed at a given time after startup.



Message (1)	・Sets own (retained) ECHONET address with SEA. ・Stipulates default router address with DEA. ・Stipulates (with DEOJ) router profile object (0x0EF101) ・No SEOJ requirement. ・Stipulates NetID properties (0xE1) with EPC. ・Stipulates read request (0x62) with ESV.
Message (2)	・Individual response message. (The DEA is the SEA of Message (1). The SEA is the router EA whose NetID is 0x00.) ・Stipulates read request by Message (1) (SEOJ = 0x0EF101, No SEOJ requirement , EPC = 0xE1, ESV = 0x72, EDT = NetID data).
Message (3)	・Stipulates broadcast to all nodes within a domain (0x00FF) with DEA. ・Stipulates node profile object (0x0EF001) with SEOJ. ・No DEOJ requirement. ・Stipulates instance change class announcement property (0xD5) with EPC. ・Stipulates notification announcement (0x73) with ESV.
T3	Message (2) reception wait timeout. When Message (2) is not received by time T3 *(design guideline: 60 sec)*, begin cold start processing.

Fig. 5.8-1 Basic Sequence for ECHONET Node Startup (2)



Message (1)	・Sets own (retained) ECHONET address with SEA. ・Stipulates default router address with DEA. ・Stipulates (with DEOJ) router profile object (0x0EF101) ・No SEOJ requirement. ・Stipulates NetID properties (0xE1) with EPC. ・Stipulates read request (0x62) with ESV.
Message (2)	(Stipulates broadcast to all nodes within a subnet with DEA. Others are the same as those of Message (1).)
Message (3)	・Individual response message. (DEA is SEA of Message (1); SEA is EA of router.) ・Stipulates read request by Message (1) (SEOJ = 0x0EF101, EPC = 0xE1, ESV = 0x72, EDT = NetID data).
Message (4)	・Stipulates broadcast to all nodes within a domain (0x00FF) with DEA. ・Stipulates node profile object (0x0EF001) with SEOJ. ・No DEOJ requirement. ・Stipulates instance change class announcement property (0xD5) with EPC. ・Stipulates notification announcement (0x73) with ESV.
T3	Timeout period for waiting to receive Message (2) (60 seconds – Design Guidelines).
T2	Timeout period for waiting to receive the NetID read response (60 seconds – Design Guidelines). If Message (3) is not received during this period, operation will be started using the current EA.

Fig. 5.8-2 Basic Sequence for ECHONET Node Startup (3)

5.4 Basic Sequences for Startup of NetID Servers and ECHONET Routers

ECHONET defines NetID servers and ECHONET routers with the objective of facilitating the creation of networks encompassing different subnets. Device specifications for NetID servers and ECHONET routers are provided in Part VII. This section describes the basic sequences for startup of NetID servers and ECHONET routers.

There can be only one NetID server within an ECHONET domain; this server assigns NetIDs to ECHONET routers.

When two or more ECHONET routers exist within a subnet, one ECHONET router is positioned on the path to the NetID server and is given the home router data by the NetID server before the other routers within the subnet. This router becomes the "master router" in the subnet (specified by the "master router data" property of the router profile object). In all subnets except the one that is closest to the NetID server (i.e. the one located on the path to the NetID server), any ECHONET router becomes the "master router (see Fig. 5.9)." The master router data is used in cold and warm starts of ECHONET routers.

The following 4 basic sequences are defined as the basic sequences for startup of NetID servers and ECHONET routers.

(1) Basic sequence for NetID server cold start
(2) Basic sequence for NetID server warm start
(3) Basic sequence for ECHONET router cold start
(4) Basic sequence for ECHONET router warm start

The sequences specified in this section are for those nodes whose NetIDs are within the NetID assignment code area. No requirements are specified for ECHONET nodes whose NetIDs are within the area open to users (0x90 to 0xFF).

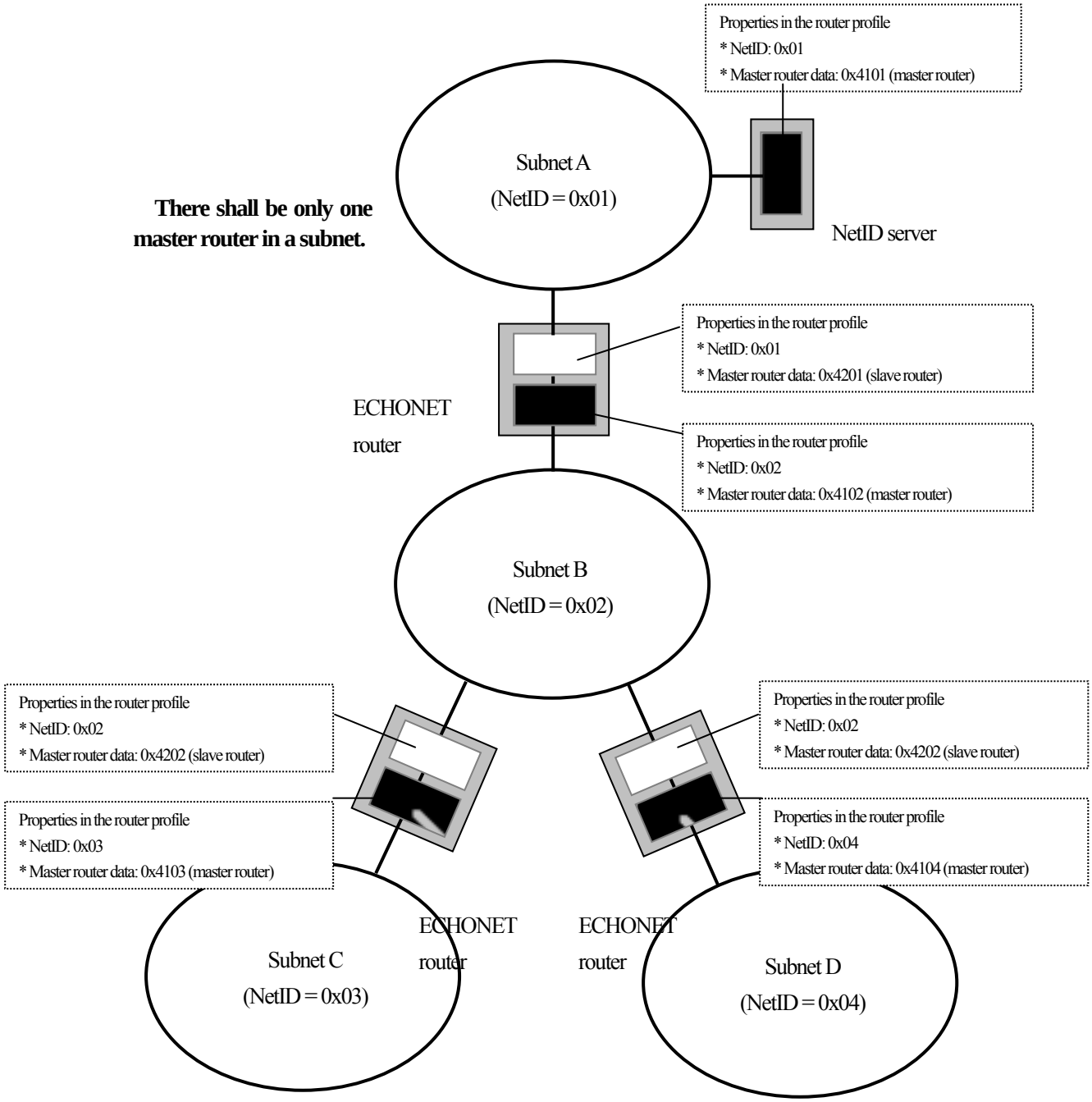


Fig. 5.9 Master Router Data

5.4.1 Basic Sequence for NetID Server Cold Start

There shall be only one NetID server within a domain. This NetID server is responsible for setting and managing NetIDs and router IDs of ECHONET routers. A NetID server cold start shall occur after a new device is installed or data is rewritten and shall be called when a NetID startup is specified (see Part VII, Section 2.4 "NetID Server Functions").

(1) Initial internal processing

During a cold start, the NetID server shall acquire the NodeID from the lower-layer transmission medium or application software settings.

(2) Startup as an ordinary node

The NetID server whose NodeID has been determined shall start up as an ordinary node, because the old one would be replaced. The NetID server shall check the NetID settings in the subnet, and, depending on the number of nodes detected and the NetID settings, behave as specified for one of the cases (CASEs 1 to 5) shown in Table 5.1.

If "there is no node with an established NetID," "there is only one node with an established NetID" or "there are two or more nodes with an established NetID and all such NetIDs are 0x00 or there is one or more different NetIDs," the NetID server shall newly set the NetIDs for that subnet within the NetID assignment code area (from 0x01 to 0x8F). If "there is no node with an established NetID," a shift to normal operation may be made immediately because it is implicitly shown that neither master router nor NetID server exists. If "there are two or more nodes with an established NetID and all such NetIDs are the same," it shall be presumed that a NetID server replacement is required and the detected NetID shall be used as the NetID of that subnet.

Table 5.2 Possible Actions after NetID Detection (Cold Start)

CASE	Number of Detected Nodes	NetID Setting	Processing
1	0	-	The NetID server newly sets the NetID for the subnet within the NetID assignment code area.
2	1	-	
3	Two or more	One or more different NetIDs	
4		All 0x00	
5		All the same	The detected NetID shall be used as the NetID server's subnet side NetID.

"-" Indicates that the item is not a case selection criterion.

(3) Confirmation that a master router is located nearby

A NetID server that has started up as an ordinary node shall make a check to see whether or not a

master router exists within the subnet, and if a master router exists, shall acquire the NetID server EA from the master router's NetID server data.

(4) Processing to avoid a situation in which the server coexists with an old NetID server

If a master router exists and the NetID server has successfully acquired the EA from the NetID server data, it shall send a "router function deactivation request" to the NetID server having that EA in the form of an individual transmission.

(5) Behavior of the NetID server that has received a router function deactivation request

The old NetID server that has received the router function deactivation request shall change its status in the master router data to "slave."

(6) NetID setting alteration processing

A new NetID server that has not detected any master router within the subnet in (3) above or a new NetID server that has replaced the old NetID server in accordance with (3) through (5) above shall make a simultaneous broadcast of a "NetID write request" throughout the closest subnet.

The ECHONET routers whose NetIDs have been rewritten by this shall perform the normal cold start sequence, write their home router data to the NetID server and update the data on all routers, and then make a simultaneous broadcast of a "NetID write request" throughout the subnet managed by the router.

The above sequence is performed in a "recursive" manner using the routers and ultimately, the NetID server's data on all routers is appropriately updated.

If, in the case where an old NetID server exists in (3) above, an ECHONET router is not able to detect the old NetID server and fails to start up successfully because of the detection of two master routers, it shall set "router startup failed" in the error description property of the router profile class and send it by means of a simultaneous broadcast throughout the domain.

(7) Handling of two or more subnets

If there are two or more subnets to be managed by the NetID server, the above sequence shall be performed for all such subnets.

A reference diagram for the above sequence is shown in Appendix 6.

5.4.2 Basic Sequence for NetID Server Warm Start

A NetID server warm start shall be used when a startup is to be performed without a NetID startup specified, such as when a recovery is necessary from a domain-wide simultaneous shutdown (due to a blackout, etc.). (See Part VII, Section 2.4 "NetID Server Functions.")

(1) Startup as an ordinary node

When a warm start is performed, the NetID server shall start up as an ordinary node, check the NetID settings in the subnet, and, depending on the number of nodes detected and the NetID settings, behave as specified for one of the 4 cases (CASEs 1 to 4) shown in Table 5.2.

That is, if the number of nodes detected is 0 or the number of nodes detected is 1 or more and all of the NetIDs match the NetID stored, the warm start sequence shall be continued. Otherwise, a shift is made to the cold start sequence.

Table 5.3 Possible Actions after NetID Detection (Warm Start)

CASE	Number of Detected Nodes	NetID Setting	Processing
1	0	-	Starts up using the current NetID.
2	One or more	One or more is different from the current value	Shifts to the cold start sequence.
3			
4		All 0x00	
		All are the same as the current value	Starts up using the current NetID.

"-" Indicates that the item is not a case selection criterion.

(2) Confirmation that a master router is located nearby

A NetID server that has started up as an ordinary node shall make a check to see whether a master router exists within the subnet, and if a master router exists, shall acquire the NetID server EA from the master router's NetID server data.

(3) Processing to avoid a situation in which the server coexists with an old NetID server

If a master router exists and the NetID server has successfully acquired the EA from the NetID server data, it shall make a check to confirm that the EA matches its own EA, and if it does not match its own EA, make a shift to the cold start sequence. If the EA matches its own EA, it shall terminate the startup sequence and start normal operation (no NetID write request shall be broadcast in the subnet).

A reference diagram for the above sequence is shown in Appendix 6.

5.4.3 Basic Sequence for ECHONET Router Cold Start

ECHONET routers are divided into automatic setting routers and manual setting routers. An automatic setting router is an ECHONET router that automatically acquires the NetID from the NetID server and a manual setting router is an ECHONET router that requires manual NetID setting by the user. ECHONET routers perform startup processing that is different from that for ordinary nodes for each of the connected

subnets. An ECHONET router performs a cold start (i) when it newly joins a network, (ii) when a warm start fails, or (iii) when it receives a write request for the NetID for the node profile class of the node held by the ECHONET router.

The conditions for an ECHONET router to operate in the "normal operation" mode are explained below for each case.

(1) Startup conditions for a cold start of an automatic setting router

1) One master router exists in a subnet to which an automatic setting router that is going to perform a cold start is connected. If no master router is detected, or two or more master routers are detected, the router must not start up as an ECHONET router.

2) The home router data and the data on all routers are acquired from the NetID server. If it is not possible to communicate with the NetID server, it shall not start up as an ECHONET router.

Table 5.4 summarizes the startup conditions for a cold start of an automatic setting router.

Table 5.4 Startup Conditions for a Cold Start of an Automatic Setting Router

CASE	Number of master routers detected	Communication with the NetID server	Processing
1	Two or more	-	Does not start up as a router.
2	0	-	Does not start up as a router.
3	1	Not possible	Does not start up as a router.
4		Possible	Performs the startup processing as a router.

(Note) "-" means "don't care."

The basic sequence for a cold start of an automatic setting router is as follows:

First, the initial internal processing is performed and the home NodeID setting is made. The automatic setting router then broadcasts, in the form of intra-subnet simultaneous broadcasts, a master router data read request (addressed to the router profile) in each of the subnets to which it is connected. If the number of master routers detected is "0" or "2 or more" after receiving the master router data read response and acquiring the master router data, it shall not start up as an ECHONET router, in which case it shall start up as an ordinary node and the following processing shall not be performed.

The automatic setting router broadcasts, in the form of intra-subnet simultaneous broadcasts, a NetID read request (addressed to the node profile) in each of the subnets to which it is connected. The ECHONET router performs the processing even if it receives a direct message from another node having a NetID that is different from the home NetID. The ECHONET router retains, as a provisional NetID, the NetID contained in the received NetID read response.

The automatic setting router then broadcasts, in the form of intra-subnet simultaneous broadcasts, a

NetID server data read request to the router profile objects in each of the subnets having a master router. If the number of NetID servers detected is "0" or "2 or more" after acquiring the NetID server data, it shall not start up as a router, in which case it shall start up as an ordinary node and the following sequence shall not be performed.

The automatic setting router sends a read request for the data on all routers (addressed to the router profile) in each of the subnets to which it is connected. Data on all routers is acquired from the ECHONET routers residing in all subnets that are connected to the automatic setting router in the process of cold starting. If two or more "accept" response messages are received from one subnet and the values of the data on all routers received in one message are different from those received in another, it shall start up as an ordinary node with no router startup processing performed and the following sequence shall not be performed.

After acquisition of the data on all routers, a registration request router data write request is sent to the NetID server. The registration request router data is as follows:

Router attribute: 0x01 (with automatic setting function)

Home router ID: 0x00

Number of connected subnets: Number of subnets to which the router is connected

EA data:

The router shows this data to the connected subnets (The router lists the ECHONET addresses). The NetID of subnets which are master routers shall be 0x00, and the NetID's of other subnets shall be provisional NetID's.

The NetID's of slave routers in the EA data at the time of transmission of a registration request router data write request shall be values acquired from node profiles located in the connected subnets (provisional NetID's), and the NetID of master routers shall be 0x00.

If no home router data write request is received from the NetID server within the specified period of time after the transmission of the registration request router data write request, the registration request router data write request shall be resent. If a router registration status notification (EDT: router registration busy state 0x30) is received from the NetID server, the registration request router data write request shall be resent after the specified period of time elapses. If a home router data write request is received from the NetID server, a home router data write response shall be sent to the NetID server. If no write request for the data on all routers is received from the NetID server within the specified period of time after the transmission of the home router data write response, the registration request router data write request shall be resent. If a write request for the data on all routers is received from the NetID server, a write response for the data on all routers shall be sent to the NetID server. If no ECHONET router registration completion notification is received from the NetID server within the specified period of time after sending the write response for the data on all routers, the registration request router data write request shall be resent. If an ECHONET router registration completion notification is received from the NetID server, an ECHONET router registration completion notification response shall be sent to the NetID server. Upon completion of the transmission of the

ECHONET router registration completion notification response to the NetID server, the ECHONET router shall start up as an ECHONET router.

If all the acquired sets of data on all routers are the same, and the NetID contained in the home router data write request received from the NetID server is different from the NetID contained in the NetID read responses received from the node profiles in the subnets to which it is connected, the ECHONET router shall send, in the form of intra-subnet simultaneous broadcasts, a NetID write request addressed to the node profiles in the subnets having the different NetIDs using the NetID acquired from the home router data write request received from the NetID server. If the received sets of data on all routers are different, the ECHONET router shall send, in the form of intra-subnet simultaneous broadcasts, a NetID write request to the subnets to which it is connected and which do not have a master router, using the NetID acquired from the home router data write request received from the NetID server.

(2) Conditions for newly joining a network as a manual setting router

1) In a subnet to which a manual setting router that is going to perform a cold start is connected, there can be one or no master router. If two or more master routers are detected, the router must not start up as a router.
2) If the NetID that the manual setting router wants to set is being used in the domain, the router must not start up as a router.

Table 5.5 summarizes the startup conditions for a cold start of a manual setting router.

Table 5.5 Possible Occurrences When Cold Starting a Manual Setting Router

CASE	Number of detected master routers	Communication with the NetID server	NetID overlapping	Processing
1	Two or more	-	-	Does not start up as a router.
2	0	-	-	Performs the startup processing as a router.
3	1	Not possible	-	Performs the startup processing as a router.
4		Possible	Does not overlap.	Performs the startup processing as a router.
5			Overlapping occurs.	Does not start up as a router.

(Note) "-" means "don't care."

Manual setting routers shall start the sequence without losing the NetID that is set during the cold start.

The basic sequence for a cold start of a manual setting router is as follows:

First, the initial internal processing is performed and the home NodeID is set. Then, the manual setting router sends, in the form of intra-subnet simultaneous broadcasts, a master router data read request to the router profiles in all subnets to which it is connected. The manual setting router then receives a master router data read response or master router data read responses and acquires master router data. Depending on the content of the master router data, the manual setting router performs one of the following processing sequences:

(1) If no master router data response is received from the subnets:

The manual setting router shall start up as an ECHONET router and send, in the form of intra-subnet simultaneous broadcasts, a NetID write request to the node profiles in all subnets to which it is connected. The manual setting router shall then start operating as a router in normal operation mode. The written NetID's must be different between subnets.

(2) If a master router data response is received and the second byte (NetID data) of the received data is different from the NetID that the manual setting router intended to assign to the responded subnet:

The manual setting router must start up as an ordinary node and must not start up as an ECHONET router. The manual setting router shall deem that an abnormal condition has occurred, set the value of the abnormal condition description property of the router profile object to 0x0010 and broadcast it in the form of an intra-domain simultaneous broadcast.

(3) If a master router data response is received and the number of master routers detected is 1:

The manual setting router shall send, in the form of an intra-subnet simultaneous broadcast, a NetID server data property (EPC = 0xE3) read message to the router profile object of the responded subnet. Depending on the content of the response to the message, one of the following processing sequences ((a), (b) and (c)) shall be performed:

(a) If the response shows that 2 or more NetID servers exist:

The manual setting router shall deem that an abnormal condition has occurred, set the value of the abnormal condition description property of the router profile object to 0x0010 and broadcast it in the form of an intra-domain simultaneous broadcast. The manual setting router shall start up as an ordinary node.

(b) If the response shows that one NetID server exists:

The manual setting router shall send a "data on all routers" property (EPC = 0xE4) read request to the router profile objects in all subnets to which it is connected. The manual setting router shall then check the acquired data on all routers to confirm whether a subnet exists which has a NetID that is the same as a NetID set by the manual setting router.

If such a subnet exists, the manual setting router shall deem that an abnormal condition has occurred, set the value of the abnormal condition description property of the router profile object to 0x0010 and broadcast it in the form of an intra-domain simultaneous broadcast. The manual setting router shall start up as an ordinary node, and the following router startup sequence shall not be

performed.

If no such subnet exists, the manual setting router shall write an unused NetID to all nodes in the subnets for which the manual setting router is the master router. The manual setting router shall then send the registration request router data to the NetID server. Both the router attribute and home router ID in the registration request router data shall be 0x00.

If the manual setting router does not receive a request from the NetID server for a write to the home router data property of the router profile object within the specified period of time after the transmission of the registration request router data, or the manual setting router receives from the NetID server a notification that the value of the router registration status property of the NetID server object is "busy," the manual setting router shall resend the registration request router data to the NetID server.

If the manual setting router receives a request from the NetID server for a write to the home router data property of the router profile object within the specified period of time after the transmission of the registration request router data, the manual setting router shall send a response to the NetID server.

The manual setting router shall then check the values it received. If there is a discrepancy between the value of the second byte (number of subnets connected) or the third and succeeding bytes (EA data) and the value it intends to set, the manual setting router shall start up as an ordinary node and the following sequence shall not be performed.

If there is no such discrepancy, the manual setting router shall write the values to its home router data property. If the manual setting router does not receive a request from the NetID server for a write to the "data on all routers" property of the router profile object within the specified period of time after the writing, the manual setting router shall resend the request for a write to the registration request router data property of the NetID server profile object. If the manual setting router receives a request from the NetID server for a write to the "data on all routers" property of the router profile object within the specified period of time after the writing, the manual setting router shall write the values to its "data on all routers" property and send a response to the NetID server.

If the manual setting router does not receive from the NetID server a notification about the ECHONET router registration completion notification property of the NetID server profile class within the specified period of time after the transmission of the response, the manual setting router shall resend the request for a write to the registration request router data property of the NetID server profile object.

If the manual setting router receives from the NetID server a notification about the ECHONET router registration completion notification property of the NetID server profile class within the specified period of time after the transmission of the response, the manual setting router shall send a

response to the NetID server and then start operating in normal operation mode.

If the acquired sets of data on all routers are different, the manual setting router (that has started up) must send, in the form of intra-subnet simultaneous broadcasts, a NetID write request to the subnets to which it is connected and which do not have a master router, using a NetID that it has set.

(c) If the response shows that no NetID server exists:

The manual setting router shall send a "data on all routers" property (EPC = 0xE4) read request to the router profile objects in all subnets to which it is connected. The manual setting router shall then check the acquired data on all routers to confirm whether a subnet exists which has a NetID that is the same as a NetID set by the manual setting router. If no such subnet exists, the manual setting router shall generate data on all routers that includes a NetID the manual setting router has set and write it to the "data on all routers" property of the router profile object of each of the ECHONET routers that exist. The manual setting router shall then start operating in normal operation mode.

If such a subnet exists, the manual setting router shall deem that an abnormal condition has occurred, set the value of the abnormal condition description property of the router profile object to 0x0010, broadcast it in the form of an intra-domain simultaneous broadcast and start up as an ordinary node.

(4) If a master router data response is received and the number of master routers detected is 0:

The manual setting router shall start up as an ECHONET router. The manual setting router shall send, in the form of intra-subnet simultaneous broadcasts, a NetID write request to the node profiles in all subnets to which it is connected. The manual setting router shall then start operating as a router in normal operation mode. The written NetID's must be different between subnets.

(5) If a master router data response is received and the number of master routers detected is 2 or more:

The manual setting router must start up as an ordinary node and must not start up as an ECHONET router. The manual setting router shall deem that an abnormal condition has occurred, set the value of the abnormal condition description property of the router profile object to 0x0010 and broadcast it in the form of an intra-domain simultaneous broadcast.

5.4.4 Basic Sequence for ECHONET Router Warm Start

For an ECHONET router to perform a warm start, it must have once performed a cold start and joined a network. During the warm start processing, the ECHONET router may perform routing based on the data it acquired at the time of the previous startup, but it does not necessarily have to do this.

① If two or more master routers are detected in a subnet to which an ECHONET router that is going to perform a warm start is connected, the ECHONET router shall not start up as a router.
② If no master router is detected in a subnet to which an ECHONET router that is going to perform a warm start is connected, the ECHONET router shall not start up as an ECHONET router if it is an automatic setting router.
③ If the configuration of a subnet to which an ECHONET router that is going to perform a warm start is connected is different from that at the time of the previous startup, the ECHONET router shall not start up as an ECHONET router.
④ If the data on all routers acquired by an ECHONET router that is going to perform a warm start is different from that at the time of the previous startup, the ECHONET router shall not start up as an ECHONET router.

Table 5.6 summarizes the conditions for a warm start of an ECHONET router.

Table 5.6 Possible Occurrences When Warm Starting an ECHONET Router

CASE	Number of master routers detected	Configuration of the connected subnet	Data on all routers from the master router or NetID server	Manual setting router / automatic setting router	Processing
1	Two or more	-	-	-	Processing B
2	0	Same	-	Automatic setting router	Processing C
3				Manual setting router	Processing A
4		Different	-	-	Processing C
5	1	Same	Same	-	Processing A
6			Different	-	Processing C
7		Different	-	-	Processing C

(Note) "-" means "don't care."

Processing A: Starts up using the data it has held since before the warm start.
Processing B: Starts up as an ECHONET node, with the router functions deactivated.
Processing C: Shifts to the cold start processing, with the router functions deactivated.

(1) Conditions for a warm start of an automatic setting router

The basic sequence for a warm start of an automatic setting router is as follows:

First, the initial internal processing is performed and the home NodeID is set. The automatic setting router then sends, in the form of intra-subnet simultaneous broadcasts, a master router data read request to the router profiles in each of the subnets to which it is connected. If it is found that the number of master routers detected is 2 or more after receiving the master router data read response and acquiring the master router data, it shall not start up as an ECHONET router, in which case it shall start up as an ordinary node and the following sequence shall not be performed. If the number of master routers detected is 0, a cold start shall be performed.

The automatic setting router broadcasts, in the form of intra-subnet simultaneous broadcasts, a NetID read request to the subnets to which it is connected. It shall then compare (based on the acquired NetID value) the configurations of the subnets with those at the time of the previous startup, and if there is a difference, it shall shift to ECHONET router cold start.

The automatic setting router broadcasts, in the form of intra-subnet simultaneous broadcasts, a NetID server data read request in each of the subnets having a master router. If the received NetID server data is different from the NetID server data at the time of the previous startup, it shall shift to ECHONET router cold start.

If a NetID server exists, the automatic setting router shall send a read request for the data on all routers directly to the NetID server. If no NetID server exists, the automatic setting router shall send to the master routers, in the form of intra-subnet simultaneous broadcasts, a read request for the data on all routers. If the data on all routers is different from that at the time of the previous startup, it shall shift to ECHONET router cold start. If the acquired data on all routers does not contain its own EA, it shall shift to ECHONET router cold start. If the data on all routers is the same as that at the time of the previous startup and contains its own EA, it shall start up as an ECHONET router.

(2) Conditions for a warm start of a manual setting router

The basic sequence for a warm start of a manual setting router is as follows:

First, the initial internal processing is performed and the home NodeID is set. The manual setting router then sends, in the form of intra-subnet simultaneous broadcasts, a master router data read request to the router profiles in each of the subnets to which it is connected. If it is found that the number of master routers detected is 2 or more after receiving the master router data read response and acquiring the master router data, it shall not start up as an ECHONET router, in which case it shall start up as an ordinary node and the following sequence shall not be performed.

The manual setting router shall send, in the form of intra-subnet simultaneous broadcasts, a NetID read request to the node profile objects of the connected subnets. It shall then compare (based on the

acquired NetID value) the configurations of the subnets with those at the time of the previous startup, and if there is a difference, it shall shift to ECHONET router cold start. If the number of master routers detected is 0 and the configurations of the subnets are the same as those at the time of the previous startup, it shall start up as an ECHONET router based on the conditions at the time of the previous startup.

The ECHONET router broadcasts, in the form of intra-subnet simultaneous broadcasts, a NetID server data read request in each of the subnets having a master router. If the received NetID server data is different from that at the time of the previous startup, it shall shift to the ECHONET router cold start sequence.

If a NetID server exists, the ECHONET router shall send a read request for the data on all routers directly to the NetID server. If no NetID server exists, the ECHONET router shall send to the master routers, in the form of intra-subnet simultaneous broadcasts, a read request for the data on all routers. If the data on all routers is different from that at the time of the previous startup, it shall shift to the ECHONET router cold start sequence. If the acquired data on all routers does not contain its own EA, it shall shift to the ECHONET router cold start sequence. If the data on all routers is the same as that at the time of the previous startup and contains its own EA, it shall start up as an ECHONET router.

5.5 Basic Sequence for ECHONET Node Normal Operation

During normal operation, the ECHONET node performs the sequence described in Section 5.2 *Basic Sequence for Object Control*. To prevent system operating errors, the following special sequences are specified for processing by the ECHONET Communication Middleware.

Basic sequence for detecting ECHONET address (EA) duplication
Basic sequence for detecting nodes with improper NetIDs
NetID (basic sequence for the processing after receiving a write request)

The basic sequences stated in this section relate to all nodes. Processing also covers ECHONET nodes whose NetIDs are within the range open to users (0x90 to 0xFF).

5.5.1 Basic Sequence for Detecting EA Duplication

The following types of ECHONET address (EA) duplication are considered:

Duplicate NodeID setting within the subnet (this means MAC address duplication)
Duplicate NetID setting

Case ① could occur during a warm start or when an ECHONET node is powered on and moved to another subnet. Here, communication itself would be impossible because an error would be detected in the

lower-layer transmission media. Therefore, the communications sequence for detecting EA duplication shall not be specified. For devices capable of warm starts, however, ECHONET Communication Middleware saves the duplicated EA. Therefore, application software designers should take this into account.

Case ② could occur in either of the two situations described for Case ① and also when an EA is manually set. In the latter case, communications over the lower-layer transmission media would be possible as long as there is no duplication of MAC addresses. Therefore, the following sub-cases are presented. In the first case, router functions are specified and manual setting of NetIDs for the router subnets is not permitted. (Note, however, that manual setting of ECHONET node EAs is not specified.) In the second case, methods for avoiding the problem in the node are specified in Section 5.5.2 "Basic Sequence for Detecting Nodes with Improper Net IDs".

Subnets with the same NetID exist within the same domain, causing duplication of NetIDs.

Subnet has a (duplicate) NetID that is the same as that of another subnet in a domain having a different NetID. Consequently, there is no duplication of MAC addresses within the subnet, allowing communications over the transmission media.

5.5.2 Basic Sequence for Detecting Nodes with Improper NetIDs

One NetID is set for each subnet, and its value is unique within the domain. In the two cases listed below, a device having a NetID that is different from the set NetID could exist within the subnet.

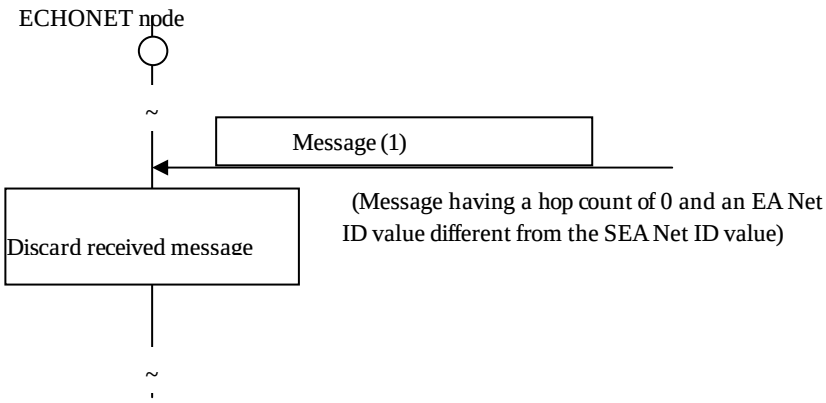
① A device active in another subnet is moved into the current subnet.
② There exists a device currently performing startup processing within the subnet for which a NetID has been assigned.

In Case ②, the device does not become operative until the router properly assigns the NetID in the startup sequence, so there is no need to specify a new sequence. In Case (1), when an improper NetID is detected, the received message is discarded even when the DEA is intended for the node's self-EA in order to prevent a system error. A NetID error is detected when the hop count for the received message is 0 and the NetID value of the received message SEA differs from the NetID value of the node's self-EA (Fig. 5.10). However, messages from the SEA having a NetID setting of 0x00 will be processed in compliance with special specifications even if the hop count is 0.

For ECHONET devices that operate while moving from one subnet to another without turning OFF the power, applications should be designed on the presumption that no response might be returned in the following sequence.

For a router, however, a response will be returned in response to a message concerning a router profile property read request, even if the hop count is 0 and the received message SEA's NetID value differs from the NetID retained by the router. Here, the NetID value for the SEA in the response message is 0x00.

Here,



Message (1)	・Hop count is 0. ・SEA NetID differs from node's own NetID (except when NetID = 0x00). ・DEA specifies destination ECHONET node (regardless of individual/broadcast). ・EOJ, EPC, and ESV are optional.

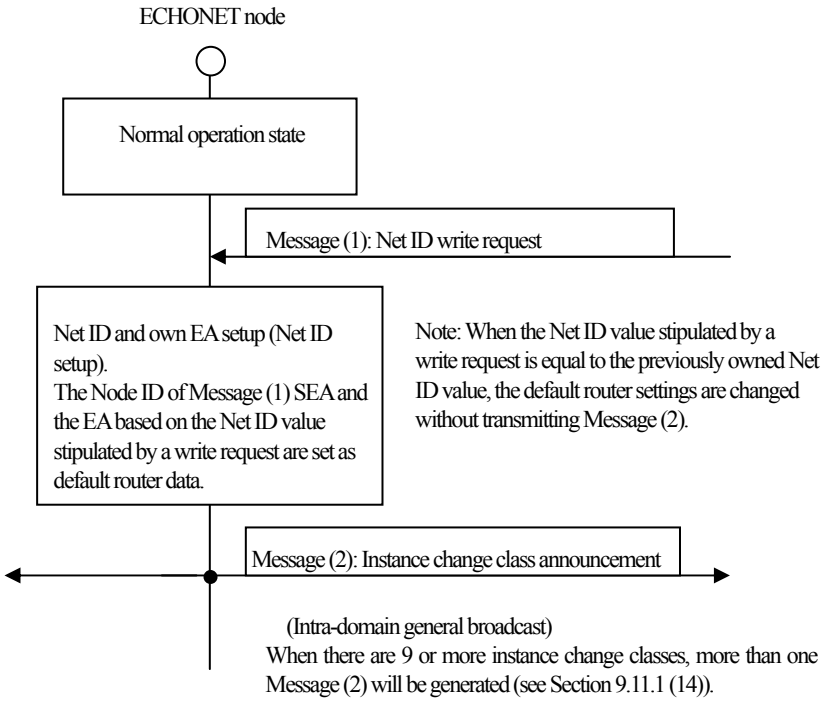
Fig. 5.10 Basic Sequence for Detecting Improper Node NetID

5.5.3 Basic Sequence for NetID Write Request Reception

NetID write requests to nodes that are not an ECHONET router shall only be made by master routers. A master router shall only accept write requests from NetID servers, and a NetID server shall not accept any NetID write request.

A "non-ECHONET router" node that receives a NetID write request shall in principle perform the sequence processing shown in Fig. 5.11, but it may perform a cold start or a warm start using the written NetID as the NetID of its own EA.

An ECHONET router that receives a NetID write request shall stop operating and perform the cold start processing. The basic sequence for this is described in 5.4.3.



Message (1)	・SEA for NetID = 0x00 or the same NetID as for the target node. ・Stipulates node profile object (0x0EF001) with DEOJ. ・Stipulates router profile object (0x0EF101) with SEOJ. ・Stipulates NetID property (0xE1) with EPC. ・Stipulates write request (0x60) with ESV.
Message (2)	・Stipulates intra-domain general broadcast (0x00FF) with DEA. ・Stipulates node profile object (0x0EF001) with SEOJ. ・Nothing is stipulated with DEOJ. ・Stipulates instance change class announcement property (0xD5) with EPC. ・Stipulates notification (0x73) with ESV.

Fig. 5.11 Basic Sequence for ECHONET Node NetID Setting Change

5.6 Basic Sequence for Normal NetID Server Operation

The normal operation of a NetID server shall be defined as the state in which the NetID server can perform the processing to respond to a write or read request for an object that it holds and can perform the NetID determination and dissemination processing.

NetID determination and dissemination processing

The "NetID determination and dissemination processing" shall mean the processing in response to a request from an ECHONET router that is in the process of cold or warm starting.

5.6.1 NetID Server Processing

If a NetID server receives a registration request router data write request from an ECHONET router, the NetID server must perform the NetID determination and dissemination processing for the ECHONET router that is the sender of the registration request router write data. However, if registration request router data write data is received from another ECHONET router during this processing, no NetID determination and dissemination processing shall be performed for the ECHONET router that is the sender of the registration request router write data and a response shall be sent to the ECHONET router that is the sender of the registration request router write data to notify it that the NetID determination and dissemination processing cannot be performed.

The NetID determination and dissemination processing for a new startup of an ECHONET router consists of (1) the processing to confirm the data on all routers, (2) the new registration router data setting processing and (3) the processing to update the data on all routers. Each of these is explained below.

(1) Processing to confirm the data on all routers

The processing to confirm the data on all routers shall mean the processing to respond to the following request messages from an ECHONET router:

1) NetID server data read request message

If a message requesting a read of the NetID server data property of the router profile object is received, a response message (NetID server data read response message) shall be sent back within the specified period of time.

2) Read request message for the data on all routers

If a message requesting a read of the "data on all routers" property of the NetID server profile object is received, a response message (read response message for the data on all routers) shall be sent back within the specified period of time.

New registration router data setting processing

① If a message is received from an ECHONET router that requests a write to the registration request router data property of the NetID server profile object, it must generate the home router data after confirming the registration request router data and send a message, within the specified period of time, that requests a write of the generated home router data to the home router data property of the router profile object of the ECHONET router that is requesting the registration. The router registration status property must be set to "busy" prior to performing these processing steps. The processing step to confirm the registration request router data and the processing step to generate the home router data

are as follows:

1) Router function check

If bit 1 of the router property of the home router data of the registration request router data is 0 (i.e. the registration requesting node does not have the router functions), the registration request router data shall be used as the home router data to be written into the registration requesting ECHONET router except that 0x00 shall be used as the home router ID.

2) Processing to be performed when the home router ID is not 0x00

If the home router ID in the registration request router data is not 0x00, a check shall be made to confirm that there is a router in the domain that has the same router ID as the home router ID through collation with the data on all routers held. If there is such a router, the number of subnets associated with that router and the EA data held shall be collated with the registration request router data, and if they match the registration request router data, the registration request router data shall be used as the home router data to be written into the requesting ECHONET router. If there is any item that does not match, the registration request router data shall be used as the home router data to be returned except that 0x00 shall be used as the home router ID.

3) Processing to be performed in relation to potential replacement routers

A check shall be made to confirm that there is a router whose home router ID in the registration request router data is 0x00 and the number of networks connected in the registration request router data and the NetID in the EA data match ("potential replacement router") using the data on all routers held. If there is such a router, the home router data of the potential replacement router shall be used as the home router data. After the transmission of a home router data write request, an attempt must be made to stop the operation of the potential replacement router by setting the home router ID in the home router data to 0x00 and sending that home router data to the potential replacement router.

4) Processing to be performed in relation to loop-forming routers

If the home router ID in the registration request router data is 0x00 and a collation of the registration request router data with the data on all routers held shows that the location of the registration requesting router is such that it forms a loop, a check shall be made to confirm that there is a slave router in the subnet in which the requesting router is the master router.

If there is such a router (see Fig. 5.12), the registration request router data shall be used as the home router data except that the home router ID shall be 0x00.

If there is no slave router (see Fig. 5.13), a new router ID shall be created in such a way that no ID overlap occurs and the registration request router data shall be used as the home router data after incorporating the newly created router ID as the home router ID. In this case, an attempt must be made to stop the operation of the master router of the subnet in which the requesting router is the master router by sending the home router data to the master router after setting the home router ID to 0x00.

5) Processing to be performed in relation to routers that are not a loop-forming or potential replacement router

If the home router ID in the registration request router data is 0x00 and neither 3) nor 4) above applies, a new router ID shall be created in such a way that no ID overlap occurs and the

registration request router data shall be used as the home router data after incorporating the newly created router ID as the home router ID.

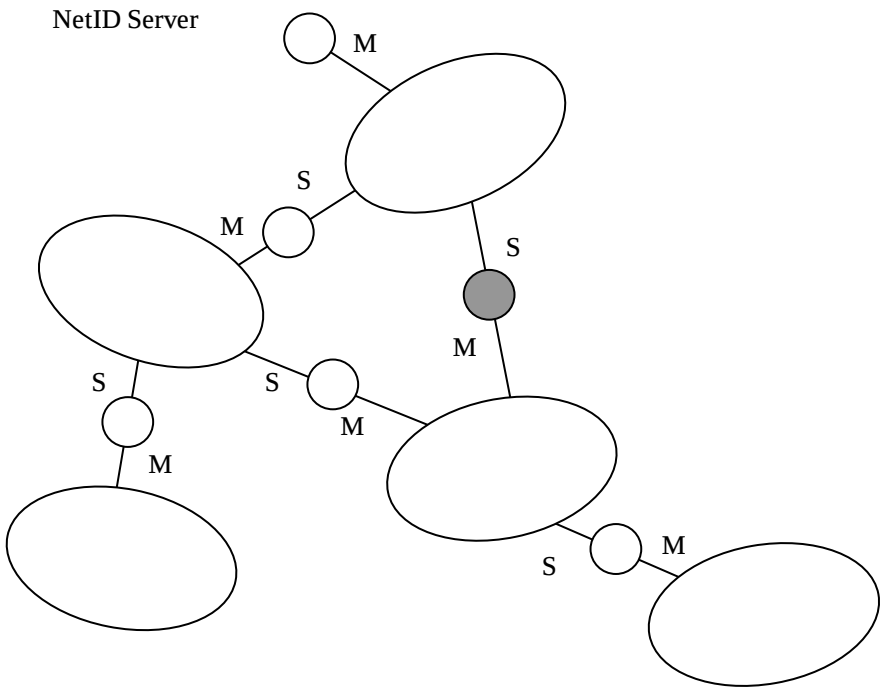


Fig. 5.12 Example Case: Slave Router Existing in the Loop-Forming Router's Master Side Subnet

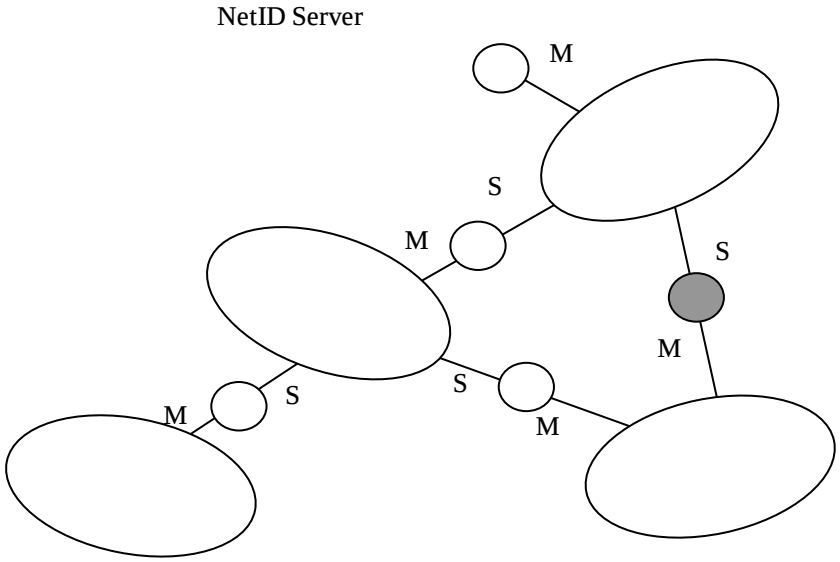


Fig. 5.13 Example Case: No Slave Router in the Loop-Forming Router's Master Side Subnet

② If a response message to a request to write data to the home router data property of the router profile object is received from an ECHONET router, a message shall be sent within the specified period of time that requests a write of 'the data on all routers to which the requesting router's data has been added' to the "data on all routers" property of the router profile object of the ECHONET router.

③ If a response message for a write to the "data on all routers" property is received from an ECHONET router, the registration completion notification property of the NetID server profile object held shall be set to 0x00 and the registration completion notification property shall be reported to the registration requesting ECHONET router within the specified period of time.

④ If a response message to a registration completion notification property notification is received within the specified period of time after the notification, the new ECHONET router registration processing shall be completed and the router registration status property shall be set to "ready." If no response message is received within the specified period of time after the notification, the registration processing shall be terminated and the router registration status property shall be set to "ready."

⑤ If the NetID server receives a request message from another ECHONET router while performing Sequences ① through ④ for a write to the registration request router property of the NetID server profile object, it shall set the registration completion notification property value of the NetID server profile object to 0x00 and notify the registration requesting ECHONET router of the new registration completion notification property within the T4 period.

If a request message for a write to the registration request router data property of the NetID server profile object is received during the processing, ① through ⑤ above shall be repeated for the requesting ECHONET router.

If the NetID server does not receive a response message to a request message that it sent for a write to the home router data property of the router profile object within the T4 period after transmission of the request message, it shall resend the request message for a write to the home router data property. There is no requirement with regard to the number of retransmission attempts.

If the NetID server does not receive a response message to a request message that it sent for a write of 'the data on all routers to which the requesting router's data has been added' to the "data on all routers" property of the router profile object within the T4 period after transmission of the request message, it shall resend the request message for a write to the "data on all routers" property. There is no requirement with regard to the number of retransmission attempts.

If the NetID server does not receive a response message to its notification within the T4 period after setting the registration completion notification property value of its NetID server profile object to 0x00 and notifying the registration requesting ECHONET router of the new registration completion notification property, it shall resend the message to notify the registration completion notification property. There is no requirement with regard to the number of retransmission attempts.

(2) Processing to update the data on all routers

The NetID server shall send individually transmitted messages requesting a write of the data on all routers created by processing (2) to the "data on all routers" properties of the router profile objects of the ECHONET routers under its control with the exception of the newly started up ECHONET routers and receive and confirm the response messages.

If there is an ECHONET router that does not respond (with a response message) to the message requesting a write to the "data on all routers" property of the router profile object of the ECHONET router within the T4 period after the transmission of the request message, the request message for a write to the "data on all routers" property shall be resent to the ECHONET router. There is no requirement with regard to the number of retransmission attempts.

5.7 Basic Sequence for Normal Operation of ECHONET Routers

The normal operation of an ECHONET router shall be defined as the state in which the ECHONET router can perform the processing to respond to a write or read request for an object it holds and can perform the following processing:

(1) Received message routing processing
(2) Default router processing

The received message routing processing is the processing that is performed when an ECHONET router is requested to perform message routing and the ECHONET router is located on the direct routing path, whereby the received message is sent to a node located in an adjacent subnet other than the adjacent subnet that received the message. An adjacent subnet shall mean a subnet whose location is such that an ECHONET router can exchange messages directly with that subnet (see Fig. 5.14).

The default router processing is the processing that is performed when an ECHONET router is requested to perform message routing and the ECHONET router is not located on the direct routing path, whereby the received message is sent to a router located in the subnet that received the message and on the direct routing path for the received message.

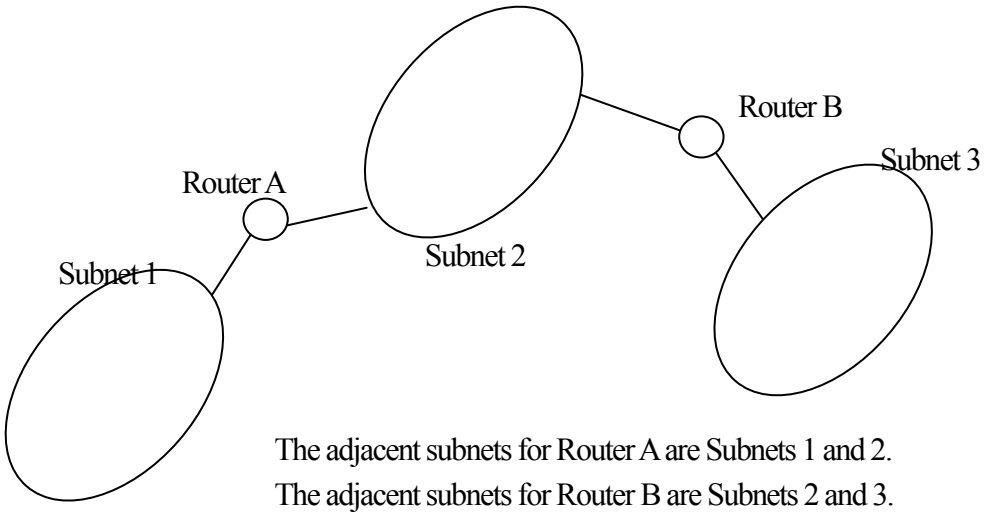


Fig. 5.14 Subnet Connections

When an ECHONET router is requested to perform message routing, it shall perform the received message routing processing if any one of the following conditions is satisfied:

- The DEA code type in the EHD is "individual" and the path to the subnet specified by the NetID in the DEA is located on the side of the adjacent subnet that is different from the adjacent subnet that received the message.
- The DEA code type in the EHD is "broadcast," the DEA broadcasting type code is 0x02 and the path to the subnet specified by the NetID specified by the broadcasting target code is located on the side of the adjacent subnet that is different from the adjacent subnet that received the message,' or 'the DEA code type in the EHD is "broadcast" and the DEA broadcasting type code is 0x00.'

If neither of these conditions is satisfied, it shall attempt to perform the default router processing. If the processing cannot be completed successfully, the received message must be discarded. Each of the two types of processing is explained below.

5.7.1 Received Message Routing Processing

When an ECHONET router receives a message, it shall check the ECHONET frame of the message, and if one of the following conditions is satisfied, perform the received message routing processing:

- b3 of the EHD is 0 (individual message) and the DEA is different from its own DEA (individual message processing).
- b3 of the EHD is 1 (broadcast message) and the destination NetID (broadcasting type code) is 0x00 (intra-domain broadcasting processing).
- 'b3 of the EHD is 1 (broadcast message), the destination NetID (broadcasting type code) is 0x02 and the destination NodeID (broadcasting target code) is different from its own NetID (subnet-specific broadcasting message processing for a subnet different from the one it belongs to),' or 'b3 of the EHD

is 1 (broadcast message) and the destination NetID (broadcasting type code) is 0x03 (intra-domain group broadcasting message processing).'

The received message routing processing shall include the following:

(1) A check to confirm that it is justifiable to perform routing processing

The received message routing processing shall be terminated if:

- The value of the hop counter in the EHD is 7 (maximum hop value error);
- The DEA code type in the EHD is "individual" and the NetID in the DEA is not included in the ECHONET router's own data on all routers (destination unknown);
- The DEA code type in the EHD is "individual" and the NetID in the DEA is included in the ECHONET router's own data on all routers, but the path thereto is not included (path unknown);
- The DEA code type in the EHD is "broadcast," the DEA broadcasting type code is 0x02, and the NetID specified by the broadcasting target code is not included in the ECHONET router's own data on all routers (destination unknown); or
- The DEA code type in the EHD is "broadcast," the DEA broadcasting type code is 0x02, the NetID specified by the broadcasting target code is included in the ECHONET router's own data on all routers, but the path thereto is not included (path unknown).

(2) Adding 1 to the hop counter value

The EHD hop counter value shall be increased by 1.

(3) Determination of the destination adjacent subnet

Based on the data on all routers, the adjacent subnet located on the path to the subnet that has the destination ECHONET router or node shall be determined.

(4) Transmission request

The lower-layer communication software shall be requested to send the processed ECHONET frame as an ECHONET message and the received message routing processing shall be terminated.

- If individual transmission is specified and the destination node is located in the selected adjacent subnet, the lower-layer communication software shall be requested to send the ECHONET frame to that node.
- If subnet-specific broadcasting is specified and the selected adjacent subnet is the destination subnet for the subnet-specific broadcast, the lower-layer communication software shall be requested to broadcast the ECHONET frame.
- If individual transmission or subnet-specific broadcasting is specified and there is an ECHONET router that is located on the routing path to the destination subnet for the subnet-specific broadcast or the destination node located in the selected adjacent subnet, the lower-layer communication software shall be requested to send the ECHONET frame to that ECHONET router.
- If intra-domain simultaneous broadcast is specified, the lower-layer communication software shall be requested to broadcast the ECHONET frame to all adjacent subnets with the exception of the adjacent subnet that received the message.

5.7.2 Default Router Processing

If an ECHONET router receives a message, it shall check the ECHONET frame of the message, and if one of the following conditions is satisfied, perform the default router processing:

- b3 of the EHD is 0 (individual message) and the DEA is different from its own DEA (individual message processing).
- b3 of the EHD is 1 (broadcast message), the destination NetID (broadcasting type code) is 0x02, and the destination NodeID (broadcasting target code) is different from its own NetID (subnet-specific broadcasting processing for a subnet different from the one it belongs to).

The default router processing shall include the following:

(1) A check to confirm that it is justifiable to perform default router processing

The default router processing shall be terminated if:

- The DEA code type in the EHD is "individual" and the NetID in the DEA is not included in the ECHONET router's own data on all routers (destination unknown);
- The DEA code type in the EHD is "individual" and the NetID in the DEA is included in the ECHONET router's own data on all routers, but the path thereto is not included (path unknown);
- The DEA code type in the EHD is "broadcast," the DEA broadcasting type code is 0x02, and the NetID specified by the broadcasting target code is not included in the ECHONET router's own data on all routers (destination unknown); or
- The DEA code type in the EHD is "broadcast," the DEA broadcasting type code is 0x02, and the NetID specified by the broadcasting target code is included in the ECHONET router's own data on all routers, but the path thereto is not included (path unknown).

(2) Transmission request

The lower-layer communication software shall be requested to send the ECHONET frame as an ECHONET message and the processing shall be terminated. The destination subnet shall be the adjacent subnet that received the message.

- If individual transmission or subnet-specific broadcasting is specified and there is an ECHONET router that is located on the routing path to the destination subnet for the subnet-specific broadcast or the destination node located in the destination adjacent subnet, the lower-layer communication software shall be requested to send the ECHONET frame to that ECHONET router.

5.8 Types of Messages that ECHONET Nodes Are Not Allowed to Send

This section specifies the types of messages that ECHONET nodes are not allowed to send, for each of the 3 types of ECHONET nodes; ordinary nodes, NetID servers and ECHONET routers.

5.8.1 Ordinary nodes

Ordinary nodes must not send the following types of messages:

- Request for a write to EA (EPC: 0xE0) of node profile class
- Request for a write to NetID (EPC: 0xE1) of node profile class
- Request for a write to “default router data” (EPC: 0xE3) of node profile class
- Request for a write to “data on all routers” (EPC: 0xE4) of node profile class
- Request for a write to router profile class
- Request for a write to NetID server profile class

Ordinary nodes that are not equipped with the secure communication common key setting node class are prohibited from sending the following types of messages, in addition to being prohibited from sending the above-mentioned types of messages:

- Request for a write to “secure communication common key setting (User Key)” (EPC: 0xC0) of node profile class
- Request for a write to “secure communication common key setting (Service Provider Key)” (EPC: 0xC1) of node profile class
- Request for a write to “secure communication common key switchover setup (User Key)” (EPC: 0xC2) of node profile class
- Request for a write to “secure communication common key switchover setup (Service Provider Key)” (EPC: 0xC3) of node profile class

5.8.2 NetID servers

NetID servers must not send the following types of messages:

- Request for a write to EA (EPC: 0xE0) of node profile class with NetID specified to be 0x00
- Request for a write to EA (EPC: 0xE0) of node profile class with NetID specified to be a user-defined area
- Request for a write to NetID (EPC: 0xE1) of node profile class with NetID specified to be 0x00
- Request for a write to NetID (EPC: 0xE1) of node profile class with NetID specified to be a user-defined area
- Distribution of NetID to ECHONET routers with NetID specified to be 0x00
- Distribution of NetID to ECHONET routers with NetID specified to be a user-defined area

NetID servers that are not equipped with the secure communication common key setting node class are prohibited

from sending the following types of messages, in addition to being prohibited from sending the above-mentioned types of messages:

- Request for a write to “secure communication common key setting (User Key)” (EPC: 0xC0) of node profile class
- Request for a write to “secure communication common key setting (Service Provider Key)” (EPC: 0xC1) of node profile class
- Request for a write to “secure communication common key switchover setup (User Key)” (EPC: 0xC2) of node profile class
- Request for a write to “secure communication common key switchover setup (Service Provider Key)” (EPC: 0xC3) of node profile class

5.8.3 ECHONET routers

ECHONET routers must not send the types of messages specified below. The types of messages that must not be sent differ depending on whether the ECHONET router in question is a master router or a slave router. The types of messages that master routers must not send are as follows:

- Request for a write to EA (EPC: 0xE0) of node profile class with NetID specified to be 0x00
- Request for a write to EA (EPC: 0xE0) of node profile class with NetID specified to be a user-defined area
- Request for a write to NetID (EPC: 0xE1) of node profile class with NetID specified to be 0x00
- Request for a write to NetID (EPC: 0xE1) of node profile class with NetID specified to be a user-defined area
- Request to a node located in an unconnected subnet for a write to EA (EPC: 0xE0) of node profile class
- Request to a node located in an unconnected subnet for a write to NetID (EPC: 0xE1) of node profile class
- Request for a write to “operating status” (EPC: 0x80) of NetID server profile class

Master routers that are not equipped with the secure communication common key setting node class are prohibited from sending the following types of messages, in addition to being prohibited from sending the above-mentioned types of messages:

- Request for a write to “secure communication common key setting (User Key)” (EPC: 0xC0) of node profile class
- Request for a write to “secure communication common key setting (Service Provider Key)” (EPC: 0xC1) of node profile class
- Request for a write to “secure communication common key switchover setup (User Key)” (EPC: 0xC2) of node profile class
- Request for a write to “secure communication common key switchover setup (Service Provider Key)” (EPC: 0xC3) of node profile class

The types of messages that slave routers must not send are as follows:

- Request for a write to EA (EPC: 0xE0) of node profile class
- Request for a write to NetID (EPC: 0xE1) of node profile class
- Request for a write to router profile class (In the case of a manual setting router, data on all routers shall be excluded.)
- Request for a write to NetID server profile class

Slave routers that are not equipped with the secure communication common key setting node class are prohibited from sending the following types of messages, in addition to being prohibited from sending the above-mentioned types of messages:

- Request for a write to “secure communication common key setting (User Key)” (EPC: 0xC0) of node profile class
- Request for a write to “secure communication common key setting (Service Provider Key)” (EPC: 0xC1) of node profile class
- Request for a write to “secure communication common key switchover setup (User Key)” (EPC: 0xC2) of node profile class
- Request for a write to “secure communication common key switchover setup (Service Provider Key)” (EPC: 0xC3) of node profile class

Chapter 6 ECHONET Communications Processing Block Processing Specifications

6.1 Basic Concept

This section presents the specifications for ECHONET communications processing in the ECHONET Communication Middleware as shown in the figure below. Note that the processes shown in the figure are used simply to describe basic processing in the ECHONET Communications Processing Block and are not intended as specifications for actual software structure.

(1) Received message judgment processing
(2) Routing processing
(3) Object processing
(4) Basic API processing
(5) Send message assembly and management processing
(6) Startup processing

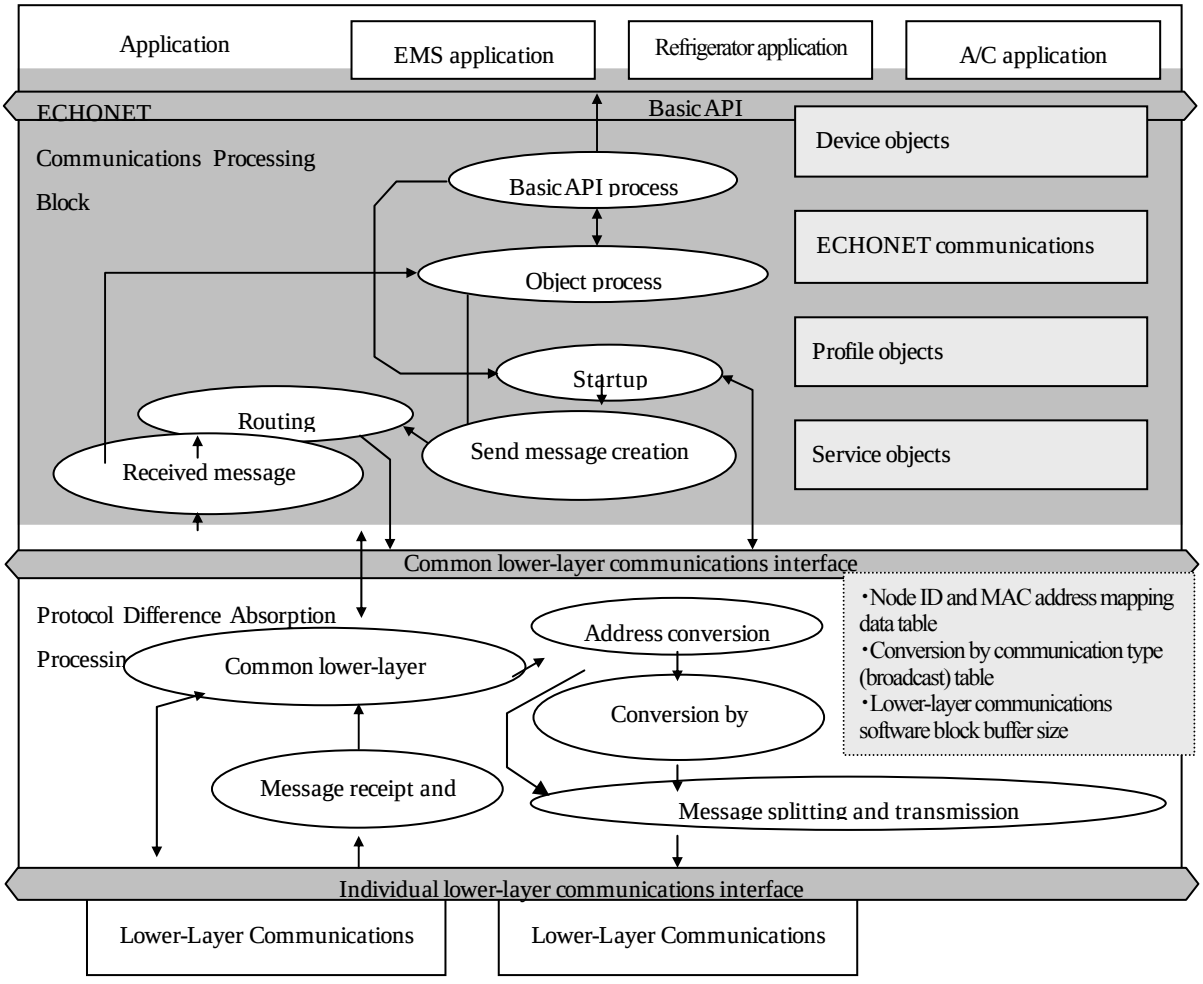


Fig. 6.1 Overview of Communication Middleware Processing (Layer Configuration)

6.2 Received Message Determination Processing Specifications

Processing shall be performed to confirm the recipients of messages received from the common lower-layer communication interface. The received message identification processing specifications are divided into those for ECHONET routers and those for non-ECHONET router devices. Each of the 2 types of processing is defined below. The home EA value, the home subnet NetID value, etc. shall be retained as properties of the "node profile class" of the profile object.

(1) Received message identification processing specifications for non-ECHONET router devices
If none of the cases ① to ⑦ listed below applies, the received message shall be discarded and the processing shall be terminated.

① b3 of the EHD is 0 (individual message), the value of the hop counter in the EHD is 0, the NetID in the SEA is the same as the NetID in the home EA, and the DEA matches the home EA.
② b3 of the EHD is 1 (broadcast message), the value of the hop counter in the EHD is 0, the NetID in the SEA is the same as the NetID in the home EA, the broadcasting type code is 0x00 or 0x01, and the broadcasting target code matches the home node group.
③ b3 of the EHD is 1 (broadcast message), the value of the hop counter in the EHD is 0, the NetID in the SEA is the same as the NetID in the home EA, the broadcasting type code is 0x02, and the broadcasting target code points to the home NetID.
④ b3 of the EHD is 1 (broadcast message), the value of the hop counter in the EHD is 0, the NetID in the SEA is 0x00 or the same as the NetID in the home EA, the broadcasting type code is 0x03 and the broadcasting target code is a group broadcast number that has been set for the home node. However, the intra-domain group broadcast function is an optional function. In the case of nodes that do not have the intra-domain group broadcast function, the received message shall be discarded and the processing shall be terminated.
⑤ b3 of the EHD is 1 (broadcast message), the value of the hop counter in the EHD is 0, the NetID in the SEA is 0x00 or the same as the NetID in the home EA, the broadcasting type code is 0x04 and the broadcasting target code is an inside group broadcast number that has been set for the home node. However, the intra-home subnet group broadcast function is an optional function. In the case of nodes that do not have the intra-home subnet group broadcast function, the received message shall be discarded and the processing shall be terminated.
⑥ b3 of the EHD is 0 (individual message), the value of the hop counter in the EHD is 1-7, the NetID in the SEA is not the same as the NetID in the home EA, and the DEA matches the home EA.
⑦ b3 of the EHD is 1 (broadcast message), the value of the hop counter in the EHD is 1-7, the NetID in the SEA is not the same as the NetID in the home EA, the broadcasting type code is 0x00, and the broadcasting target code is 0xFF or matches the home node group.
⑧ b3 of the EHD is 1 (broadcast message), the value of the hop counter in the EHD is 1-7, the NetID in the SEA is not the same as the NetID in the home EA, the broadcasting type code is 0x02, and the broadcasting target code points to the home NetID.
⑨ b3 of the EHD is 1 (broadcast message), the value of the hop counter in the EHD is 1 to 7, the NetID in the SEA is not the same as the NetID in the home EA, the broadcasting type code is 0x03 and the

broadcasting target code is a group broadcast number that has been set for the home node. However, the intra-domain group broadcast function is an optional function. In the case of nodes that do not have the intra-domain group broadcast function, the received message shall be discarded and the processing shall be terminated.Even if one of the cases listed above applies, the received message shall be discarded and the processing shall be terminated if:

⑩ A node that does not have the secure communication function receives a message for which the EHD b2 value is 1 (secure message);

⑪ A node that does not have the composite message processing function receives a message for which the EHD b1 and b0 are 0 and 1, respectively (composite message);

⑫ A node that does not have the group broadcast function receives a message for which b3 of the EHD is 1 (broadcast message) and the broadcasting type code is 0x03 or 0x04.

⑬ A message is received for which the EHD b7 is 0;

⑭ A message is received for which the EHD b1 and b0 are 1 and 0, respectively or a message is received for which the EHD b1 and b0 are 1 and 1, respectively;

⑮ The DEOJ of the received message is different from the home EOJ;

⑯ A message is received for which the EPC b7 is 0;

⑰ A message is received for which the b7 and b6 values are other than 0 and 1, respectively (ESV and CpESV);

⑱ A message is received for which the secure message SKH b11, b10 and b9 are values other than 0, 0 and 0, respectively;

⑲ A message is received for which the secure message SKH b5 and b4 are values other than 0 and 0, respectively; or

⑳ The BCC of the received message is different from the BCC calculated from the received message.

(2) Specifications for the received message identification processing for ECHONET routers

In the case of an ECHONET router, the recipient of the received message shall be confirmed (based on the EHD and DEA data), and if one of the cases listed below applies, the processing of the received message shall be handed only to the routing processing part. With regard to ① and ②, however, one of the sets of processing specifications shall be implemented.

① The DEA code (designation of type) in the EHD specifies individual transmission (b3 = 0), the DEA does not match the home EA, and the NetID in the DEA does not match the NetID in the home EA.

② The DEA code (designation of type) in the EHD specifies individual transmission (b3 = 0) and the DEA does not match the home EA.

③ The DEA code (designation of type) in the EHD specifies broadcasting (b3 = 1) and the DEA specifies subnet-specific intra-subnet simultaneous broadcasting for a subnet other than the home subnet.

④ The DEA code (designation of type) in the EHD specifies broadcasting (b3 = 1), the DEA broadcasting type code specifies intra-domain group broadcasting and the broadcasting target code specifies a group broadcast number that the home node does not have.If the following applies, the

received message processing shall be handed to the routing processing and object processing parts:

⑤ The DEA code (designation of type) in the EHD specifies broadcasting (b3 = 1), the DEA broadcasting type code specifies intra-domain broadcasting, and the broadcasting target code points to a node group that includes the home NodeID.

⑥ The DEA code (designation of type) in the EHD specifies broadcasting (b3 = 1), the DEA broadcasting type code specifies intra-subnet broadcasting, and the broadcasting target code points to a node group that includes the home NodeID.If one of the cases listed below applies, the processing shall be handed only to the object processing part.

⑦ The DEA code (designation of type) in the EHD specifies individual transmission (b3 = 0) and the DEA matches the home EA.

⑧ The DEA code (designation of type) in the EHD specifies broadcasting (b3 = 1), the DEA broadcasting type code specifies intra-subnet broadcasting, and the broadcasting target code points to a Node group that includes the home NodeID.

⑨ The DEA code (designation of type) in the EHD specifies broadcasting (b3 = 1), the DEA broadcasting type code specifies intra-home subnet group broadcasting and the broadcasting target code specifies a group broadcast number that the home node has.

If none of the cases listed above applies, the received message shall be discarded and the processing shall be terminated.

6.3 Routing Processing Specifications

The routing processing specifications are divided into specifications for ECHONET routers and specifications for non-ECHONET router devices. The routing processing uses data held as properties of the "node profile class" and "router profile class" in the profile object.

6.3.1 Routing Processing Specifications for non-ECHONET Router Devices

There are two types of routing processing specifications for non-ECHONET router devices: the simple type and the advanced type. Either type of specifications may be implemented. The processing to be performed differs between those cases in which the simple type specifications are implemented and those in which the advanced type specifications are implemented.

Simple type routing processing specifications: All messages to other subnets are handed to the default router. Specifically, the default router NodeID data is specified as the data to specify the destination location within the subnet, the message to be sent is handed together with the processing to the protocol difference absorption processing section through the common lower-layer communication interface, and the processing at the ECHONET communication processing section is terminated.

Advanced type routing processing specifications: Only the messages that have been identified as deliverable messages shall be transmitted. The NodeID data of the appropriate router is specified, the processing is handed to the protocol difference absorption processing section through the common lower-layer communication interface, and the processing at the ECHONET communication processing

section is terminated. "The appropriate router" shall mean the router located on the transmission path as determined based on the transmission path identified using the data on all routers. The messages that have been identified as undeliverable messages shall be discarded instead of being transmitted. In this case, the processing at the ECHONET communication processing section shall be terminated after the message is discarded. Implementation of the advanced type specifications requires that the data on all routers be acquired from the router in advance.

6.3.2 Routing Processing Specifications for ECHONET Routers

The routing processing specifications for ECHONET routers are as specified in Section 5.7.

6.4 Object Processing Specifications

In the ECHONET Communications Processing Block, device functions are expressed as objects, and through these objects operations are performed between nodes. See Chapter 9 and the Appendix for detailed information on objects.

Object processing can be divided into the three main categories shown below on the basis of conditions required for startup:

Data (reference/control content) is received from basic API processing, and the stipulated object property is controlled.

Received message data is received from received message determination processing, and the stipulated object property is controlled.

The action specified in the object property is managed, and the stipulated object property is controlled based on elapsed time, etc.

(1) through (3) above are designated as object processing (1)–(3), and processing specifications for each are provided below.

6.4.1 Object Processing (1)

Processing using operation data (reference/control content) from the basic API processing can be divided into two main categories: current device object[1] processing and other device object[2] processing. Object processing (1) uses data for all objects. When data is received from the basic API, the block first determines which type of object the data concerns and then performs the appropriate processing. Processing specifications for the two categories are presented below.

Notes:

1) Objects corresponding to functions that are actually present on the self-node. Includes communications definition objects, profile objects, and device objects. Can be referenced and controlled from other nodes.

2) Objects corresponding to functions not present in the self-node and designed to control the status of other nodes. Includes communications definition objects, profile objects, and device objects.

① Current device object processing specifications

When the data (reference/control content) is received from the basic API processing and the stipulated object and property exist, processing is performed in accordance with the request stipulated in the basic API processing.

② Other device object processing specifications

The data (reference/control content) is received from the basic API processing, the stipulated object and property data and the intended recipient EA data are handed off to send message assembly/management processing, and processing is terminated.

6.4.2 Object Processing (2)

Processing based on received message data from received message judgment processing can also be divided into two categories: current device object[1] management and other device object[2] management. In object processing (2), which controls the stipulated object property, when received message data is received from received message judgment processing, the block first decides which type of object the data concerns and then performs the appropriate processing. Processing specifications for the two categories are presented below.

Notes:

1) Objects corresponding to functions that are actually present on the self-node. Includes communications definition objects, profile objects, and device objects. Can be referenced and controlled from other nodes.

2) Objects corresponding to functions not present in the self-node and designed to control and manage the status of other nodes. Includes communications definition objects, profile objects, and device objects. Cannot be accessed or controlled (i.e., cannot be seen) from other nodes.

① Current object management processing specifications

Received message data is received from received message judgment processing, and processing is performed in accordance with the request stipulated in the ECHONET service (ESV).

② Other object management processing specifications

When received message data is received from received message judgment processing, and when the stipulated ESV indicates a "request," the received message is discarded and processing is terminated. If the stipulated ESV indicates a "response" or a "notification," processing is performed in accordance with the ESV.

6.4.3 Object Processing (3)

Periodic notification to the communications definition object is specified, and the data necessary for periodic notification of the object's stipulated property value is handed off to send message assembly and processing. As long as there are objects for which periodic notification is specified, processing, including time count processing, will continue.

Current object status is also monitored, and when a change is detected the data necessary to create a notifying API is handed off to basic API processing (startup processing completion notification, etc.).

6.5 Basic API Processing

Provides application software with basic APIs. Basic APIs enable reception of control (read/write) request data and settings from application software, and the data is then handed off to object processing. Conversely, data is received from objects to be notified to application software, and the application software is notified using a format specified in the basic API.

When content received from the application software is stipulated for initial processing, processing is handed off to startup processing.

6.6 Send Message Creation/Management Processing

When the data necessary to create an ECHONET message is received from startup processing or object processing, the data required for an ECHONET message, such as self-EA, ECHONET header (EHD), and ECHONET byte counter (EBC), is added to create the message, and processing is then handed off to routing processing.

6.7 Startup Processing

When processing begins, the Protocol Difference Absorption Processing Block and connected lower-layer transmission media data required for setting the profile object are received via the common lower-layer communication interface and set to the given property of the object.

When internal processing is completed, the startup sequence processing specified in Chapter 5 is performed, and the message data to be transmitted is handed off to send message assembly/management processing. The system then waits for the required data to be written to the object in line with the sequence and, if necessary, performs time-out management and sends the next message to complete startup processing.

When startup processing is completed, the object property value indicating the status of the Communications Middleware is set, and processing is terminated.

6.7.1 Node Startup Processing

Figure 6.2 shows the internal processing sequence performed when the ECHONET Communications Processing Block is started from the basic API at node startup to start lower-layer transmission media from the ECHONET Communications Processing Block via the common lower-layer communication interface. The sequence shown in Fig. 6.2 is an example sequence for the case where all of the warm start, cold start (1), cold start (2) and cold start (3) functions are implemented.

ECHONET nodes are not required to have the warm start function, but are required to have at least one of the cold start (1), cold start (2) and cold start (3) functions

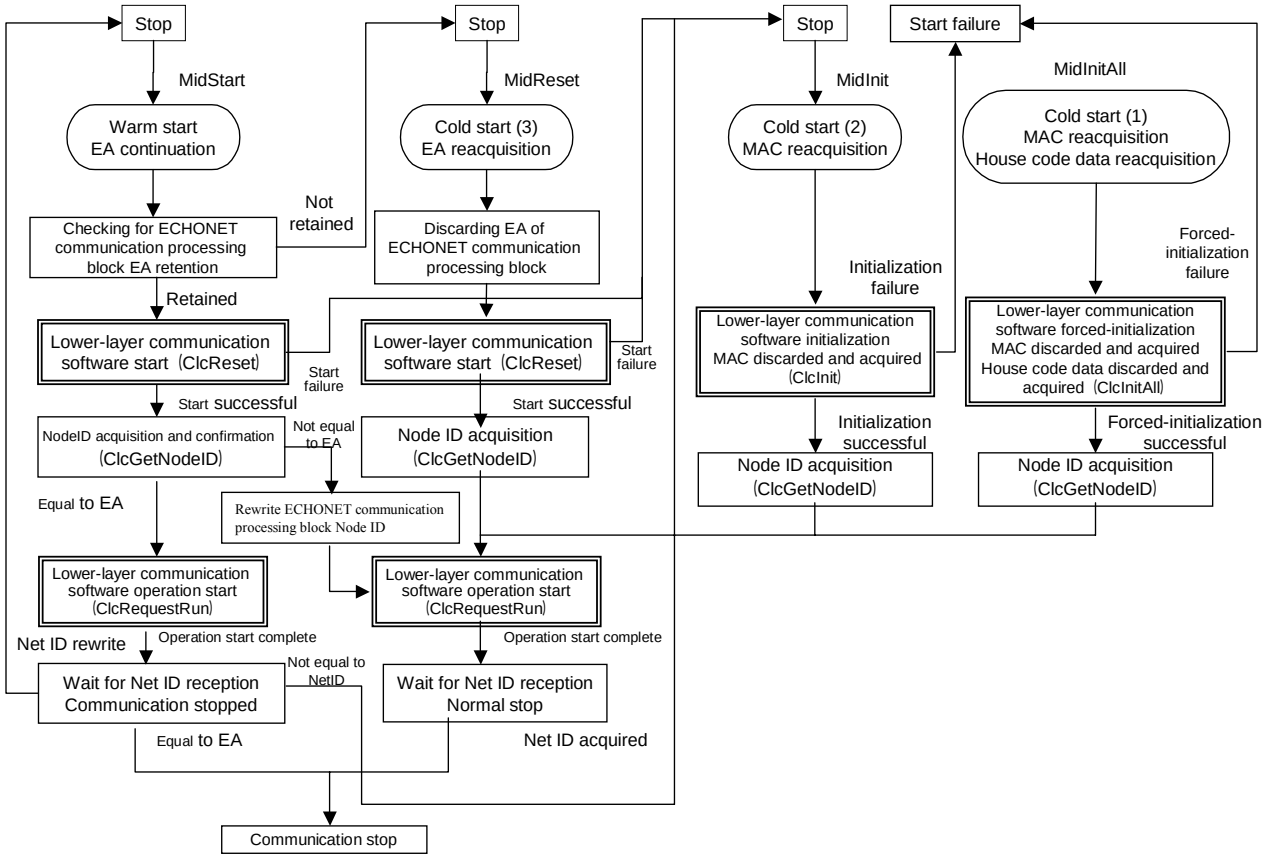


Fig. 6.2 ECHONET Communications Processing Block Internal Processing Sequence for Node Startup

6.8 Description of Processing Functions

Table 6-1 shows a list of the functions processed in the ECHONET Communications Processing Block, together with the implementation status. The "implementation status" column indicates whether or not the given function is required, with N specifying normal (non-router) nodes and R, routers. The function numbers shown in the first column are used as symbols when presenting the processing functions of the ECHONET Communications Processing Block. (For example, "M1a2bcde" would indicate six functions shown in Table 6-1: M1a, M2b, M2c, M2d, and M2e.)

Table 6-1 List of ECHONET Communications Processing Block Functions (1/4)

Function No.		Functions (Overview)	Implementation Status	Remarks
M1	a	Detection of nodes with improper NetID processing Processing functions in Section 5.5.2	Required (N+R)	
M2	a	Processing of basic sequence for object control in general Processing functions in Section 5.2	Required (N+R)	
	b	Set processing Processing functions in Section 4.2.8 (1). Returns "response."	Required (N+R)	Required because node profile class must be implemented. Differs from services that must be processed for each property (i.e., not required for all properties).
	c	Get processing Processing functions in Section 4.2.8 (2). Returns process response.		
	d	Property value notification processing Processing functions in Section 4.2.8 (3). Returns "response" and sends "autonomous notification."		
	e	SetM processing Processing functions in Section 4.2.8 (4). Returns "response."		
	f	GetM processing Processing functions in Section 4.2.8 (5). Returns "response."		
	g	Array element notification processing Processing functions in Section 4.2.8 (6). Returns "response" and sends "autonomous notification."		
	h	AddM processing Processing functions in Section 4.2.8 (7). Returns "response."		

Table 6-1 List of ECHONET Communications Processing Block Functions (2/4)

Function No.		Functions (Overview)	Implementation Status	Remarks
M2	i	DelM processing Processing functions in Section 4.2.8 (8). Returns "response."		
	j	CheckM processing Processing functions in Section 4.2.8 (9). Returns "response."		
	k	AddMS processing Processing functions in Section 4.2.8 (9). Returns "response."		
	l	Communications definition object management processing (1) Processes communications definition object of status change notification requirement indicated in Section 9.13.		
	m	Communications definition object management processing (2) Processes communications definition object of periodic communications requirement indicated in Section 9.13.		
	n	Communications definition object management processing (3) Processes communications definition object of set control reception method requirement indicated in Section 9.14.		
	o	Communications definition object management processing (4) Processes communications definition object of action setting indicated in Section 9.15.		
	p	Communications definition object management processing (5) Processes communications definition object of trigger setting indicated in Section 9.16.		
	A	Get processing expansion When a Get "request" is received, returns object property value held in Communications Middleware.		
	B	GetM processing expansion When a Get "request" is received, returns object property value held in Communications Middleware.		
	C	Property value notification processing expansion When a property value notification "request" is received, returns object property value held in Communications Middleware.		
	D	Array element notification processing expansion When an array element notification "request" is received, returns object property value held in Communications Middleware using Communications Middleware.		

Table 6-1 List of ECHONET Communications Processing Block Functions (3/4)

Function No.		Functions (Overview)	Implementation Status	Remarks
MO	E	Other device object status management processing (1) When a "request" to read a property held as other device objects is received, the property value of the other device object held in Communications Middleware is changed to the notified value.		
	F	Other device object status management processing (2) When a status notification for a property held as other device objects is received, the property value of the other device object held in Communications Middleware is changed to the notified value.		
	G	Other device object status management processing (3) When a status announcement for or a "request" to read a property not held as other device objects is received, the received message is discarded.		
	H	Other device object status management processing (4) When a status announcement for or a "request" to read a property not held as other device objects is received, the received message is not discarded, and the application is notified.		
	I	Current device object management processing (1) "Requests" for properties not held as current device objects are not discarded, and the application is notified.		
	J	Current device object management processing (2) When a "request" for properties held as current device objects is received, a receipt response is returned, and the application is notified of the "request."		

Table 6-1 List of ECHONET Communications Processing Block Functions (4/4)

Function No.		Functions (Overview)	Implementation Status	Remarks
M3	a	API processing (1) Processing indicated in Section 6.5; required API processing indicated in Level 1 of Part IV specifications	Required (N+R)	
	b	API processing (2) Processing indicated in Section 6.5; required API processing indicated in Level 1 of Part IV specifications		
	c	API processing (3) Processing indicated in Section 6.5; required API processing indicated in Level 2 of Part IV specifications	Required (N+R)	
	d	API processing (4) Processing indicated in Section 6.5; required API processing indicated in Level 2 of Part IV specifications		
M4	a	NetID server Parent router processing indicated in Sections 5.4.2 and 5.4.3 (allocation of NetIDs to normal routers)	Required (R)	Required only for parent router.
	b	Routing processing Routing processing indicated in Section 6.3	Required (R)	
	c	Simple routing message processing "Simple" routing processing for non-router nodes indicated in Section 6.3.2	Required (N)	Not required when node has M3d function.
	d	Advanced routing message processing "Advanced" routing processing for non-router nodes indicated in Section 6.3.2		
M5	a	Send message creation/management processing Processing indicated in Section 6.6	Required (N+R)	
M6	a	Detection of nodes with improper NetID processing Processing indicated in Section 5.5.2	Required (N+R)	
	b	Basic sequence for node cold start processing: non-router side Non-router-side processing indicated in Section 5.3.1	Required (N+R)	When using the NetID of the range open to users, only instance change class notification is required (N+R).
	c	Basic sequence for node cold start processing: router side Router-side processing indicated in Sections 5.3.1 and 5.3.2	Required (R)	
	d	Basic sequence for node warm start processing: non-router side Non-router-side processing indicated in Section 5.3.2		
	e	Non-automatic NetID acquisition Setting of NetIDs for code range open to users, as indicated in Section 2.3		

Chapter 7 Protocol Difference Absorption Processing Block Processing Specifications

7.1 Basic Concept

This section describes the processing specifications that are to be specified by the ECHONET Communications Processing Block in the Protocol Difference Absorption Processing Block as shown in the figure below. Note that the processes shown in the figure are used simply to describe basic processing in the ECHONET Communications Processing Block and are not intended as specifications for actual software structure.

(1) Message receipt/assembly processing
(2) Message splitting/transmission processing
(3) Address conversion processing
(4) Conversion by communications type processing
(5) Lower-layer common lower-layer communication interface processing

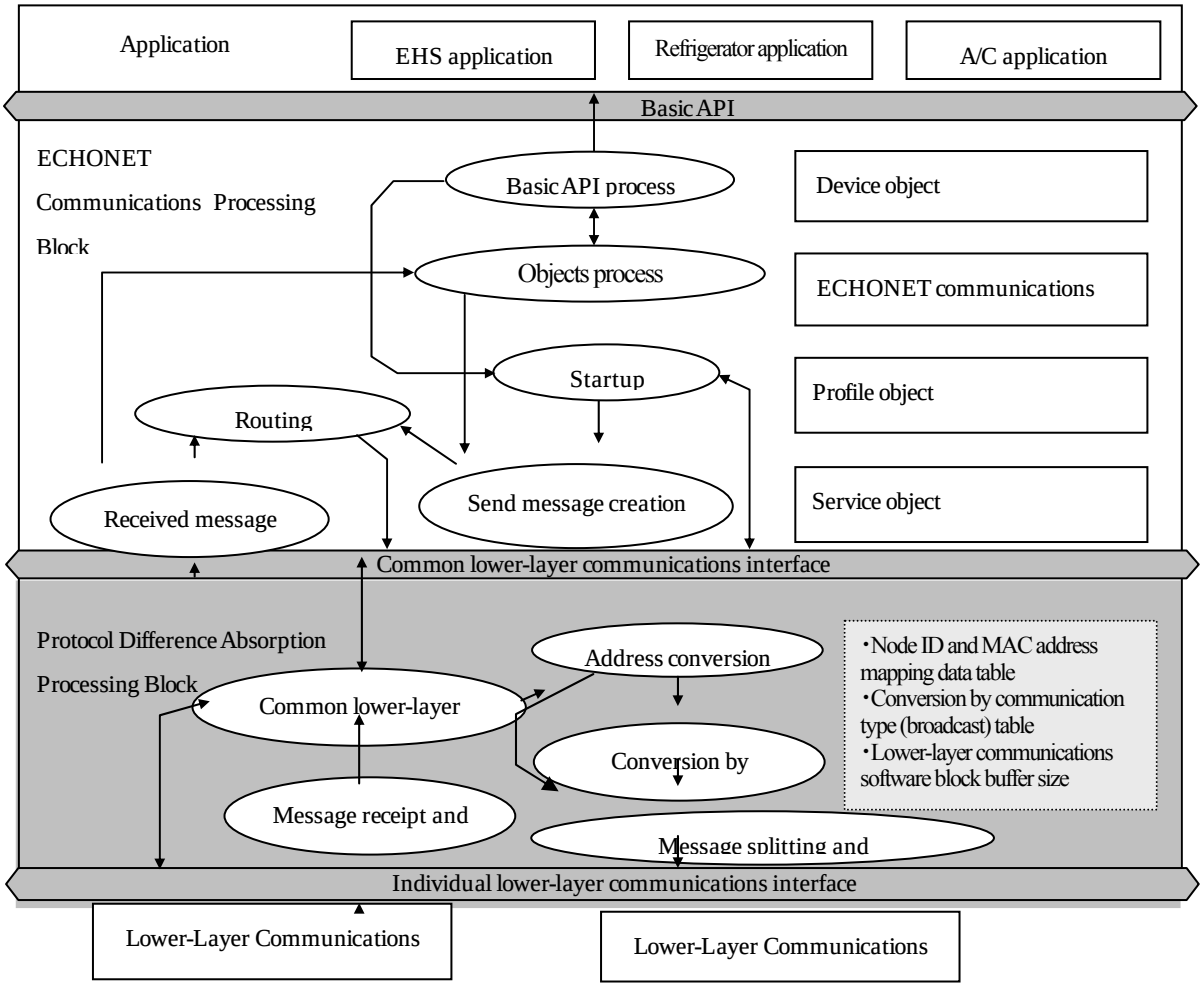


Fig. 7.1 Overview of Communication Middleware Block Processing (Layer Configuration Overview)

7.2 Message Receipt/Assembly Processing

A message is received from the Lower-Layer Communications Software Block via an individual lower-layer communications interface. Depending on the message header (EDC) in the Protocol Difference Absorption Processing Block, two types of processing are possible. Both are specified below.

(1) When received message is a complete (not split) message
(2) When received message is a split message

7.2.1 Message Receipt/Assembly Processing (1)

When the received message is complete (not split), message data (ESDATA) in the Protocol Difference Absorption Processing Block is handed off to common lower-layer communication interface processing as data to be forwarded to the ECHONET Communications Processing Block, and processing is terminated.

7.2.2 Message Receipt/Assembly Processing (2)

When the received message is a split message, message assembly processing is performed in the ECHONET Communications Processing Block using the received message, the MAC address of the source, the message identification stipulator within the EDC, and the split message number, all of which were received from the lower-layer communications software. The received message is held until the message is properly assembled. Once assembly is complete, the message is handed off to common lower-layer communication interface processing as data to be forwarded to the ECHONET Communications Processing Block, and processing is terminated.

Such items as wait-time for the next message (required for assembly) and the number of messages that can be processed simultaneously are not specified. Also, split message receiving functions are not required.

MAC address data passed on from the lower-layer communications software as an individual lower-layer communications interface is used only for message assembly and does not require address conversion processing.

7.3 Message Splitting/Transmission Processing

7.3.1 Message Splitting/Transmission Processing (1)

When the send message length is smaller than the transmission buffer size (splitting not required), an EDC stipulating no splitting is created, the send message data and MAC address data for the intended recipient are handed off to the Lower-Layer Communications Software Block via the individual lower-layer communications interface, and processing is terminated.

7.3.2 Message Splitting/Transmission Processing (2)

When the send message length is larger than the transmission buffer size (splitting required), the message data is split into pieces of an appropriate size smaller than the transmission buffer size. These pieces, designated ESDATA(1)–ESDATA(n), are then handed off in order to the Lower-Layer Communications Software Block via the individual lower-layer communications interface, together with an EDC for each (EDC[1]–EDC[n]) and the MAC address of the intended recipient, starting with the first message. Once all messages have been handed off, processing is terminated.

The actual size and number of split messages, and the decision on whether or not to incorporate splitting functions, are implementation issues and therefore are not specified.

7.4 Address Conversion Processing

Two types of processing are possible depending on the address data (consists of an ECHONET header code and a NodeID code; detailed specifications are provided in Part V *Lower-Layer Common Lower-Layer Communication Interface Specifications*) received together with the send message from common lower-layer communication interface processing:

(1) When the address stipulates broadcast, processing is passed to communications type conversion processing.
(2) When the address stipulates individual, address conversion processing is performed for the specified NodeID and MAC address for each lower-layer communications protocol, the address is designated as the intended recipient address, and processing is passed to message splitting/transmission processing.

NodeID and MAC address conversion processing specifications for each lower-layer communications protocol are described below.

As stated in the address specifications for lower-layer communication software in Part 3, the number of available MAC addresses varies from one lower-layer communications protocol to another. Furthermore, a special MAC address use is defined by some lower-layer communications protocols. To comply with two or more lower-layer communication software programs, the NodeID and MAC address must be selected while giving due consideration to the aforementioned matter.

7.4.1 Address Conversion Specifications for Power Line Communications Protocol

The MAC address is 2 bytes long, and the value of the lower byte is the same as in the NodeID.

No.	Object	MAC address (HEX)	
1	Plug-and-play manager address	40	00
2	Individual address	40	01-EF
3	General broadcast address	40	F0
4	Reserved for future use	40	F1-FE
5	Plug-and-play reserved	40	FF

7.4.2 Address Conversion Specifications for Low-power Wireless Protocol

Since NodeID = MAC address, conversion is not required.

7.4.3 Address Conversion Specifications for Extended HBS Protocol

Since NodeID = MAC address, conversion is not required.

7.4.4 Address Conversion Specifications for IrDA Control Protocol

IrDA Control conversion processing differs for host and peripheral.

① Host

Since the peripheral NodeID is a virtual MAC address (See Part III, Chapter 6 for the NodeIDs of peripherals managed by lower-layer communications software), conversion is not required.

② Peripheral

The NodeID of the intended recipient is converted to the MAC address of the host, and the message is sent to the host. (Message is sent from host to intended recipient peripheral.)

7.4.5 Address Conversion Specifications for LonTalk® Protocol

Each node uses the Neuron® chip NodeID (7-bit data) as its own MAC address. Therefore, the converted 8-bit data value, with the MSB set to "0", is used as the NodeID.

7.4.6 Address Conversion Requirements for IP/Bluetooth Protocol

No conversion is required because the NodeID value is identical to the MAC address value.

7.4.7 Address Conversion Requirements for IP/Ethernet/IEEE802.3 Protocol

No conversion is required because the NodeID value is identical to the MAC address value.

7.4.8 Address Conversion Requirements for IEEE802.11/11b Protocol

No conversion is required because the NodeID value is identical to the MAC address value

7.5 Communications Type Conversion Processing

Broadcast addresses are converted based on the broadcast stipulated intended recipient data received from address conversion processing (this consists of the ECHONET header code and the code for b2 of the DEA). Here, one of two types of processing is performed, based on whether or not broadcast is stipulated in the lower-layer communications protocol:

(1) When broadcast is stipulated in the lower-layer communications protocol
The intended recipient requirement data is converted in accordance with the lower-layer communications protocol broadcast requirement specifications. The broadcast requirement data, intended recipient address, and the send message are handed off to message splitting/transmission processing, and processing is terminated.

(2) When broadcast is not stipulated in the lower-layer communications protocol
All transmission recipient MAC addresses are extracted, and the MAC address data (extracted in order) and the send message are handed off to message splitting/transmission processing until transmission of the stipulated message to all MAC addresses has been completed. When the requests intended for all MAC addresses have been handed off to message splitting/transmission processing, processing is terminated.

Specifications for establishing broadcast address data using b2 of the DEA are described below for each lower-layer communications protocol.

7.5.1 Communications Type Conversion Specifications for Power Line Communications Protocol

If broadcasting is specified, it shall be deemed that the second byte of the DEA is 0xF0 and a notification for simultaneous broadcasting shall be made.

7.5.2 Communications Type Conversion Specifications for Low-power Wireless Protocol

Since the code for b2 of the DEA is the same as the MAC address for broadcast, conversion of the broadcast address is not required. However, in addition to the address data, the broadcast requirement data must be notified to the lower-layer communications software. The broadcast requirement data, the broadcast recipient addresses, and the send message are handed off to message splitting/transmission processing, and processing is terminated.

7.5.3 Communications Type Conversion Specifications for Extended HBS Protocol

Since the code for b2 of the DEA is the same as the MAC address for broadcast, conversion of the broadcast address is not required. However, in addition to the address data, the broadcast requirement data must be notified to the lower-layer communications software. The broadcast requirement data, the broadcast recipient addresses, and the send message are handed off to message splitting/transmission processing, and processing is terminated.

7.5.4 Communications Type Conversion Specifications for IrDA Control Protocol

IrDA Control conversion processing differs for host and peripheral.

① Host

Since b2 of the peripheral DEA is a virtual MAC address (See Part III, Chapter 6 for the NodeIDs of peripherals managed by lower-layer communications software), conversion is not required.

② Peripheral

Here, b2 of the intended recipient DEA is converted to the MAC address of the host, and the message is sent to the host. (Message is sent from host to intended recipient peripheral.)

7.5.5 Communications Type Conversion Specifications for LonTalk® Protocol

The broadcast requirement data, broadcast recipient addresses, and the send message are handed off to message splitting/transmission processing, and processing is terminated. Conversion to broadcast address is performed by the lower-layer communications software. Detailed specifications are provided in Part III, Section 6.4.2. When the current subnet is not included in the broadcast list, the router address (b2 of DEA = MAC address) is specified as the intended recipient address. In all other cases, the intended recipient address is set to NULL. The broadcast requirement data is notified to the lower-layer communication software.

7.5.6 Communication Type Conversion Requirements for the IP/Bluetooth Protocol

The message to be sent is handed together with the broadcast specification data and broadcast destination address to the "message splitting and transmission processing" part and the processing shall be terminated. The conversion into the broadcast address shall be done by the lower-layer communication software.

7.5.7 Communication Type Conversion Requirements for IP/Ethernet/IEEE802.3 Protocol

The message to be sent is handed together with the broadcast specification data and broadcast destination address to the "message splitting and transmission processing" part and the processing shall be terminated. The conversion into the broadcast address shall be done by the lower-layer communication software.

7.5.8 Communications Type Conversion Requirements for IEEE802.11/11b Protocol

The message to be sent is handed, together with the broadcast specification data and broadcast recipient addresses, to the "message splitting and transmission processing" part and the processing shall be terminated. Conversion to broadcast addresses shall be performed by the lower-layer communication software.

7.6 Common Lower-Layer Communications Interface Processing

Provides common lower-layer communications interface to the ECHONET Communications Processing Block. Settings and control request data (send messages, etc.) are received from the ECHONET Communications Processing Block via the common lower-layer communications interface. If the data consists of send message data, it is handed off to address conversion processing; if it consists of lower-layer communications block settings or data request data, it is handed off to the Lower-Layer Communications Software Block via the individual lower-layer communications interface.

By contrast, when received message data is received from message receipt/assembly processing, or when settings/data response data is received from the Lower-Layer Communications Software Block via the individual lower-layer communications interface, it is notified to the ECHONET Communications Processing Block in the format specified in the common lower-layer communication interface.

7.7 Description of Processing Functions

Table 7-1 lists the functions processed in the Protocol Difference Absorption Processing Block and indicates their implementation status. The function numbers shown in Table 7-1 are used as the symbols for presenting Protocol Difference Absorption Processing Block processing functions.

Table 7-1 List of Protocol Difference Absorption Processing Block Functions (1/2)

Function No.		Functions (Overview)	Implementation Status	Remarks
C1	a	Message assembly processing Message transmission processing indicated in Sections 7.2, 4.2, and 4.2.10	Required	
C2	a	Message splitting processing Message transmission processing indicated in Sections 7.3, 4.2, and 4.2.10	Required	
C3	a	Address conversion processing for power line communications protocol Processing indicated in Section 7.4.1	Required *	*Those relating to non-implemented lower-layer communications software protocols need not be implemented.
	b	Address conversion processing for low-power wireless protocol Processing indicated in Section 7.4.2		
	c	Address conversion processing for extended HBS Processing indicated in Section 7.4.3		
	d	Address conversion processing for IrDA Control protocol Processing indicated in Section 7.4.4		
	e	Address conversion processing for LonTalk® protocol Processing indicated in Section 7.4.5		
	f	Address conversion processing function for the IP/Bluetooth® protocol The processing function described in Section 7.4.6		
	g	Address conversion for the IP/Ethernet/IEEE802.3 protocol The processing function described in Section 7.4.7		
	h	Address conversion for the IEEE802.11/11b protocol The processing function described in Section 7.4.8		
C4	a	Communications type conversion processing for power line communications protocol Processing indicated in Section 7.5.1	Required *	*Those relating to non-implemented lower-layer communications software protocols need not be incorporated.
	B	Communications type conversion processing for low-power wireless protocol Processing indicated in Section 7.5.2		
	c	Communications type conversion processing for extended HBS Processing indicated in Section 7.5.3		
	d	Communications type conversion processing for IrDA Control protocol Processing indicated in Section 7.5.4		
	e	Communications type conversion processing for LonTalk® protocol Processing indicated in Section 7.5.5		
	f	Communications type conversion processing function for the IP/Bluetooth® protocol The processing function described in Section 7.5.6		

	g	Communications type conversion processing function for the IP/Ethernet/IEEE802.3 protocol		
		The processing function described in Section 7.5.7		
	h	Communications type conversion processing function for the IEEE802.11/11b protocol		
		The processing function described in Section 7.5.8		

Table 7-1 List of Protocol Difference Absorption Processing Block Functions (2/2)

Function No.		Functions (Overview)	Implementation Status	Remarks
C5	A	Common Lower-Layer Communication Interface processing (1)	Required	
		Processing indicated in Section 7.6; required API processing indicated in Level 1 of Part V specifications		
	B	Common Lower-Layer Communication Interface processing (2)		
		Processing indicated in Section 7.6; optional API processing indicated in Level 1 of Part V specifications		
	C	Common Lower-Layer Communication Interface processing (3)	Required	
		Processing indicated in Section 7.6; required API processing indicated in Level 2 of Part V specifications		
	d	Common Lower-Layer Communication Interface processing (4)		
		Processing indicated in Section 7.6; optional API processing indicated in Level 2 of Part V specifications		

Chapter 8 ECHONET Communication Middleware State Transitions

8.1 Basic Concept

This section specifies ECHONET Communication Middleware state transitions. The specifications stated in this chapter enable the application software to determine the operating status of communication middleware. The common lower-layer communication interface divides the ECHONET Communication Middleware into two layers: the ECHONET Communications Processing Block and the Protocol Difference Absorption Processing Block. This chapter presumes that the Protocol Difference Absorption Processing Block operates in synchronism with the lower-layer communication software, and stipulates the state transitions of the ECHONET Communications Processing Block only.

8.2 State Transitions in ECHONET Communications Processing Block

Figure 8.1 summarizes the state transitions of the ECHONET Communications Processing Block. Shaded events (MidInitAll, MidInit, etc.) in the figure indicate requests from application software. The term "Error detection" in the figure refers to the detection of an error in the ECHONET Communications Processing Block. The ECHONET Communications Processing Block remains in the normal operation state even when an upper-layer error (application software error) or lower-layer error (Protocol Difference Absorption Processing Block or lower-layer communication software error) is detected. Table 8.1 outlines the various states.

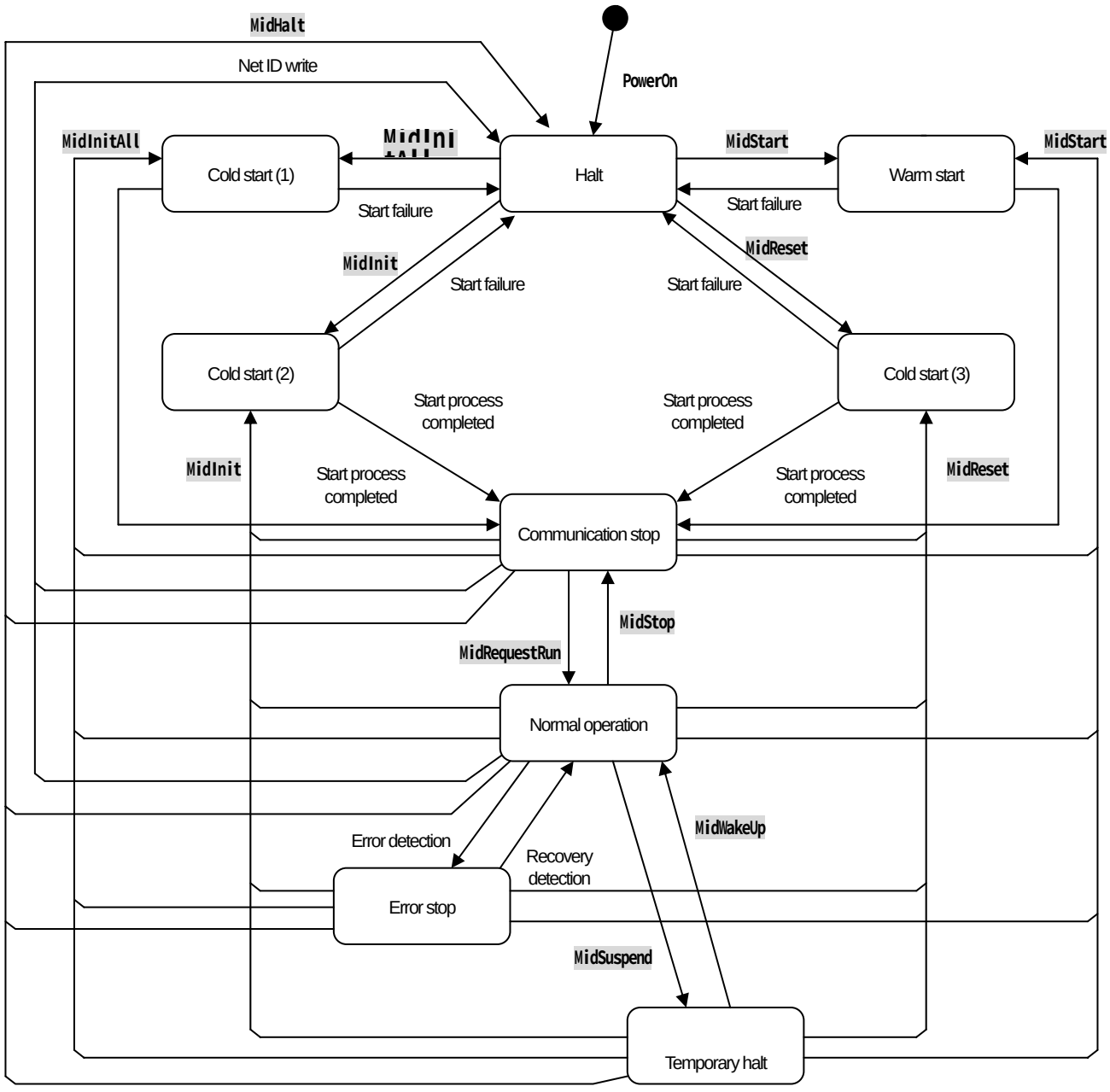


Fig. 8.1 ECHONET Communications Processing Block State Transition Diagram

Table 8.1 ECHONET Communications Processing Block State Overview (1/2)

State name	Sequence of execution in ECHONET Communications Processing Block	Instructions to lower layer
Stop	-State prevailing after power ON. -Waits for the instruction for initiating a cold start (1), cold start (2), cold start (3), or warm start process.	
Cold start (1)	-Initializes various parameters within the middleware. -State prevailing during a start process performed with the house code data, MAC address, and NetID discarded. -MidInitAll invokes a status change from the stop state. -Instructs the lower layer to discard and update the house code data and MAC address. -Requests that the lower layer furnish a NodeID when the lower-layer house code data and MAC address are successfully discarded and updated. -Requests that the lower layer start communication and then searches the default router within the subnet to acquire a NetID. -Sets NetID to 0x00 if the default router is not found. -Switches to the communication stop state when a series of processes ends normally. -Switches to the stop state if the processes are not successfully completed.	-Requests that the lower layer start by discarding and updating the house code data and MAC address. (ClcInitAll) -Requests a NodeID. (ClcGetNodeID) -Requests the start of communication. (ClcRequestRun) -Searches the default router. -Issues the instruction for a NetID read into the default router.
Cold start (2)	-Initializes various parameters within the middleware. -State prevailing during a start process performed with the MAC address and NetID discarded. -MidInit invokes a status change from the stop state. -Instructs the lower layer to discard and update the MAC address. -Requests that the lower layer furnish a NodeID when the lower-layer MAC address is successfully discarded and updated. -Requests that the lower layer start communication and then searches the default router within the subnet to acquire a NetID. -Sets NetID to 0x00 if the default router is not found. -Switches to the communication stop state when a series of processes ends normally. -Switches to the stop state if the processes are not successfully completed.	-Requests that the lower layer start by discarding and updating the MAC address. (ClcInit) -Requests a NodeID. (ClcGetNodeID) -Requests the start of communication. (ClcRequestRun) -Searches the default router. -Issues the instruction for a NetID read into the default router.
Cold start (3)	-Initializes various parameters within the middleware. -State prevailing during a start process performed with the NodeID and NetID discarded. -MidReset invokes a status change from the stop state. -Discards the NodeID and NetID retained by the ECHONET Communications Processing Block, then requests and acquires a NodeID based on the MAC address currently retained by the lower layer. Switches to the stop state if the lower layer does not retain a MAC address. -Requests that the lower layer start communication and searches the default router within the subnet to acquire a NetID when the NodeID is successfully acquired. -Sets NetID to 0x00 if the default router is not found. -Switches to the communication stop state when a series of processes ends normally. -Switches to the stop state if the processes are not successfully completed.	-Requests that the lower layer start while retaining the MAC address. (ClcReset) -Requests a NodeID. (ClcGetNodeID) -Requests the start of communication. (ClcRequestRun) -Searches the default router. -Issues the instruction for a NetID read into the default router.

Table 8.1 ECHONET Communications Processing Block State Overview (2/2)

State name	Sequence of execution in ECHONET Communications Processing Block	Instructions to lower layer
Warm start	- Initializes various parameters within the middleware. -State prevailing during a start process performed with the NodeID and NetID retained. -MidStart invokes a status change from the stop state. -Requests and acquires the NodeID currently possessed by the lower layer. Switches to the stop state if the lower layer does not retain the NodeID. -Compares the new NodeID acquired from the lower layer against the current NodeID. Switches to the stop state if they do not match. -Requests that the lower layer start communication and searches the default router within the subnet to acquire a NetID when the NodeIDs match. -Switches to the stop state if the newly acquired NetID does not match the currently possessed NetID. -Uses the NetID possessed by the ECHONET Communications Processing Block as a new NetID if the default router is not found. -Switches to the communication stop state when processing ends normally.	-Requests that the lower layer start while retaining the MAC address. (ClcReset) -Requests a NodeID. (ClcGetNodeID) -Requests the start of communication. (ClcRequestRun) -Searches the default router. -Issues the instruction for a NetID read into the default router.
Communication stop	-Standby state ready for communication with the ECHONET address determined. -MidRequestRun invokes a status change to the communication operation state. -Does not accept an ECHONET communication or object operation process request from application software. -Switches to the stop state when the possessed NetID is rewritten.	
Normal operation	-State in which an ECHONET communication or object operation process can be performed in compliance with a request from application software. -Switches to the stop state when the possessed NetID is rewritten.	-Start of operation (ClcRun)
Temporary halt	-State in which no ECHONET communication or object operation process is performed and no ECHONET communication process request is issued to the Protocol Difference Absorption Processing Block.	-Start of operation (ClcWakeUp)
Error stop	-State in which communication is stopped due to an abnormality.	

Chapter 9 ECHONET Objects: Detailed Specifications

9.1 Basic Concept

This section specifies specific values for the class codes of ECHONET objects processed in the ECHONET Communication Middleware, whose types and overview were given in Chapter 4, along with property configurations and their detailed specifications. In the case of class codes, as shown in Chapter 2, rather than providing entirely new specifications, standards already being studied by the industry were applied whenever possible to capitalize on past work. Regarding object properties, the operands (control content) of JEM-1439 were analyzed and used as reference. ECHONET objects described in this section and in the Appendix are, as noted in Chapter 3, divided into three main classes: device objects, profile objects, and communications definition objects. In terms of the code structure, they are divided into the class groups shown below. After presenting the shared ECHONET property specifications and object super classes that form ECHONET objects, this section will provide guidelines for each class group (except for the service group) and details for each class.

(1) Device objects
- Sensor-related device class group
- Air conditioning-related device class group
- Housing-related device class group
- Cooking/housework-related device class group
- Health-related device class group
- Management and control-related device class group
- AV-related device class group

(2) Profile objects
- Profile class group

(3) Communications definition objects
- Sensor-related device communications definition class group
- Air conditioning-related device communications definition class group
- Housing-related device communications definition class group
- Cooking/housework-related device communications definition class group
- Health-related device communications definition class group
- Management and control-related device communications definition class group
- Profile communications definition class group
- AV-related device communications definition class group

Detailed specifications for each device object class are provided in the Appendix (ECHONET Device Objects: Detailed Specifications).

ECHONET nodes shall be equipped with node profile class and at least one of the following:
- Device object

- Non-node profile class profile object
- Service middleware object

Each ECHONET node must implement a device object for at least one representative device.

9.2 ECHONET Properties: Basic Specifications

This section discusses the specifications shared by all ECHONET object classes, the details of which are provided in this section and in the Appendix.

9.2.1 ECHONET Property Value Data Types

The ECHONET property value is expressed as an unsigned integer when the value is a non-negative integer value; it is expressed as a signed integer when the value is an integer value containing negatives.

When the value is a small value, it is handled as a fixed point type; when it is a non-negative small value, it is treated as an unsigned integer; and when it is a small value containing negatives, it is treated as a signed integer. Data types and sizes are specified individually for each property.

Although property data size is specified individually for each property, property value data of 2 bytes or larger comprises ECHONET Communication Middleware data as ECHONET property value data (EDT) beginning from the most significant byte.

9.2.2 ECHONET Property Value Range

The definition range for the ECHONET properties specified in this section and in Part 2, Paragraph 9.2.2, and the treatment of property values when the corresponding actual device property value operating range is not in agreement, are specified below.

(1) When the actual device property value operating range corresponding to the ECHONET property is smaller than the ECHONET property definition range and the actual device property value assumes the upper or lower limit value, the upper or lower limit value of the operating range is considered to be the property value.

Assuming that the ECHONET property definition range is 0x00–0xFD (0°C–253°C) and the corresponding actual device operating range is 0x0A–0x32 (10°C–50°C), when the actual device property value is the upper limit (50°C) of the operating range, the upper limit value 0x32 (50°C) of the actual device operating range is considered as the ECHONET property value, and when the actual device property value is the lower limit value (10°C), the lower limit value 0x0A (10°C) is considered to be the ECHONET property value.

(2) When the actual device property value operating range corresponding to the ECHONET property is larger than the ECHONET property definition range and the actual device property value assumes a value outside the ECHONET property definition range, a code showing an underflow or overflow becomes the property value.

Assuming that the ECHONET property definition range is 0x00–0xFD (0°C–253°C) and the corresponding actual device operating range is –10°C to 300°C, when the actual device property value assumes a value below the ECHONET property definition range, the underflow code 0xFE becomes the property value; when the actual device property value assumes a value above the ECHONET property definition range, the overflow code 0xFF becomes the property value.

Table 9.1 shows the underflow and overflow codes for each data type.

Table 9.1 Data Types, Data Sizes, and Overflow/Underflow Codes

data type	data size	Underflow	Overflow
signed char	1 Byte	0x80	0x7F
Signed short	2 Byte	0x8000	0x7FFF
signed long	4 Byte	0x80000000	0x7FFFFFFF
unsigned char	1 Byte	0xFE	0xFF
unsigned short	2 Byte	0xFFFE	0xFFFF
unsigned long	4 Byte	0xFFFFFFFE	0xFFFFFFFF

(3) On other ECHONET property values, refer to the section 10.4 of part X.

9.2.3 Class-specific Compulsory Properties

The properties defined as the "compulsory" properties for specific classes in the property specifications in this chapter and Appendix shall be implemented as part of the respective classes.

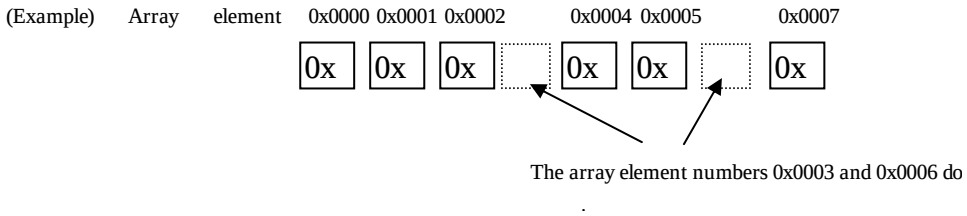
9.2.4 Properties that Must Have a Status Change Announcement Function

Any property may transmit a property value notification service message at any time. However, the implementation of a property defined as a "property that must have a status change announcement function" in the property specifications in this chapter or Appendix requires the incorporation of a function to send a property value notification service message in the form of an intra-domain simultaneous broadcast upon a change in the status (property value) of that property. This announcement is not required for a node startup, as it is not to be considered as a property status change.

A property that is not defined as a "property that must have a status change announcement function" may also transmit a property value notification service message upon a change in the property value of the property. This message does not have to be sent in the form of an intra-domain simultaneous broadcast.

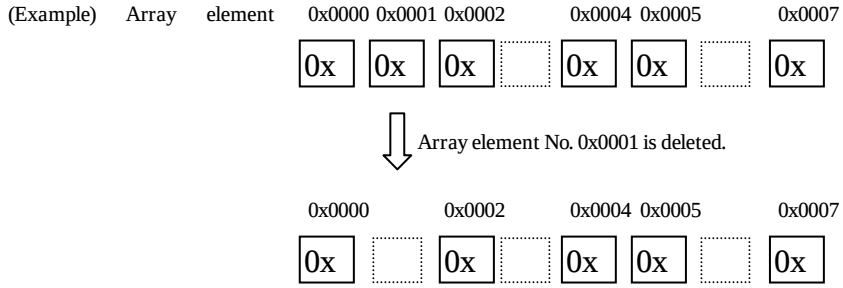
9.2.5 Array

ECHONET properties can be in the form of an array. Array elements are stipulated by an array element number, which ranges from 0x0000 to 0xFFFF. Array elements may be noncontiguous. The data type of each array element must be unique within a property.

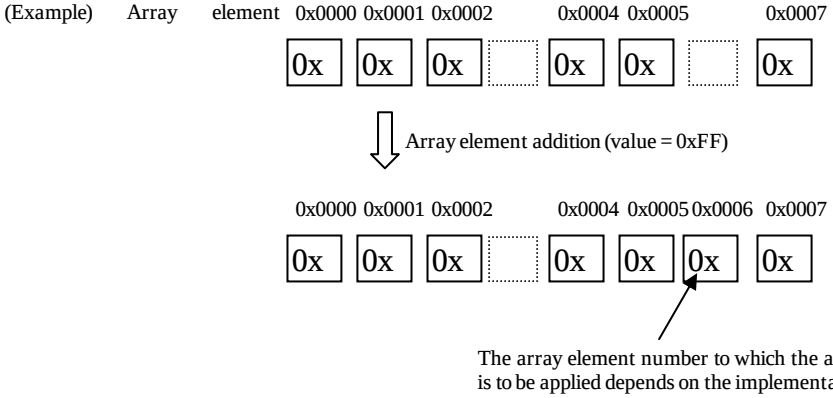


For the property value element-stipulated write service (ESV = 0x64, 0x65), property value element-stipulated read service (ESV = 0x66), property value element-stipulated notification service (ESV = 0x67), and property value element-stipulated deletion service (ESV = 0x6A, 0x6B), the "response not possible" is returned if the array element does not exist. In the case of the property value element-stipulated addition service (ESV = 0x68, 0x69), the "process not possible" response is returned if the array element already exists.

The property value element-stipulated deletion service deletes a specified array element but does not shift subsequent elements forward.

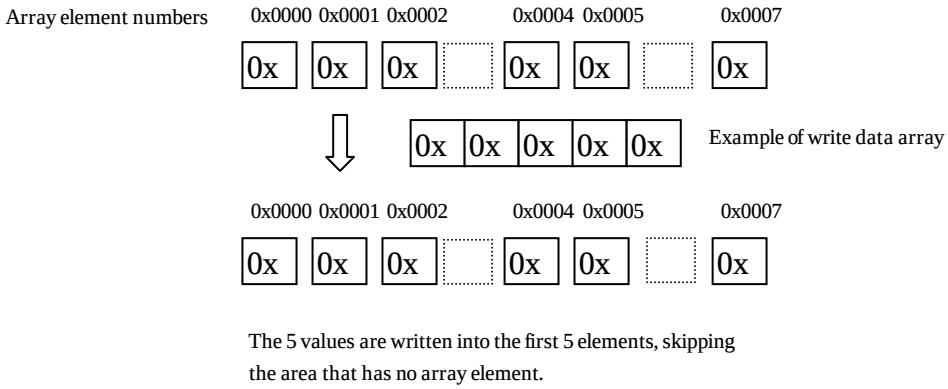


The property value element addition service (ESV = 0x6D, 0x6E) does not specify the array element number to which an element addition is to be applied.



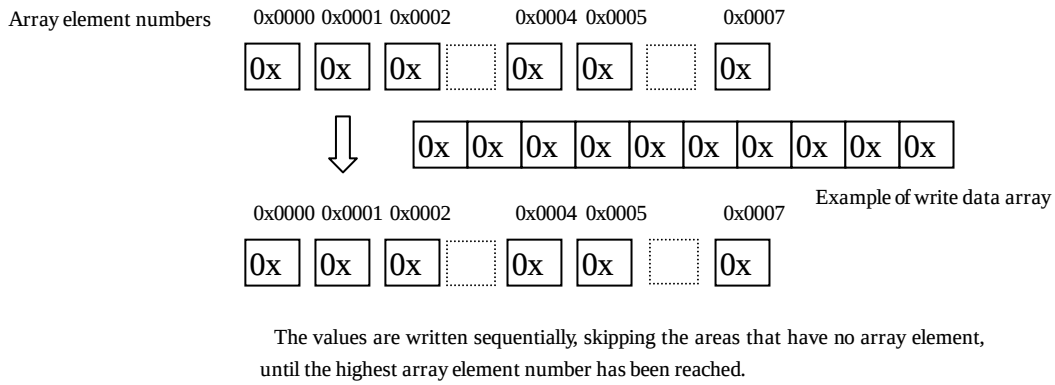
In the case of the property value write service (ESV = 0x60, 0x61), the data is written (in the form of a lump write) into elements of the array property specified by the EPC, with each data piece occupying one element. The data pieces are written sequentially, starting with array element No. 0x0000 until all of them have been written or the highest array element number of the array property has been reached. In the latter case, the excess part of the data is discarded. Data shall not be written into areas that do not have an array element.

(Example) Five values (0x01, 0x02, 0x03, 0x04 and 0x05) are written into an array property whose highest array element number is 0x0007.

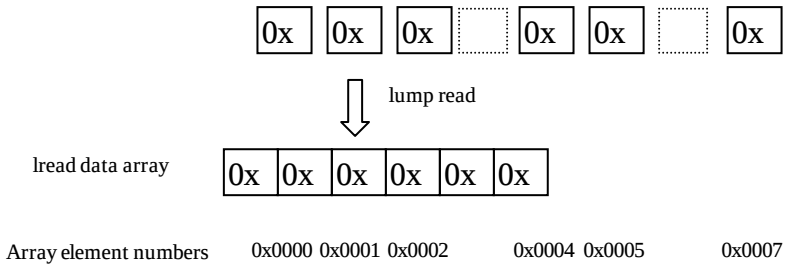


(Example)

Ten values (0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09 and 0x0a) are written into an array property whose highest array element number is 0x0007.



In the case of the property value read service (ESV = 0x62), elements of the array property specified by the EPC are read sequentially (a lump read), starting with array element No. 0x0000 until all values have been read, the highest array element number of the array property has been reached, or the size of the data has reached the maximum size permitted for transmission.



9.3 Device Object Super Class Specifications

This section provides detailed specifications for the property configurations shared by all device object classes in the class groups corresponding to device objects (class group codes 0x00–0x06). These specifications are presented as the device object super class.

9.3.1 Overview of Device Object Super Class Specifications

The device object super class property is implemented by each device object class. Specifications for the device object super class are shown below.

The "Operating status" (EPC = 0x80) property implements the "Get" access rule for all device object classes, that it can be referenced from other nodes. Similarly, the "Status change announcement property map" (EPC = 0x9D), "Fault status" (EPC = 0x88), "Set properties map" (EPC = 0x9E), and "Get properties map" (EPC = 0x9F) properties also implement the "Get" access rule, that they can be referenced.

Table 9.2 shows the list of device object super class configuration properties.

Table 9.2 List of Device Object Super Class Configuration Properties (1/2)

Property Name	EPC	Property Content Value range (decimal notation)	Data Type	Data Size (Bytes)	Access Rule	Mandatory	Announce Status Change	Remarks
Operating status	0x80	Indicates ON/OFF status. ON = 0x30, OFF = 0x31	unsigned char	1	Set Get	 ○	○	
Installation location	0x81	Indicates the ECHONET instance installation location. See *Section 9.3.4 Installation Location Properties*	unsigned char	1	Set Get	 ○	○	
Specification	0x82	Indicates applicable specification version.	unsigned	4	Get	○		

version information		1st byte: Indicates major version number (digits to the left of the decimal point) in binary notation. 2nd byte: Indicates minor version number (digits to the right of the decimal point) in binary notation. 3rd byte: Indicates the order of release in ASCII notation. 4th byte: Reserved for future use (fixed at 0x00).	char					
Node identification number	0x83	Number to identify the node implementing the device object in the domain. First byte: Lower-layer communications software ID field 0x11 to 0x1F: Power Line Communication Protocol A and D Systems 0x31 to 0x3F: Designated low-power radio 0x41 to 0x4F: Extended HBS 0x51 to 0x5F: IrDA 0x61 to 0x6F: LonTalk®-dependent 0x71 to 0x7F: Bluetooth® 0x81 to 0x8F: Ethernet 0x91 to 0x9F: IEEE802.11/11b 0xA1: Power Line Communication Protocol C System The above is used to indicate the lower-layer communications software on which the node depends. 0xFF: Random number-based generation 0x00: Node identification number has not been set. Second to ninth bytes: Unique number field	unsigned char * 9	9	Get			
Manufacturer error code	0x86	Indicates a manufacturer error code. 1st byte: Indicates the data size of error code part. 2nd to 4th byte: Manufacture code Over 5th byte: Individual manufacture error code part	unsigned char x (Max) 225	Max 225	Get			
Current limit setting	0x87	Indicates set value of current limit (0 to 100%) 0x00 to 0x64 (= 0 to 100%)	unsigned short	3	Get	○	○	
Fault status	0x88	Indicates an encountered abnormality (sensor trouble, etc.). Fault encountered = 0x41, no fault encountered = 0x42	unsigned char	1	Get	○	○	
Error description	0x89	Error description See Table 9.4	unsigned short	2	Get			
Manufacturer code	0x8A	Stipulated in 3 bytes (To be specified by ECHONET Consortium)	unsigned char	3	Get	○		
Place of business code	0x8B	Stipulated in 3-byte place-of-business code (Specified individually by each manufacturer)	unsigned char	3	Get			

Table 9.2 List of Device Object Super Class Configuration Properties (2/2)

Property Name	EPC	Property Content Value range (decimal notation)	Data Type	Data Size (Bytes)	Access Rule	Mandatory	Announce Status Change	Remarks
Product code	0x8C	Stipulated in ASCII code (Specified individually by each manufacturer)	unsigned char	12	Get			
Serial number	0x8D	Stipulated in ASCII code (Specified individually by each manufacturer)	unsigned char	12	Get			
Date of manufacture	0x8E	Stipulated in 4 bytes Indicates the date in YYMD format (1 byte per character). YY: Year (07CF for 1999) M: Month (0C for December) D: Day (14 for 20th)	unsigned char	4	Get			
Power-saving operation setting	0x8F	Indicates the power-saving operation status of the equipment Power-saving operation = 0x41 Normal operation = 0x42	unsigned char	1	Set/ Get			
Time setting	0x97	Current time HH: MM 0x00 to 0x17 : 0x00 to 0x3B (= 0 to 23) : (= 0 to 59)	unsigned char *2	2	Set/ Get			
Date setting	0x98	Date YYYY: MM: DD 0 to 0x270F : 0 to 0x0C : 0 to 0x1F (= 0 to 9999) : (= 0 to 12) : (= 0 to 31)	unsigned char *4	4	Set/ Get			
Cumulative operation hours	0x9A	Indicates the cumulative operation hours up to the present using 1 byte for the unit and 4 bytes for the time. First byte: indicates the unit second: 0x41, minute: 0x42 hour: 0x43, day: 0x44 2nd to 5th bytes: Indicates the cumulative operation time as expressed in the unit specified by the first byte. 0x00000000 to 0xFFFFFFFF (0 to 4294967295)	unsigned char + unsigned long	1+4 Byte	Get			
SetM property map	0x9B	See Supplement 2	unsigned char* (MAX17)	Max. 17	Get	○		
GetM property map	0x9C	See Supplement 2	unsigned char* (MAX17)	Max. 17	Get	○		
Status Change Announcement property map	0x9D	See Supplement 2	unsigned char* (MAX17)	Max. 17	Get	○		

Set property map	0x9E	See Supplement 2	unsigned char* (MAX17)	Max. 17	Get	○		
Get property map	0x9F	See Supplement 2	unsigned char* (MAX17)	Max. 17	Get	○		

Note: A "o" in the status change announcement column denotes mandatory processing when the property is implemented.

9.3.2 Operating Status Property

The device object super class "Operating status" property indicates the operating status (ON/OFF) of the functions unique to each class in the actual device. In nodes implementing each device object class, when the functions unique to each class begin operation along with the node, this property can be implemented with a fixed value of 0x30. (However, the operating status of node communication functions is indicated in the node profile object "Operating status" property.)

9.3.3 Installation Location Property

The installation location property specifies with 1-byte bitmap data the location where the device is installed. This is a mandatory property that can be rewritten. When there has been a change to the value, the changed value must be announced as a general broadcast within the domain.

The 8 bits of the installation location property are assigned with a freely-defined designation bit, an installation location code, and a location number. When all bits are zero, it is a special code indicating that the installation location has not been set, and when all bits are 1, it is a special code indicating that the installation location is unfixed.

An explanation of each of the bits follows. Table 9.3 shows the relationship between the installation location, freely-defined designation bit, installation location code, and location number.

- Freely-defined designation bit (b7)

  This comprises a single b7 bit. When b7 = 1, the installation location code and location number can be freely defined.

  When b7 = 0, the installation location code and location number indicate the installation location of the device as specified in Table 9.3.
- Installation location code (b3–b6)

  This comprises 4 bits, b3 through b6. When b7 = 1, it can be freely defined. When b7 = 0, it indicates the type of the installation location of the device as specified in Table 9.3.
- Location number (b0–b2)

  This comprises 3 bits, b0 through b2. When b7 = 1, it can be freely defined. When b7 = 0, it is used to distinguish among 2 or more spaces of the same type. For example, when there are 2

bathrooms, they can be distinguished from each other by assigning the location number 001b to the first-floor bathroom and 010b to the second-floor bathroom.

When b7 = 0 and the location number field is 000b, it indicates that the installation location property has been initialized assuming that a device will be installed in the installation location shown by the installation location code ("location number not set.")

When the installation location property has been initialized without assuming a device installation location type, the value must be the "installation location not set" code (0x00). When it is inappropriate to set a particular type for the device installation location, the installation location property value must be the "installation location unfixed" code (0xFF).

0x01 through 0x07 shall be reserved for future use.

Table 9.3 Relationship Between Installation Location Space Name and Bit Assigned

	MSB							LSB
Type of installation location	Freely defined designation bit	Installation location code				Location number		
	b7	B6	b5	b4	b3	b2	b1	b0
Living room	0	0	0	0	1	"000b" –"111b" ("000b" means that the location number has not been set.)		
Dining room	0	0	0	1	0			
Kitchen	0	0	0	1	1			
Bathroom	0	0	1	0	0			
Toilet	0	0	1	0	1			
Washbowl	0	0	1	1	0			
Corridor	0	0	1	1	1			
Room	0	1	0	0	0			
Stairs	0	1	0	0	1			
Hall	0	1	0	1	0			
Spare room	0	1	0	1	1			
Garden/exterior	0	1	1	0	0			
Carport	0	1	1	0	1			
Veranda/Balcony	0	1	1	1	0			
Others	0	1	1	1	1			
Freely defined*	1	"0000000b" through "1111110b"						
Installation location not set	1	1	1	1	1	1	1	1
Installation location unfixed	1	1	1	1	1	1	1	1
Reserved for future use	1	"00000001b" through "00000111b"						

* "Freely defined" locations are provided mainly for use in stores and small- and medium-sized buildings.

9.3.4 Specification Version Information

Indicates the applicable specification version number with a 2-byte binary value and the order of Appendix release with a 1-byte ASCII code.

The first byte indicates the major version number (digits to the left of the decimal point). The second byte indicates the minor version number (digits to the right of the decimal point). The third byte indicates the order of release. To indicate Version 2.10 Release a, for instance, the contents of the first, second, and third bytes are 0x02 (2), 0x0A (10), and 0x61 (a), respectively.

The fourth byte, which is reserved for future expansion, is fixed at 0x00 in this Version.

9.3.5 Fault Status Property

The "fault status" property of the device object super class indicates the occurrence of an error in an actual device. The property code used as a property value is 0x41 when an error exists and 0x42 when no error exists.

9.3.6 Error Description Property

Table 9.4 shows the values that can be used as error description property values. A "restorable error" shall mean an error that is impeding the normal operation of the equipment but whose cause can be removed by the user. An "error that requires repair" shall mean an error that is impeding the normal operation of the equipment and requires repair by a technician to remove the cause.

The lower-order byte of the error description code shall indicate the broad classification of the error and the higher-order byte shall indicate the detailed classification of the error. The detailed classification of restorable errors will be defined for individual classes in future versions of the ECHONET Specification. If the intention is to use only the broad classification to classify restorable errors or errors that require repair, the higher-order byte of the error description code must be 0x00.

(1) Lower-order byte of the error description code

The lower-order byte of the error description code shall indicate the broad classification of the error. 0x00 shall mean that there is no error in the equipment and 0x01 to 0x09 shall mean that there is an error in the equipment that can be removed by the user. Errors are classified based on what user action is required to remove them. 0x01 shall indicate an error that can be removed by restarting the equipment by turning off the power once and then turning the power back on. 0x02 shall indicate an error that can be removed by restarting the equipment by reset operation. 0x03 shall indicate an error that can be removed by changing the mounting position, etc. of the equipment or opening or closing a lid or door. 0x04 shall indicate an error that can be removed by supplying fuel, water, air, etc. 0x05 shall indicate an error that can be removed by cleaning the equipment. 0x06 shall indicate an error that can be removed by replacing the battery, etc. 0x09 is a user-defined area and can be defined freely. 0x07 to 0x08 are reserved for future addition of definitions of restorable errors.

0x0A to 0x6E shall mean that there is an error in the equipment that requires repair. Errors are classified

based on their locations. A value between 0x0A and 0x13 shall indicate that a safety device is operating. A value between 0x14 and 0x1D shall indicate an error in the user interface system. A value between 0x1E and 0x3B shall indicate an error in the sensor system. A value between 0x3C and 0x59 shall indicate an error in an actuator, etc. A value between 0x5A and 0x6E shall indicate an error in the control board.

(2) Higher-order byte of the error description code

The higher-order byte of the error description code shall indicate the detailed error classification (subclassification of the broad classification).

When the lower-order byte is 0x00 (no error), the higher-order byte must be 0x00. 0x04 to 0xFF shall be reserved for future definitions.

When the lower-order byte is a value between 0x01 and 0x06, the higher-order byte may take 0x00 or any value between 0x04 and 0xFF. If the higher-order byte is 0x00, it shall mean that no detailed classification is made. 0x04 to 0xFF are reserved for future class-specific definitions.

When the lower-order byte is a value between 0x07 and 0x08, the higher-order byte may take 0x00 or any value between 0x04 and 0xFF. The value 0x00 and the values 0x04 to 0xFF are all reserved for future definitions.

When the lower-order byte is 0x09, the higher-order byte may take 0x00 or any value between 0x04 and 0xFF. The value 0x00 and the values 0x04 to 0xFF shall be freely defined by the user.

When the lower-order byte is a value between 0x0A and 0x6E, the higher-order byte may take 0x00 or any value between 0x04 and 0xFF. If the higher-order byte is 0x00, it shall mean that no detailed classification is made. The values 0x04 to 0xFF shall be freely defined by the user.

The values 0x006F to 0x03E8 shall be freely used by the user, for both the higher- and lower-order bytes, to indicate errors that require repair.

When the higher- and lower-order bytes are 0x03FF, it shall mean that there is an error but the method to remove the error or the location of the error cannot be identified.

The values 0x03E9 to 0x03FE are reserved, for both the higher- and lower-order bytes, for future definitions.

Table 9.4 Error Description Code Table

		Error Description Code (0x**%%)	
Broad classification		Lower-order byte (%%)	Higher-order byte (**)
No error		0x00	0x00: no error 0x04 to 0xFF: reserved for future use
Restorable error	The error can be removed by turning off the operation/ power switch and turning it back on or by unplugging the power plug and plugging it again.	0x01	0x00: no detailed classification

	The error can be removed by pressing the reset button.	0x02	0x04 to 0xFF: reserved for future use
	The error can be removed by changing the mounting position, etc. of the equipment or opening or closing a lid or door.	0x03	
	The error can be removed by supplying fuel, water, air, etc.	0x04	
	The error can be removed by cleaning the equipment (filter cleaning, etc.).	0x05	
	The error can be removed by replacing the battery, etc.	0x06	
	Reserved for future use.	0x07 to 0x08	0x00, 0x04 to 0xFF
	User definition area	0x09	0x00, 0x04 to 0xFF
Error that requires repair	The operation or status is abnormal or a safety device is operating.	0x0a to 0x13	0x00: no detailed classification 0x04 to 0xFF: user definition area
	A switch-related error	0x14 to 0x1D	
	A sensor-related error	0x1E to 0x3B	
	An error in a functional component	0x3C to 0x59	
	A control board-related error	0x5A to 0x6E	
	User definition area	0x006F to 0x03E8	
There is a serious problem.		0x03FF	
Reserved for future use.		0x03E9 to 0x03FE, 0x**6F to 0x**FF (**: 04 to FF)	

9.3.7 Manufacturer Code Property

The property value of the manufacturer code property uses 3-byte codes to indicate individual manufacturers. The ECHONET Consortium assigns a manufacturer-specific property value to each ECHONET Consortium member.

9.3.8 Place-of-Business Code Property

The property value of the place-of-business code property uses 3-byte codes to indicate the place of business of individual manufacturers. The property value of the place-of-business code property is not stipulated by the ECHONET Consortium, but instead shall be stipulated by individual manufacturers.

9.3.9 Product Code Property

The property value of the product code property uses 12-byte ASCII codes to indicate the products of various manufacturers. The property value of the product code property is not stipulated by the ECHONET Consortium, but instead shall be stipulated by individual manufacturers. If the length of the property value of the product code property is less than 12 bytes, the value shall be stored in the data field in such a way that the data field is filled from the front-most subfield toward the rear-most subfield, and the remainder of the data field shall be padded with zeros or spaces.

9.3.10 Serial Number Property

The property value of the serial number property uses 12-byte ASCII codes to indicate the product serial numbers of various manufacturers. The property value of the serial number property is not stipulated by the ECHONET Consortium, but instead shall be stipulated by individual manufacturers. If the length of the property value of the serial number property is less than 12 bytes, the value shall be stored in the data field in such a way that the data field is filled from the front-most subfield toward the rear-most subfield, and the remainder of the data field shall be padded with zeros or spaces.

9.3.11 Date-of-Manufacture Property

The property value of the date-of-manufacture property uses four bytes to indicate the date of manufacture of various manufacturer products. Specifically, it uses two bytes to indicate the year and one byte each to indicate the month and day.

9.3.12 Property Map Property

The device object super class provides data that comprises 5 property maps and describes the services that can be provided by the properties published by the objects. Four of the 5 property maps, namely, the "Set property map," "Get property map," "SetM property map" and "GetM property map" show what access rules (see Part II, Section 4.2.8; hereinafter referred to as "AR" or "ARs") are supported by the properties published by the implemented objects in terms of product specifications.

The "Status Change Announcement property map" indicates that an intra-domain simultaneous broadcast must be performed when a property value change occurs.

The map formats are as shown in Appendix 2. If no applicable property exists, the number of properties shall be set to "0" and the remaining bytes (the second byte and the succeeding bytes) shall be left blank.

The definitions of the property maps are as follows:

(1) Set property map

A property map that shows the properties associated with the AR "Set." Array properties that permit lump writes require that their EPCs be registered in the Set property map.

(2) Get property map

A property map that shows the properties associated with the AR "Get." Array properties that permit lump reads require that their EPCs be registered in the Get property map.

(3) SetM property map

This property map indicates the properties relating to the "SetM" AR.

(4) GetM property map

This property map indicates the properties relating to the "GetM" AR.

(5) Status Change Announcement property map

This property map lists the properties that are set for a general broadcast of changes in their values. In addition to the intra-domain general broadcast stipulated in the "Status Change Announce" column for

ECHONET specifications for various object properties supported by product specifications, properties for making a unique "status change announcement" in compliance with product specifications are included as well. This property map does not include a status notification that is set by the "communication definition object for specifying the status notification method", which is described later.

No associated property maps are stipulated for the "AddM", "DelM", "AddMS", "Anno", "AnnoM", and "CheckM" ARs.

A property that is defined in a property map as a property that supports an access rule must support that access rule. In the case of a property that is not defined in a property map as a property that supports an access rule, whether or not that property supports an access rule shall depend on the actual implementation in the device.

However, the above requirements do not apply for properties that have been altered by a "property status referencing request compliability communication definition" object or a "property status notifiability communication definition" object.

9.3.13 Node Identification Number Property

The node identification number is used to identify nodes within a domain as unique entities. This property comprises a lower-layer communication software ID field that stores an ID showing the type of lower-layer communication software and a unique number field that stores a unique identification number assigned to each product using a specified method for each lower-layer communication software program. The definition of this unique number is given in the specifications for lower-layer communication software programs in Part III (However, Version 3.00 and later versions of the ECHONET Specifications define this unique number for IP/Bluetooth-dependent lower-layer communication software and IP/Ethernet/IEEE802.3-dependent lower-layer communication software.) This unique number indicates the hardware address. If the hardware address is less than 8 bytes, it shall be stored in the unique number field from the highest-order byte and the remaining bytes shall be padded with 0.

The ECHONET Communication Middleware can acquire this number from the lower-layer communication software by using the node identification number request service (common lower-layer communication interface). However, the lower-layer communication software must be Version 3.00 or a later version. When the ECHONET Communication Middleware cannot acquire the unique number from the lower-layer communication software (i.e. when a node identification number request is rejected), the ECHONET Communication Middleware shall generate a unique number using a random number generation algorithm. The random number generation algorithm must be implemented in such a way that the possibility of an overlap with a unique number held by another node is minimized. It is recommended that the node identification number generated using the random number generation algorithm be stored in a nonvolatile memory managed by the ECHONET Communication Middleware so that it can be read out when a warm start is performed. If two or more device objects are implemented in one node, the node identification numbers held by the device objects must match. By the same token, the node identification number held by the device object and that held by the node profile object must also match.

9.3.14 Manufacturer Error Code Property

When there is an error in the equipment, it shall be indicated using a manufacturer-defined error code.

The first byte shall indicate the data size of the error code section.

The second through fourth bytes shall give the 3-byte manufacturer code that is specified by the ECHONET Consortium.

The error code section, which starts from the fifth byte, shall store the manufacturer-defined error code

If this property is implemented, the "error description" property must also be implemented.

Data size of the error code section (1 byte)	Manufacturer code (3 bytes)	Error code section (defined by the manufacturer) (Max. 221 bytes)

9.3.15 Current Limit Setting Property

Contains a value to specify the maximum consumable current (current limit). This property shall take a value between 0 and 100 (0x00 and 0x64). The unit shall be in %. The percentage value multiplied by the specified maximum current value for the equipment associated with the object shall be the maximum consumable current at that point in time. The value 100 means that no limit is imposed. If it is not possible to impose a current consumption limit equal to the value specified by this property, the nearest value shall be used provided that the value does not exceed the setting.

9.3.16 Power-saving Operation Setting Property

Contains a value to indicate whether or not the equipment associated with the object is operating in power-saving mode. 0x41 indicates that the equipment is operating in power-saving mode and 0x42 indicates that the equipment is operating in normal (non-power-saving) mode.

9.3.17 Cumulative Operation Hours Property

Indicates the cumulative operation time.

The first byte shall indicate the unit for the cumulative operation time. 0x41, 0x42, 0x43 and 0x44 shall mean that the unit is in seconds, minutes, hours and days, respectively.

The second through fifth bytes shall indicate the cumulative operation time as expressed in the unit specified by the first byte. These bytes shall be treated as a single piece of unsigned long data. The value range for the cumulative operation time (second through fifth bytes) shall be between 0x0000 and 0xFFFFFFFF (0 and 4294967295). 0xFFFFFFFF shall be the overflow code.

The counting specifications and the conditions for starting and stopping counting shall be equipment-dependent and are not defined herein.

9.3.18 Time Setting Property

Indicates the current time using a value between 0x00 and 0x17 (0 and 23) for the hour and a value between 0x00 and 0x3B (0 and 59) for the minute. The first byte shall indicate the hour and the second byte shall indicate the minute.

9.3.19 Date Setting Property

Indicates the date using a value between 0x0000 and 0x270F (0 and 9999) for the year, a value between 0x00 and 0x0C (0 and 12) for the month and a value between 0x00 and 0x1F (0 and 31) for the day.

The first and second bytes (2 bytes in total) shall indicate the year and shall be treated as a single piece of unsigned short data. The third and fourth bytes shall indicate the month (1 byte) and day (1 byte), respectively.

9.4 Sensor-related Device Class Group Objects: Detailed Specifications

Stated in the Appendix (*Detailed Requirements for ECHONET Device Objects*)

9.5 Air Conditioning-related Device Class Group Objects: Detailed Specifications

Stated in the Appendix (*Detailed Requirements for ECHONET Device Objects*)

9.6 Housing/Equipment-related Device Class Group Objects: Detailed Specifications

Stated in the Appendix (*Detailed Requirements for ECHONET Device Objects*)

9.7 Cooking/Housework-related Device Class Group Objects: Detailed Specifications

Stated in the Appendix (*Detailed Requirements for ECHONET Device Objects*)

9.8 Health-related Device Class Group Objects: Detailed Specifications

Stated in the Appendix (*Detailed Requirements for ECHONET Device Objects*)

9.9 Management/Control-related Device Class Group Objects: Detailed Specifications

Detailed class specifications other than those in Section 9.9.1 "Detailed Specifications for Secure Communication Common Key Setup Node Class", are stated in the Appendix "Detailed Requirements for ECHONET Device Objects".

9.9.1 Detailed Specifications for Secure Communication Common Key Setup Node Class

This class is to be implemented by nodes having the key setup function. It is used to request ECHONET secure communication common key redistribution by writing to the common key distribution request property of this class in a shared-key-based authentication/enciphered message format.

Class group code: 0x05
Class code: 0xFC
Instance code: 0x01

Table 9.4 Secure Communication Common Key Setup Node Class

Property Name	EPC	Property Content Value range (decimal)	Data Type	Size (Bytes)	Access Rule	Mandatory	Announce Status Change	Remarks
Common key distribution request	0xC0	Receives a request for ECHONET secure communication common key (User Key/Service Provider Key) setup. Request trigger for ECHONET secure communication common key (User Key/Service Provider Key) setup = 0x00.	Unsigned char	1	Set	○		(2)

(1) Operating status (inherited from the properties of the device object super class)
Indicates whether or not the function native to this class is operating (ON/OFF). In the node mounting this class, if the function of this class is started concurrently with the start of node operation, this property may be implemented at a fixed value of 0x30 (operating status ON).

(2) Common key distribution request
Requests the redistribution of an ECHONET secure communication common key when a property value (ESV = 0x60, 0x61) is written to this property (0x00).

9.10 Profile Object Class Group Specifications

This section provides detailed specifications for the property configurations shared by all profile object classes in the profile object class group (class group code 0x0E). These specifications are presented as the

profile object super class.

9.10.1 Overview of Profile Object Super Class Specifications

Profile object super class properties are implemented by each profile object class. Specifications for the profile object super class are shown in Table 9.4 below.

Table 9.4 List of Profile Object Super Class Configuration Properties

Property Name	EPC	Contents of Property Value range (decimal notation)	Data Type	Data Size (Bytes)	Access Rule	Mandatory	Announce Status Change	Remarks
Fault status	0x88	Indicates an encountered abnormality (sensor trouble, etc.). Fault encountered = 0x41, no fault encountered = 0x42	unsigned char	1	Get			(1)
Manufacturer code	0x8A	Stipulated in 3 bytes (To be specified by ECHONET Consortium)	unsigned char	3	Get	○		
Place of business code	0x8B	Stipulated in 3-byte place-of-business code (Specified individually by each manufacturer)	unsigned char	3	Get			
Product code	0x8C	Stipulated in ASCII code (Specified individually by each manufacturer)	unsigned char	12	Get			
Serial number	0x8D	Stipulated in ASCII code (Specified individually by each manufacturer)	unsigned char	12	Get			
Date of manufacture	0x8E	Stipulated in 4 bytes YYMD (1 byte each) YY: Western calendar (1999:07CF) M: Month (Dec = 0C) D: Day (20th = 14)	unsigned char	4	Get			
SetM property map	0x9B	See Supplement 2	unsigned char	Max. 17	Get	○		
GetM property map	0x9C	See Supplement 2	unsigned char	Max. 17	Get	○		
Status Change Announcement property map	0x9D	See Supplement 2	unsigned char	Max. 17	Get	○		
Set property map	0x9E	See Supplement 2	unsigned char	Max. 17	Get	○		
Get property map	0x9F	See Supplement 2	unsigned char	Max. 17	Get	○		

Note: A "o" in the status change announcement column denotes mandatory processing when the property is implemented.

(1) Fault status

Indicates a fault in the given profile object. For example, the fault state property for the ECHONET Communications Processing Block profile object indicates whether or not there is a fault in the ECHONET Communications Processing Block software. Since specific fault content differs for each object class, detailed specifications are provided separately.

9.10.2 Property Map

For the five property maps stipulated for the profile object super class, the properties stipulated for the profile object are the same as provided in Section 9.3.5.

9.11 Profile Class Group: Detailed Specifications

This section provides detailed code and property specifications for each ECHONET object belonging to the profile class group (class group requirement code X1 = 0x0E). Table 9.4 provides a list of the objects for which detailed specifications are provided in this section. Properties shared (for which a succession relationship is established) by all profile object classes in this object class group are indicated as super classes in Section 9.5 *Profile Object Class Group Specifications*. Regarding detailed items for each object class, the properties described in these super classes will not be listed unless there are special additional specifications. In the detailed specifications, the indication of an object as being "required" signifies that, when the given object is present, the combined property and service of that object must be implemented. One profile object class exists at each node (this may not be the case when the profile object classes are not mandatory). When, for instance, an ECHONET router or other communication device consisting of two or more nodes is used, each node has a node profile and router profile. Therefore, the device consisting of such nodes has two or more node profiles and router profiles.

However, a node that has a NetID server profile must also have a router profile.

Table 9.5 Object Class List (Profile Class Group)

Class Group Code	Class Code	Object Class Name	○ = Mandatory
0x0E	0xF0	Node profile	○
	0xF1	Router profile	○ (when the router functions are implemented)
	0xF2	ECHONET communication processing section profile	
	0xF3	Protocol difference absorption processing section profile	
	0xF4	Lower-layer communication software profile	
	0xF5	NetID server profile	○ (when the NetID server functions are implemented)

9.11.1 Node Profile Class: Detailed Specifications

Class group code: 0x0E

Class code: 0xF0

Instance code: 0x01

Property Name	EPC	Contents of Property Value range (decimal notation)	Data Type	Data Size (Bytes)	Access Rule	Mandatory	Announce Status Change	Remarks
Operating status	0x80	Indicates node operating status. Booting = 0x30, not booting = 0x31	unsigned char	1	Set Get	 ○	○	(1)
Version information	0x82	Indicates ECHONET version used by communication middleware and message types supported by communication middleware. 1st byte: Indicates major version number (digits to left of decimal point) in binary notation. 2nd byte: Indicates minor version number (digits to right of decimal point) in binary notation. 3rd and 4th bytes: Indicate message types with a bitmap.	unsigned char*4	4	Get	○		(20)
Node identification number	0x83	Number to identify the node implementing the device object in the domain. First byte: Lower-layer communications software ID field 0x11 to 0x1F: Power Line Communication Protocol A and D Systems 0x31 to 0x3F: Designated low-power radio 0x41 to 0x4F: Extended HBS 0x51 to 0x5F: IrDA 0x61 to 0x6F: LonTalk® 0x71 to 0x7F: Bluetooth 0x81 to 0x8F: Ethernet 0x91 to 0x9F: IEEE802.11/11b 0xA1: Power Line Communication Protocol C System The above is used to indicate the lower-layer communications software on which the node depends. 0xFF: Random number-based generation 0x00: Node identification number has not been set. Second to ninth bytes: Unique number field	unsigned char*9	9	Get	○		(25)
Fault content	0x89	Fault content	unsigned	2	Get			(2)

		0x0000–0x03E8 (0–1000)	short					
Unique identifier data	0xBF	Stipulated in 2 bytes See (3) below.	unsigned short	2	Set/Get	○		(3)
EA	0xE0	Held value of all Eas Byte 1: Number of held EAs Byte 2 and higher: List EAs (2 bytes each)	unsigned char* (MAX)247	Max 247	Set Get	 ○		(4)
Net ID	0x E1	1-byte Net ID value Initial value = 0x00	unsigned char	1	Set/Get	○		(5)
Node ID	0x E2	1-byte Node ID value Initial value = 0x00	unsigned char	1	Set Get	 ○		(6)
Default router data	0x E3	EA value for default router Initial value = 0x0000 (no default router data)	unsigned short	2	Set/Get	○		(7)
All router data	0x E4	All router data within the domain See Supplement 3	unsigned char	Max 246	Set/Get			(8)
Lock control status	0x EE	Shows status lock control action status Control = 0x30;no control = 0x31	unsigned char	1	Get	○		(9)
Lock control data	0x EF	Lock control data Bytes 1–2: EA of lock source Byte 3: Lock time	unsigned char*3	3	Set/Get			(10)

Note: A "o" in the status change announcement column denotes mandatory processing when the property is implemented. Current device: self-node class; other device: other node class (see Part 2, Chapter 3).

Note: When there are two or more nodes within a device, each node has this profile class. However, these nodes must have common values for the EA (0xE0) and all router data (0xE4) properties because they constitute a single device. (continued on the next page)

(continued)

Property Name	EPC	Contents of Property Value range (decimal notation)	Data Type	Data Size (Bytes)	Access Rule	Mandatory	Announce Status Change	Remarks
Self-node instance list page	0x D0	List of instance numbers in the element-stipulated classes between 0x00 and 0x7F Byte 1: Number of instances in class stipulated by element Bytes 2–17: See Supplement 4	unsigned char*17 *(no. of elements)	17	GetM Gert	○		(11)
Self-node class list	0x D2	Element-stipulated class groups and code range class list Byte 1: Number of classes in class group stipulated by element Bytes 2–17: See Supplement 5	unsigned char	17	GetM Get	○		(12)
Self-node instance count	0x D3	Number of instances held in self-node Bytes 1–3: Number of instances	unsigned char	3	Get	○		(13)
Self-node class count	0x D4	Number of classes held in self-node Bytes 1–2: Number of classes	unsigned char	2	Get	○		(14)

Instance change class	0x D5	Classes with a change in instance configuration Byte 1: Number of classes reported on Byte s 2–17: List of class codes (most significant 2 bytes of EOJ)	unsigned char	Max. 17	Anno	○		(15)
Self-node instance list S	0x D6	List of instances within self-node Byte 1: Number of instances Bytes 2–16: List of class instance codes (EOJ)	unsigned char	Max. 16	Get	○		(16)
Self-node class list S	0x D7	List of classes within self-node Byte 1: Number of classes Bytes 2–17: List of class codes (most significant 2 bytes of EOJ)	unsigned char	Max. 17	Get	○		(17)
Related other node EA list	0x D8	List of EAs of other nodes related by communications Byte 1: Number of EAs in list Bytes 2–245: Lists EA2 byte codes	unsigned char	Max. 247	Set/ Get			(18)
Related other node EA count	0x D9	Number of other nodes related in terms of communications Bytes 1–2: Number of EAs	unsigned char	2	Get			(19)
Group broadcast number	0xDA	Group broadcast number First byte: Number of group broadcast numbers to setSecond and succeeding bytes: When the number of group broadcast numbers to set is 8 or less, the group numbers shall be listed.When the number of group broadcast numbers to set is 9 or more, the requirements specified in (27) below shall be satisfied.	unsigned char x (MAX) 33	Max. 33	Set/ Get			(27)

Note: A "o" in the status change announcement column denotes mandatory processing when the property is implemented. Current device: self-node class; other device: other node class (see Part II, Chapter 3).

Note: When there are two or more nodes within a device, each node has this profile class. However, these nodes must have common values for the EA (0xE0) and all router data (0xE4) properties because they constitute a single device.

*1: Must be implemented when the secure communication functions are implemented. However, a node that has the secure communication common key setting node class shall not accept a Set or SetM request. (continued on the next page)

(continued)

Property Name	EPC	Contents of Property Value range (decimal notation)	Data Type	Data Size (Bytes)	Access Rule	Mandatory	Announce Status Change	Remarks
Secure communication common key setup (User Key)	0xC0	ECHONET secure communication common key (User Key) 0x000000000000– 0xFFFFFFFFFFFFFFFF	unsigned char x 8	8	Set	○*1		(21)
Secure communication common key setup (Service Provider Key)	0xC1	ECHONET secure communication common key (Service Provider key) 0x000000000000– 0xFFFFFFFFFFFFFFFF	unsigned char x 8	8	SetM	○*1		(22)

Secure communication common key switchover setup (User Key)	0xC2	Sets the ECHONET common key (User Key) switchover state that is updated by the User Key setup property. Common key setup incomplete = 0x40 Common key distribution complete = 0x41 Common key switchover in progress = 0x42 Common key update complete = 0x43	unsigned char	1	Set/Get	○*1		(23)
Secure communication common key switchover setup (Service Provider Key)	0xC3	Sets the ECHONET common key (User Key) switchover state that reflects the ECHONET common key (Service Provider Key) update by the Service Provider Key setup property. Common key setup incomplete = 0x40 Common key distribution complete = 0x41 Common key switchover in progress = 0x42 Common key update complete = 0x43	unsigned char	1	SetM/ GetM	○*1		(24)
Secure communication common key setting (Serial Key)	0xC4	Indicates the encryption method for communication with a Serial Key 0x01: AES-CBC Other values: reserved for future use	unsigned char	1	Get			(26)

Note) A "o" in the status change announcement column means that the processing is mandatory when the property is implemented.

Equipment itself: home node class; other equipment: other node classes (see Part II, Chapter 3)

Note) When there are two or more nodes in a piece of equipment, this profile class shall be held for each node, but it is necessary to hold a common value for the equipment with regard to the EA (0xE0) and "data on all routers" (0xE4) properties.

*1: Must be implemented only when the secure communication functions are implemented. However, a node that has the secure communication common key setting node class shall not accept a Set or SetM request.

(1) Operating status

Indicates whether or not the current operating status permits ECHONET node communications.

(2) Fault content

In Version 1.0 β, this will be the same as the code assignment for fault content properties for device objects.

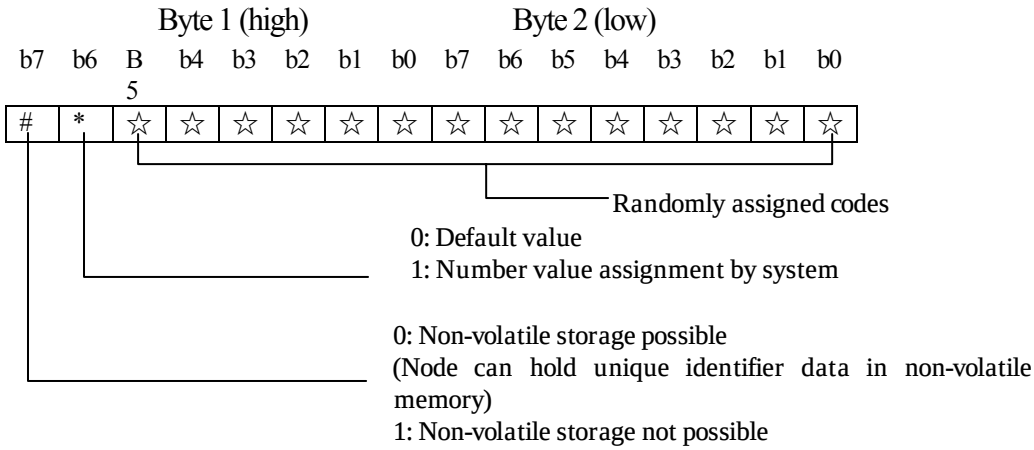
(3) Unique identifier data

Data that guarantees that each node can be uniquely identified within a domain and that each node can be treated as an unchanging entity even after devices are moved (e.g., a change in subnet). Decided using a default value or an assigned value. The unique identifier data value that has been preset on the device side shall be called the default value, and unique identifier data values that are set after entry into the ECHONET system by other ECHONET nodes shall be called assigned values.

As a rule, unique identifier data must be held in a non-volatile memory. The only exception to this rule (i.e., when unique identifier data need not be held in a non-volatile memory) is when the combination of manufacturer code property value and serial number property value guarantees unique identification. If

non-volatile storage is not provided, the second-most significant bit (b6) is set to 0 as an exceptional default value so that setup can be performed by an ECHONET node responsible for numbering (erasure upon power off is permissible). Other ECHONET nodes shall be prohibited from setting a unique identifier data value that has 0 in b6 of the first byte.

The code representation specifications are as follows:



Each node sets the default value using the following method:

- Values for the 14 bits 0x0001–0x3FFF are created randomly. Any method of random number generation is acceptable.
- The most significant bit (b7) must be either 0 or 1 in accordance with node specifications.
- The second-most significant bit (b6) is set to 0.

Even if initial values are duplicated, the duplication can be resolved by newly assigning an appropriate non-duplicate value from one of the nodes in the system. When newly assigning a value, the value of the second-most significant bit must be set to 1. Note that the value of the most significant bit is decided by the node in accordance with the above figure and cannot be changed. In response to a request to write this property, the receiving side masks the most significant bit.

(4) EA

All EAs within a device are retained. One EA exists in a node. Therefore, if a device consists of one node, there is only one EA. For a router or other device consisting of two or more nodes, however, all associated EAs are retained by this property.

(5) Net ID

Indicates the Net ID of the local subnet. It is mainly used for Net ID distribution from a router to nodes in the local subnet router startup sequence. Since this is a code in the first byte position of the EA, it is explicitly indicated in a transmitted/received ECHONET message under normal conditions. If the received request attempts to write a Net ID value different from the currently owned Net ID value into this property, the status changes to the stop state and then the execution of a warm start starts.

(6) Node ID

Indicates the Node ID of the local node.

(7) Default router data

ECHONET nodes other than a router internally retain the ECHONET address of one of the routers connected to the same subnet as the "default router" data at the time of Net ID setup. There is no requirement for determining which router becomes the "default router".

(8) All router data

This data exists in nodes that perform advanced routing processing (see Section 6.3.2). It holds all router data for the domain.

(9) Lock control status

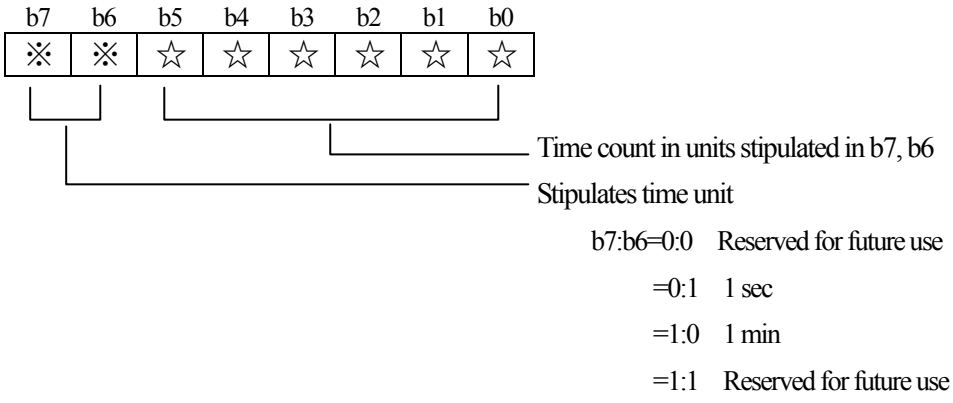
Indicates whether or not the entire node is receiving lock control from another node. When being controlled, it will not accept control from a node other than the one shown in the "lock control data" property. Note, however, that the read service can be received without specifying the other party even during lock control.

(10) Lock control data

Shows the data for the lock communications partner (i.e., the lock control source) and the time for which lock control is active. Lock control time is counted down after being set by the source of lock control; it lets nodes other than the lock control source know how long they must wait for the lock to be removed when they receive a control request. When the lock is removed, lock control time is cleared.

When the lock control status property indicates that lock control is in effect, the lock control data property, as a rule, will not accept lock control data setting requests from any node. The only exception to this rule is when it is set to a value that indicates a shortening of the lock time set from the lock control source in the lock control data. When lock time is set to 0x00, this signifies removal of the lock.

Byte 1 of the lock time is configured as shown below:



(11) Self-node instance list

List for each class stipulated for the instances disclosed by the self-node. Classes to be included are the device object and service object classes specified by ECHONET and corresponding to class group codes

0x00–0x06, 0x0D. Classes are stipulated using elements, and the instance count and list of instance numbers to be held are indicated with bitmaps. Byte 1 shows the total number of instances in the class stipulated by the element. Bytes 2–17 are bitmaps for existing instances. Instances to be included in the list are limited to those disclosing to other nodes the services provided by the given instance.

This property need not be implemented when the number of instances for the device object and service object classes disclosed by the self-node is less than 5.

(12) Self-node class list

Lists the classes for each stipulated class group disclosed by the self-node. Class group and range are stipulated by the element. Class codes disclosed in the stipulated class group range are indicated with bitmaps. There are two ranges. In Range 1, the least significant byte of the class code is in the range 0x00–0x7F; in Range 2, the least significant byte is in the range 0x80–0xFF.

This property need not be implemented when the number of classes for the self-node is 8 or less.

(13) Self-node instance count

Indicates the total number of instances in all device object and service object classes disclosed by the self-node.

(14) Self-node class count

Indicates the total number of classes disclosed by the self-node.

(15) Instance change class

Whenever a new instance is added or deleted during startup or system operation, or whenever there is some other change in the instance configuration disclosed to the network, this property announces to the network a class code corresponding to the change. This property was designed exclusively as an announcing property, with the expectation that it would trigger recognition by other nodes of the details of instance changes. The number of classes to be reported in the given message is inserted in Byte 1, while the classes experiencing changes in instance configurations are listed in Bytes 2–17 (most significant 2 bytes of EOJ). As many as 8 classes can be announced at one time. When there are changes in more than 8 classes, the announcement is split into two or more component parts, assuming that after changes in these 8 classes, the remaining classes also change. Configuration changes are to be announced for self-node device object and service object class instances.

(16) Self-node instance list S

Lists the device object and service object instances disclosed by the self-node. When the total number of instances is 6 or more, this number is inserted in the instance count in Byte 1, and Bytes 2 and after are left blank for transmission. The value of Byte 1 is specified as follows:

0x00–0xFE	Total number of instances (when 254 or less) instruction
0xFF	Overflow (when 255 or more) instruction

(17) Self-node class list S

Lists the classes disclosed by the self-node with the exception of node profiles. When the total number of classes is 6 or more, the total number is placed in the first byte position as the class count, with the second and subsequent byte positions left blank for transmission. The value of Byte 1 is specified as follows:

0x00–0xFE	Total number of classes (when 254 or less) instruction
0xFF	Overflow (when 255 or more) instruction

(18) Related other node EA list

Lists the EAs for partner nodes with instances performing status management or data exchange, such as linked relationships. EAs are listed in Byte 2. Up to 122 nodes may be listed. Specifications will be provided at a future date (in V1.0 or after) for cases in which there are relations with more than 122 nodes.

Note:

The EAs included in this list, by disclosing the fact that data is being exchanged with some instance present in the node indicated by the EA, is an attempt to comply with the plug-and-play functionality that automatically forms relationships via the network by manipulating linked relationships between instances as well as maintenance and other relationships. Therefore, when the partner instance is to be stipulated explicitly and communications performed, it is useful to include the EA of the partner.

(19) Related other node EA count

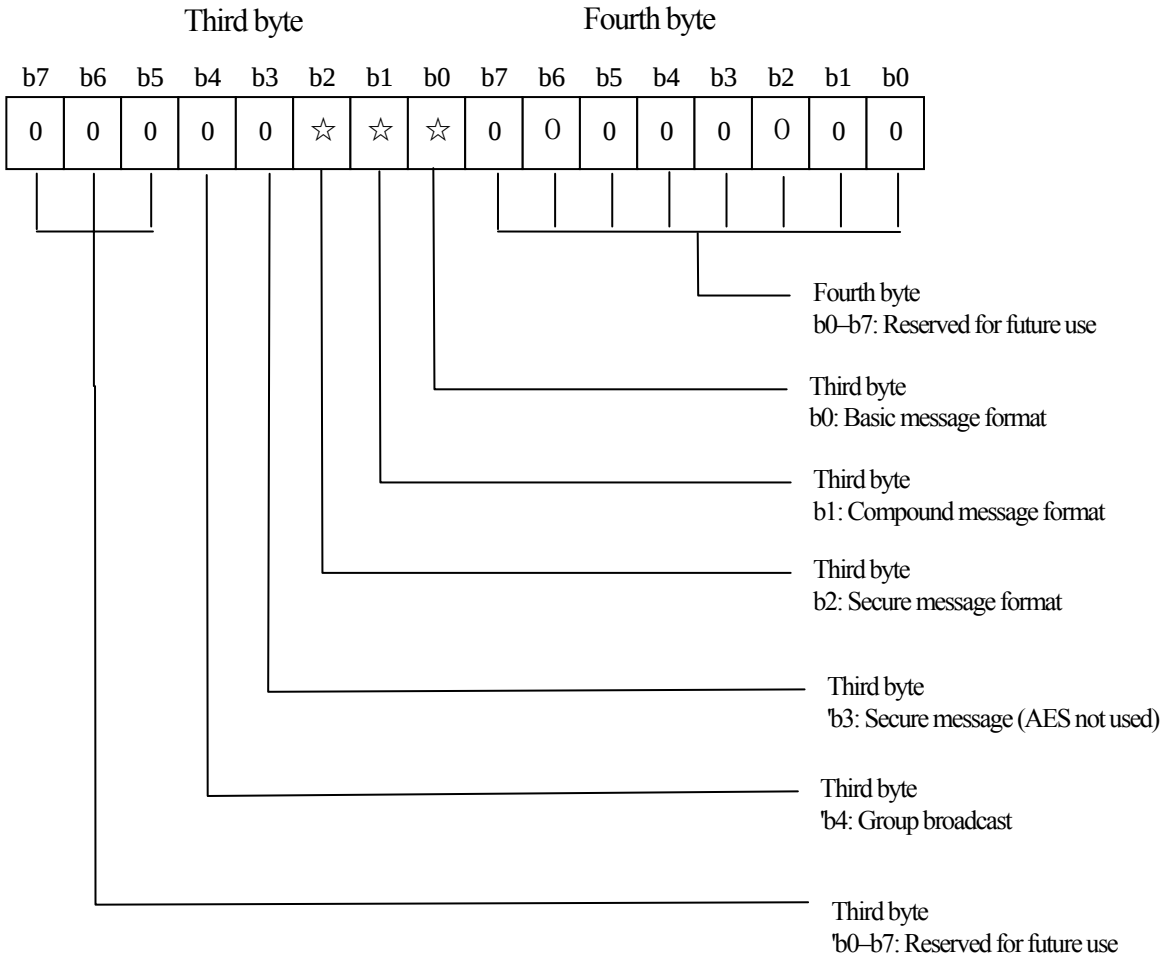
Number of EAs of partner nodes with instances performing status management or data exchange, such as linked relationships.

(20) Version data

A 2-byte binary value shows the communication middleware version number, and a 2-byte bitmap indicates the message types supported by the communication middleware.

The first byte indicates the major version number (digits to the left of the decimal point). The second byte indicates the minor version number (digits to the right of the decimal point). To indicate Version 2.10, for instance, the contents of the first and second bytes are 0x02 (2) and 0x0A (10), respectively.

The third and fourth bytes indicate the supported message types. When the bit value is 1, it means that the associated message type is supported. The figure below shows the relationship between the bits and supported message types.



(21) Secure communication common key setting (User Key)

A common key is delivered by writing a property value (ESV = 0x61) using a message authentication and encryption method that uses the node Serial Key or User Key (common key (User Key) used in User Level authentication and encryption communication) (ECHONET secure communication). The first and second bytes of the property value shall indicate the encryption method and key size, respectively, and the bytes that follow shall indicate the common key (New Master Key).

First byte:

0x01: AES-CBC method

Other values: reserved for future use

If a property value write is performed using the Serial-Key-based message authentication and encryption method, the User Key change setting property shall shift to the common key updating complete state "0x43." If a property value write is performed using the User Key (Pre Master Key)-based message authentication and encryption method, the User Key change setting property shall shift to the common key delivery complete state "0x41."

(22) Secure communication common key setting (Service Provider Key)

A common key is delivered by performing a property value element designation write (ESV = 0x65) to this property using an authentication and encryption type message that uses the Service Provider Key, User Key or Serial Key of the node which is a common key (Service Provider Key) used in Service Provider Level authentication and encryption communication (ECHONET secure communication). Each element number indicates the Service Provider Key Index and corresponds to b3-b0 of the secure key header (SKH). The first and second bytes of the property value shall indicate the encryption method and key size, respectively, and the bytes that follow shall indicate the common key (New Master Key).

First byte:

0x01: AES-CBC method

Other values: reserved for future use

If a property value element designation write is performed using a Serial-Key-based message authentication and encryption method, the Service Provider Key change setting property shall shift to the common key updating complete state "0x43."

If a property value write is performed using a Service Provider Key (Pre Master Key)-based message authentication and encryption method, the Service Provider Key change setting property shall shift to the common key delivery complete state "0x41."

(23) Secure communication common key switchover setup (User Key)

This property indicates the switchover status of the common key (User Key) for User Level authentication/enciphered communication in an ECHONET secure communication process. The switchover from Pre-Master Key to New Master Key occurs when a property value (switchover in progress (0x42) or update completion (0x43)) is written into this property in an authentication/enciphered message format based on the User Key.

When a property value write (ESV = 0x62) is performed in an authentication/enciphered message format based on the User Key to write "Common key not set" (0x40) into this property, the node status changes to the insecure communication state. "Common key distribution completion" (0x41) indicates a state, and its value cannot be written.

To warm-start a node in the secure communication state, this property must be announced (ESV = 0x73) in an enciphered message format.

(24) Secure communication common key switchover setup (Service Provider)

This property indicates the switchover status of the common key (Service Provider Key) for Service Provider Level authentication/enciphered communication in an ECHONET secure communication process. The switchover from Pre-Master Key to New Master Key occurs when a property value element-stipulated write (ESV = 0x65) is performed to write "switchover in progress (0x42)" or "update completion" (0x43) into this property in an authentication/enciphered message format based on the User Key. Element numbers indicate a Service Provider Key Index and correspond to the b3 to b0 values of the secure key header (SKH). "Common key distribution completion" (0x41) indicates a state, and its value cannot be written.

(25) Node identification number

Node identification number is a number used to identify nodes within a domain as unique entities. This property comprises a lower-layer communication software ID field that stores an ID showing the type of lower-layer communication software and a unique number field that stores a unique identification number that is assigned to each product using a specified method for each lower-layer communication software program. The definition of this unique number is given in the specifications for lower-layer communication software programs in Part III. (However, Version 3.00 and later versions of the ECHONET Specifications define this unique number for IP/Bluetooth-dependent lower-layer communication software and IP/Ethernet/IEEE802.3-dependent lower-layer communication software.) This unique number indicates the hardware address. If the hardware address is less than 8 bytes, it shall be stored in the unique number field from the highest-order byte and the remaining bytes shall be padded with 0.

Each ECHONET node must have at least one device object, but the node identification number property value must be the same as the value of the node identification number property held by the device object.

(26) Secure communication common key setting (Serial Key)

The encryption method is specified by performing a property value write (ESV = 0x61) using a message authentication and encryption method that uses the common key (Serial Key) used in supervisor-level authentication and encryption communication (ECHONET secure communication). The property value (first byte) shall indicate the encryption method.

First byte:

0x01: AES-CBC method

Other values: reserved for future use

(27) Group broadcast number

This property indicates the group broadcast numbers that have been set for the home node.

The range requirements for group broadcast numbers are as follows:

0x00 to 0x6F shall be used to set group broadcast numbers via networks. 0x70 to 0x7F shall be used to set group broadcast numbers by means of manual setting etc. and not by way of a network. 0x80 to 0xFF are reserved for future use and not used in the current version.

When the number of group broadcast numbers to set is 8 or less, the first byte shall indicate the number of group broadcast numbers to set and the second and succeeding bytes shall list the group numbers.

When the number of group broadcast numbers to set is 9 or more, the first byte shall indicate the number of group broadcast numbers to set and the second and succeeding bytes shall list data in the format explained below. A bit value of 1 shall mean that a group broadcast number has been specified, and a bit value of 0 shall mean that no group broadcast number has been specified.

When a node receives a group broadcast message and the DEA broadcasting target code specifies a group broadcast number that the node has, the node shall deem that the message is addressed to it and start the processing.

It must be possible to set at least 8 group broadcast numbers. When the value is the initial value or when

no value has been set, the property value shall be 0x00.

	Bit 0	Bit1	Bit2	Bit3	Bit4	Bit5	Bit7	Bit7
1st byte	00	01	02	03	04	05	06	07
2nd byte	08	09	0A	0B	0C	0D	0E	0F
3rd byte	10	11	12	13	14	15	16	17
4th byte	18	19	1A	1B	1C	1D	1E	1F
5th byte	20	21	22	23	24	25	26	27
6th byte	28	29	2A	2B	2C	2D	2E	2F
7th byte	30	31	32	33	34	35	36	37
8th byte	38	39	3A	3B	3C	3D	3E	3F
9th byte	40	41	42	43	44	45	46	47
10th byte	48	49	4A	4B	4C	4D	4E	4F
11th byte	50	51	52	53	54	55	56	57
12th byte	58	59	5A	5B	5C	5D	5E	5F
13th byte	60	61	62	63	64	65	66	67
14th byte	68	69	6A	6B	6C	6D	6E	6F
15th byte	70	71	72	73	74	75	76	77
16th byte	78	79	7A	7B	7C	7D	7E	7F
17th byte	80	81	82	83	84	85	86	87
18th byte	88	89	8A	8B	8C	8D	8E	8F
19th byte	90	91	92	93	94	95	96	97
20th byte	98	99	9A	9B	9C	9D	9E	9F
21st byte	A0	A1	A2	A3	A4	A5	A6	A7
22nd byte	A8	A9	AA	AB	AC	AD	AE	AF
23rd byte	B0	B1	B2	B3	B4	B5	B6	B7
24th byte	B8	B9	BA	BB	BC	BD	BE	BF
25th byte	C0	C1	C2	C3	C4	C5	C6	C7
26th byte	C8	C9	CA	CB	CC	CD	CE	CF
27th byte	D0	D1	D2	D3	D4	D5	D6	D7
28th byte	D8	D9	DA	DB	DC	DD	DE	DF
29th byte	E0	E1	E2	E3	E4	E5	E6	E7
30th byte	E8	E9	EA	EB	EC	ED	EE	EF
31st byte	F0	F1	F2	F3	F4	F5	F6	F7
32nd byte	F8	F9	FA	FB	FC	FD	FE	FF

[Note] Bit value = 0: No group broadcast number has been specified.
Bit value = 1: A group broadcast number has been specified.

9.11.2 Router Profile Class: Detailed Specifications

Class group code: 0x0E

Class code: 0xF1

Instance code: 0x01

Property Name	EPC	Contents of Property Value range (decimal notation)	Data Type	Data Size (Bytes)	Access Rule	Mandatory	Announce Status Change	Remarks
Operating status	0x80	Shows operating status of Net ID server or router functions. Booting = 0x31, not booting = 0x31	unsigned char	1	Set Get	 ○	○	(1)
Fault content	0x89	Fault content 0x0000–0x000F (0–15)	unsigned short	2	Get			(2)
Current router data	0xE0	Current router data 1st byte: Router attribute 2nd byte: Current router ID (default 0x00) 3rd byte: Number of connected networks 4th and subsequent bytes: EA data (2 bytes each; for all connected nets)	unsigned char	Max 17	Set/Get	○		(3)
Net ID	0x E1	1-byte Net ID value Initial value = 0x00	unsigned char	1	Set/Get	○		(4)
Router ID	0x E2	1-byte router ID value Initial value = 0x00	unsigned char	1	Set/Get	○		(5)
Parent router data	0x E3	Parent router EA value Initial value = 0x0000 (no parent router data)	unsigned char	2	Set/Get	○		
All router data	0x E4	All router data for domain See Supplement 3	unsigned char	Max 246	Set/Get	○		
Registration request router data	0x E5	Registration router data Byte 1: Router Attribute Byte 2: Current router ID (default 0x00) Byte 3: No. of connected networks Byte 4 and higher: EA data (2 bytes each; one for each connected network)	unsigned char	Max 17	Set	○		(6)
Master router data	0xE6	Master router data value 1st byte: Master router identifier (master router = 0x41, slave router = 0x42) 2nd byte: Net ID information	unsigned char	2	Get	○		(7)

Note) A "o" in the status change announcement column means that the processing is mandatory when the property is implemented.

Note) When there are two or more nodes in a piece of equipment, this profile class shall be held for each node, but it is necessary to hold a common value for the equipment with regard to properties other than the NetID (0xE1), error (0x88), error description (0x89) and master router data (0xE6) properties.

Note) The NetID access rule "Get" shall be permitted for master routers only.

(1) Operation status

This profile class is present if the node has the router functions. This property indicates whether or not the router functions are enabled (i.e. the node is functioning as a router).

(2) Error description

0x0000: There is no error.

0x0001: There is no NetID server.

(None of the subnets connected have been assigned with a NetID and there is no NetID server.)

0x0002: Failed to acquire data from the NetID server.

(A NetID server has been detected in the domain but an attempt toacquire the router data and the data on all outers from the NetID server has failed.)

0x0003: Subnet communication error.

(Cannot communicate with one of the two or more subnets connected.)

0x0004 to 0x000F: This is an area open to users.

0x0010: Failed to start up as a router.

The node has failed to start up as a router for some reason and is operating as an ordinary node.

0x0011 to 0xFFFF: Reserved for future use.

(3) Home router data

The first byte indicates the home router ID. Home router IDs are unique values used to identify the routers in the domain as unique entities. In the case of automatic setting, the NetID server assigns home router IDs in such a manner that the routers in the domain can be identified as unique entities. A node shall not act as a router unless a value other than 0x00 is specified by the NetID server. The router functions can be disabled by writing 0x00.

The second byte indicates the number of networks connected. The maximum allowable number is 7. The specifications for a router connected to 8 or more networks will be provided in future versions of the ECHONET Specifications (Such a case shall be treated as a change in the router property).

The third byte and the succeeding bytes shall store the EA data for all of the connected networks.

In the case of a node equipped with a NetID server function that does not perform received message routing processing, the Set access rule shall not be implemented, but the Get access rule may be implemented.

(4) NetID

Stores the NetID acquired from the master router or NetID server during a router startup sequence. Because a router spans two or more subnets, it has a router profile object in each subnet. The value of this property as seen from the subnet side differs among subnets.

Only the master router shall respond with a NetID to a read request. No slave router shall send a response to such a request including a rejection response.

In the case of a node equipped with a NetID server function that does not perform received message routing processing, the Set access rule shall not be implemented and the Get access rule shall be implemented.

(5) Router ID

Router IDs are identifiers that are assigned by the NetID server to identify routers as unique entities. Router IDs shall be centrally managed by the NetID server. The router ID value is the same as the value of the second byte of the home router data.

In the case of a node equipped with a NetID server function that does not perform received message routing processing, the router ID value shall be fixed at 0xFF and shall only be used by NetID servers that do not perform received message routing processing.

(6) NetID server data

Contains the ECHONET address of the node that has the NetID server functions. In the case of an ECHONET router, the NetID server data of the master router is read during a startup.

In the case of a node equipped with a NetID server function that does not perform received message routing processing, the Set access rule shall not be implemented and the Get access rule shall be implemented.

(7) Data on all routers

This data comprises the total number of router devices in the domain and the home router data of the individual router devices. In the case of an automatic-setting router, the NetID server writes this data. In the case of a manual-setting router, the data on all routers is read from the NetID server when there is a NetID server and from another manual-setting router when there is no NetID server (the device's data on all routers is added to update the acquired data).

In the case of a node equipped with a NetID server function that does not perform received message routing processing, the Set access rule shall not be implemented and the Get access rule shall be implemented.

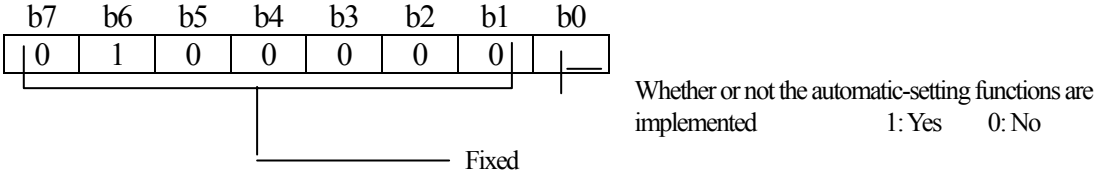
(8) Master router data

This data is used in the ECHONET router startup sequence. Because a router spans two or more subnets, it has a router profile object in each subnet. The value of this property as seen from the subnet side differs among subnets.

(9) Router property

Indicates the router property. This property is defined based on whether or not the automatic-setting functions are implemented (i.e. whether the automatic- or manual-setting router startup sequence is to be performed (see Chapter 5).

In the case of a node equipped with a NetID server function that does not perform received message routing processing, the Get access rule may be implemented as an optional access rule.



(10) Access point functions

If the access point functions are implemented, routing is performed when a received message is an individual designation message and the destination EA does not match the home EA. The routing processing shall also be performed if the sender NetID is the same as the destination NetID.

If the access point functions are not implemented, the received message is discarded when "a received message is an individual designation message, the destination EA does not match the home EA, and the destination NetID matches the NetID of the home EA." The routing processing shall not be performed when the sender's NetID is the same as the destination NetID.

9.11.3 Detailed Requirements for the NetID Server Profile Class

Class group code: 0x0E

Class code: 0xF5

Instance code: 0x01

Property Name	EPC	Content of the Property Value Range (Decimal notation)	Data Type	Size (Bytes)	Access Rule	Mandatory	Announce Status Change	Remarks
Operation status	0x80	Operation status (NetID server functions) Enabled = 0x30 Disabled = 0x31	unsigned char	1	Set/Get	○	○	(1)
Error description	0x89	Error description 0x0000 to 0x0010 (0 to 16)	unsigned short	2	Get			(2)
Data on all routers	0xE4	Data on all routers in the domain See Appendix 3.	unsigned char x (Max. 246)	Max. 246	Get	○		(3)
Registration request router data	0xE5	Registration request router data 1st byte: router property 2nd byte: home router ID (default = 0x00) 3rd byte: the number of networks connected 4th byte and the succeeding bytes: EA data (for all networks connected; 2 bytes each)	unsigned char x (Max. 17)	Max. 17	Set	○		(4)

ECHONET router registration completion notification	0xE7	ECHONET router registration completion notification 0x00	unsigned char	1	Get	○		(5)
Router registration status	0xE8	Indicates the ECHONET router registration status Router registration busy = 0x30 Router registration ready = 0x31	unsigned char	1	Get	○		(6)

Note) A "o" in the status change announcement column indicates that the processing is mandatory when the property is implemented.

Note) When there are 2 or more nodes in a single piece of equipment, there shall be only one node with the NetID server functions.

(1) Operation status

This profile class is present if the node has the NetID server functions. This property indicates whether or not the NetID server functions are enabled.

(2) Error description

0x0000: There is no error.

0x0001 to 0x0002: Reserved for future use.

0x0003: Subnet communication error.

(Cannot communicate with one of the two or more subnets connected.)

0x0004 to 0x000F: Area open to users.

0x0010: Failed to start up as a router.

The node has failed to start up as a router for some reason and is operating as an ordinary node.

0x0011 to 0xFFFF: Reserved for future use.

(3) Data on all routers

This data comprises the total number of router devices in the domain and the home router data of the individual router devices. The data on all routers is updated by collecting the home router data every time an ECHONET router starts up.

The values must be the same as those of the router profile class of the same device.

(4) Registration request router data

During the router startup sequence, the ECHONET router notifies its EA data to the NetID server profile class by writing the EAs of all nodes that comprise the ECHONET router ("all EA data") into this property.

During the ECHONET router startup sequence, the NetID server profile class shall make write requests to the "home router data" and "data on all routers" properties of the ECHONET router's router profile

based on the setting (request) made by the ECHONET router using this property.

(5) ECHONET router registration completion notification

If the NetID server profile class receives a registration request router data write request from an ECHONET router, it shall send a home router data write request and a "data on all routers" write request, and, after receiving the response to the "data on all routers" write request, send an ECHONET router registration completion notification to the ECHONET router as a notification that requires a response.

(6) Router registration status

Indicates whether or not the processing to register an ECHONET router is being performed.

9.11.4 ECHONET Communications Processing Block Profile Class: Detailed Specifications

Class group code: 0x0E

Class code: 0xF2

Instance code: 0x01

Property Name	EPC	Contents of Property Value range (decimal notation)	Data Type	Data Size (Bytes)	Access Rule	Mandatory	Announce Status Change	Remarks
Operating status	0x80	Shows operating status of ECHONET Communications Processing Block functions Booting = 0x31, not booting = 0x31	unsigned char	1	Set/Get			(1)
Fault content	0x89	Fault content 0x0000–0x000F (0–15)	unsigned short	2	Get			(2)
Transition state	0x 8E	Shows software operation transition state 0x0000–0x000F(0–15)	unsigned short	2	Get			(3)
Version data	0xB8	Version number of ECHONET Communications Processing Block (3 bytes binary)	unsigned char	3	Get			(4)
Buffer size data	0xB9	Buffer size data (max bytes) (1 byte binary) 6–256 Note: The value 256 is indicated by 0x00.	unsigned char	1	Get			(5)

(1) Operating status

Indicates the operating status of communications middleware. Used primarily to confirm the operating status and switch the communications middleware ON/OFF from the application software.

(2) Fault content

0x0000: No fault

0x0001: Exchange with application software inactive

(Indicates the state in which messages cannot be passed to the application software via basic API. Determined by timeout, etc. In this case, only reading from other nodes, including this property, is possible. Settings for this fault code will not be specified.)

0x0002: Exchange with Protocol Difference Absorption Processing Block and below inactive.

(Indicates the state in which messages cannot be passed to the Protocol Difference Absorption Processing Block via the common lower-layer communication interface. Determined by timeout, etc. In this case, only reading from application software, including this property, is possible. Settings for this fault code will not be specified.)

0x0003–0x0004: (unused)

0x0005–0x000F: Open to users

0x0010–0xFFFF: Reserved for future use

(3) Transition states

Transition states for ECHONET Communications Processing Block software are shown below. Specific code assignments are as follows:

0x0000 : Currently halted

0x0001 : Initializing

0x0002 : Normal operation

0x0003 : Normal operation by application error hold

0x0004 : Normal operation by Protocol Difference Absorption Processing Block error hold

0x0005 : Normal operation by lower-layer communications software error hold

0x0006 : Normal operation by lower-layer communications driver (hardware) error hold

0x0007 : Application error hold halt

0x0008 : Protocol Difference Absorption Processing Block error hold halt

0x0009 : Lower-layer communications software error hold halt

0x000A : Lower-layer communications driver (hardware) error hold halt

0x000B : ECHONET Communications Processing Block error hold halt

0x000C–0x000F : Open to users

0x0010–0xFFFF : Reserved for future use

State priorities for 0x0003–0x0006 and 0x0007–0x000B are as follows:

0x0006>0x0005>0x0004>0x0003

(For example, when the system is operating with both an application error and a lower-layer communications software error held, it is in normal operation by a lower-layer communications software error hold state.)

0x000B>0x000A>0x0009>0x0008>0x0007

(For example, when the system is halted with both an application error and a lower-layer communications software error held, it is in a lower-layer communications software error hold halt state.)

(4) Version data

Uses a 3-byte binary value to present version information for the ECHONET Communications Processing Block software. No specific value is stipulated.

(5) Buffer size data

Maximum EDATA size that can be processed by the ECHONET Communications Processing Block.

9.11.5 Protocol Difference Absorption Processing Block Profile Class: Detailed Specifications

Class group code: 0x0E

Class code: 0xF3

Instance code: 0x01

Property Name	EPC	Contents of Property Value range (decimal notation)	Data Type	Data Size (Bytes)	Access Rule	Mandatory	Announce Status Change	Remarks
Operating status	0x80	Shows operating status of Protocol Difference Absorption Processing Block functions Booting = 0x31, not booting = 0x31	unsigned char	1	Set/Get			(1)
Fault content	0x89	Fault content 0x0000–0x03E8 (0–1000)	unsigned short	2	Get			(2)
Transition state	0x 8E	Shows software operation transition state 0x0000–0x03E8 (0–1000)	unsigned short	2	Get			(3)
Version data	0xB8	Version number of Protocol Difference Absorption Processing Block (3 bytes binary)	unsigned char	3	Get			(4)
Buffer size data	0xB9	Buffer size data (max bytes) (1 byte binary) 6–256 Note: The value 256 is indicated by 0x00.	unsigned char	1	Get			(5)

(1) Operating status

Shows the operating status of the Protocol Difference Absorption Processing Block.

(2) Fault content

0x0000 : No fault.

0x0001–0x0002 : (Unused)

0x0003 : Exchange with lower-layer communications software inactive

(Indicates the state in which messages cannot be passed to the Protocol Difference Absorption Processing Block via the individual lower-layer communications interface. Determined by

timeout, etc. In this case, only reading from application software, including this property, is possible. Settings for this fault code will not be specified.)

0x0004 : (Unused)

0x0005–0x000F : Open to users

0x0010–0xFFFF : Reserved for future use

(3) Transition states

Indicates the state in which the Protocol Difference Absorption Processing Block is placed. The following codes are assigned:

0x0000 : Currently halted

0x0001 : Initializing

0x0002 : Normal operation

0x0003–0x0004 : (Unused)

0x0005 : Normal operation by lower-layer communications software error hold

0x0006 : Normal operation by lower-layer communications driver (hardware) error hold

0x0007 : (Unused)

0x0008 : Protocol Difference Absorption Processing Block error hold halt

0x0009 : Lower-layer communications software error hold halt

0x000A : Lower-layer communications driver (hardware) error hold halt

0x000B : (Unused)

0x000C–0x000F : Open to users

0x0010–0xFFFF : Reserved for future use

State priority for 0x0008–0x000A is 0x000A>0x0009>0x0008.

(4) Version data

Uses a 3-byte binary value to present version information for the ECHONET Communications Processing Block software. No specific value is stipulated.

(5) Buffer size data

Maximum EDATA size that the Protocol Difference Absorption Processing Block can process with the Communications Processing Block.

9.11.6 Lower-Layer Communications Software Profile Class: Detailed Specifications

Class group code: 0x0E

Class code: 0xF4

Instance code: 0x01

Property Name	EPC	Contents of Property Value range (decimal notation)	Data Type	Data Size (Bytes)	Access Rule	Mandatory	Announce Status Change	Remarks
Operating status	0x80	Shows operating status of lower-layer communications software functions Booting = 0x31, not booting = 0x31	unsigned char	1	Set/Get			(1)
Fault content	0x89	Fault content 0x0000–0x000F (0–15)	unsigned short	2	Get			(2)
Transition state	0x 8E	Shows software operation transition state 0x0000–0x000F (0–15)	unsigned short	2	Get			(3)
Version data	0xB8	Version number of ECHONET Communications Processing Block (3 bytes binary)	unsigned char	3	Get			
Lower-layer communications software type	0xE0	Stipulates lower-layer communications software type	unsigned char	1	Get			(4)
MAC address data	0xE1	MAC address data Byte 1: MAC address size Bytes 2–8: MAC address	unsigned char	Max. 8	Set/Get			(5)
House code data	0xE2	House code data Byte 1: House code length Bytes 2–9: House code	unsigned char	Max. 9	Set/Get			(6)
Bind interval data	0xE3	Bind interval data 0x0000: Infinite bind interval. 0x0001–0xFFFF (1–65535sec)	unsigned short	2	Set/Get			
Buffer size data	0xB9	Buffer size data (max bytes) (1 byte binary) 6–256 Note: The value 256 is indicated by 0x00.	unsigned char	1	Get			(7)

(1) Operating status

Shows the operating status of the lower-layer communications software.

(2) Fault content

0x0000 : No fault

0x0001–0x0003 : (Unused)

0x0004 : Exchange with lower-layer communications driver inactive

(Indicates the state in which messages cannot be sent over the network via the lower-layer communications driver. Determined by timeout, etc. In this case, only reading from application software, including this property, is possible. Settings for this fault code will not be specified.)

0x0005–0x000F : Open to users

0x0010–0xFFFF : Reserved for future use

(3) Transition states

Transition states for ECHONET communications control processing software are shown below. Specific code assignments are as follows:

0x0000 : Currently halted

0x0001 : Initializing

0x0002 : Normal operation

0x0003–0x0005 : (Unused)

0x0006 : Normal operation by lower-layer communications driver (hardware) error hold

0x0007–0x0009 : (Unused)

0x000A : Lower-layer communications driver (hardware) error hold halt

0x000B : (Unused)

0x000C–0x000F : Open to users

0x0010–0xFFFF : Reserved for future use

(4) Version data

Uses a 3-byte binary value to present version information for the lower-layer communication software. No specific value is stipulated.

(5) Lower-layer communications software type

Shows the lower-layer communications software type. Specific code assignments are as follows:

0x01 = Power Line Communication Protocol "a" and "d" Systems

0x03 = Low-power wireless

0x04 = Expanded HBS

0x05 = IrDA Control

0x06 = LonTalk®

0x07 = IP/Bluetooth®

0x08 = IP/Ethernet, IEEE802.3

0x09 = IEEE802.11/11b

0x0A = Power Line Communication Protocol "c" System

0x00, 0x02, 0x07–0xFF: Reserved for future use

(6) MAC address data

Sets the MAC address starting from Byte 2, with the number of bytes indicated by the value of Byte 1. The MAC address may be up to 7 bytes long.

(7) House code data

The second and subsequent bytes, the number of which is stipulated by the size data placed in the first byte position, set the power line house code or specific low-power wireless system identification code. Up to 8 bytes can be used for setup. For the house code and wireless system identification code, see Part

3.

(8) Bind interval data

Sets the time interval at which a peripheral periodically issues a "bind request" to the host when the lower-layer communication software is IrDA Control. However, the value 0x0000 represents an infinite bind interval, which means that a peripheral does not issue a periodic "bind request" to the host.

(9) Buffer size data

Maximum EDATA size that can be processed by the lower-layer communication software.

9.12 Requirements for the Communication Definition Class Group

The communication definition class groups are 7 class groups which, when there is a function in the communication middleware that performs processing to control/manipulate properties of the base equipment or service object on behalf of the application software, notify the settings for that function to the other nodes connected to the ECHONET network.

The groups are as follows: Status notification method specification communication definition class group, Set control acceptance method specification communication definition class group, Linkage setting (action setting) communication definition class group, Linkage setting (trigger setting) communication definition class group, Local alteration limit setting communication definition class group, Network control limit status indication communication definition class group, Secure communication access property setting class group, Property status referencing request compliability communication definition class group and Property status notifiability communication definition class group.

The class groups contain the following settings for the associated objects (device, service and profile objects) in the same node:

① Settings for the property content notification that is made immediately after a property content change
② Settings for periodic property content notifications
③ Settings relating to nodes that permit property content changes
④ Settings for linked operation with other objects
⑤ Settings for the limits imposed with respect to operation status alterations on the equipment side
⑥ Settings for the limits imposed with respect to operation status alterations by other nodes
⑦ Settings for secure communication
⑧ Status of each of the applicable properties as to whether the property is ready to accept referencing requests from other nodes, and a function that allows other nodes to alter the status setting
⑨ Status of each of the applicable properties as to whether the property is ready to provide notifications, and a function that allows other nodes to alter the status setting (property status

① and ② correspond to status notification method specification communication definition class group, ⑨ corresponds to Set control acceptance method specification communication definition class group, ④

corresponds to linkage setting (action setting) communication definition class group, linkage setting (trigger setting) communication definition class group, ⑤ cooresponds to local alteration limit setting communication definition class group, ⑥ corresponds to network control limit status indication communication definition class group, ⑦ corresponds to secure communication access property setting class group, ⑧ corresponds to property status referencing request compliability communication definition class group, and ⑨ corresponds to property status notifiability communication definition class group.

The association between the objects of these class groups and the objects relevant to the information retained by the former objects are established by ECHONET object codes. The relationship between the ECHONET class codes of objects belonging to the communications definition class groups and the ECHONET object codes of the objects associated with the former objects are as indicated below:

- Agree in the instance code X3.
- Agree in the class code X2.
- Agree in the four low-order bits of the class group code X1.

The four high-order bits of the class group code X1 specify to which one of the four communications definition class groups an object belongs. The value 0001b selects the status notification method requirement communications definition group. The value 0010b selects the Set control reception method requirement communications definition class group. The value 0011b selects the linkage (action) setting communications definition class group. The value 0100b selects the linkage (trigger) setting communications definition class group.

The 4 highest-order bits of the class group code X1 specify the communication definition class group to which the object belongs. 0001b specifies the status notification method specification communication definition class group, 0010b specifies the Set control acceptance method specification communication definition class group, 0011b specifies the linkage setting (action setting) communication definition class group, 0100b specifies the linkage setting (trigger setting) communication definition class group, 0110b specifies the local alteration limit setting communication definition class group, 0111b specifies the network control limit status indication communication definition class group and 0101b specifies the secure communication access property setting class group.

The settings are retained in the unit of a property. The communications definition setting for a certain property is retained by an associated communications definition object property having the same ECHONET property code.

Note that the objects belonging to the communications definition class groups do not have to be mounted at all times. They can be implemented as needed.

This chapter stipulates the details of the property configurations commonly applicable to the communications definition class groups as the communications definition object super class.

9.12.1 Overview of Communications Definition Object Super Class Specifications

The communications definition object super class properties are inherited and implemented by each class of the four communications definition class groups. These properties are summarized in Table 9.6.

Table 9.6 List of Communications Definition Object Super Class Configuration Properties

Property Name	EPC	Contents of Property Value range (decimal notation)	Data Type	Data Size (Byte)	Access Rule	Mandatory	Announce Status Change	Remarks
State change announce properties map	0x9D	See Supplement 2	unsigned char	Max. 17	Get	○		
Set properties map	0x9E	See Supplement 2	unsigned char	Max. 17	Get	○		
Get properties map	0x9F	See Supplement 2	unsigned char	Max. 17	Get	○		

Note: A "o" in the status change announcement column denotes mandatory processing when the property is implemented.

9.12.2 Property Map

Three property map properties defined for the communications definition object super class retain the list of properties whose value changes will be subjected to a general broadcast, the list of properties that can be edited from remote nodes, and the list of properties that can be referenced by remote nodes. The individual property map property configurations are the same as those defined for the device object super class (see Section 9.3.11).

9.13 Specifications for Status Notification Method Requirement Communications Definition Class Group

This section provides detailed specifications for ECHONET objects that belong to the status notification method requirement communications definition class group (class group code X1 = 0x10 to 0x1F). Objects belonging to this class group retain the following status notification method settings for all the properties of the associated objects (objects existing in the same node and having the same class group code X1 four low-order bits, class code X2, and instance code X3). When a status notification method is stipulated for an object by the status notification method requirement communications definition object, the former object must report the associated property value by the stipulated method.

- Setting data for property content notification at the time of a property content change (including setting data for the party being communicated with)
- Setting data for periodic property content notification (including the communication cycle and the other party being communicated with)

The above settings are retained in the unit of a property. The status notification method for a certain property is retained by an associated status notification method requirement object property having the same ECHONET property code. The property to be targeted for status notification method requirement is not specified because it is a matter of system design. (Properties to which the above status notification method is inapplicable cannot exist as a status notification method requirement object property. Furthermore, the status notification method cannot be stipulated for the property map properties (0x9B to 0x9F)). There is no need to set a status notification method for all the properties of associated objects.

The "fire sensor class status notification method requirement communications definition object" is described on the next page as an example for giving the details of properties retained by the objects of the status notification method requirement communications definition class group. The fire sensor class has a class group code X1 of 0x00 and a class code X2 of 0x19. Therefore, the associated status notification method requirement object has a class group code X1 of 0x10, a class code X2 of 0x19, and the same instance code X3 value as fire sensor object X3.

Note that this Version does not support the selection of a secure communication method.

Detailed Specifications for Fire Sensor Class Status Notification Method Requirement Communications Definition Object

Class group code: 0x10

Class code: 0x19

Instance code: 0x01–0x7F (0x00: all instance requirement code)

Property Name	EPC	Contents of Property Value range (decimal notation)	Data Type	Data Size (Bytes)	Access Rule	Mandatory	Announce Status Change	Remarks
Communications definition for operating status	0x80	Communications definition data for status notification method requirement (see below)	unsigned char	6	Set/ Get			
Communications definition for detection threshold level	0xB0	Communications definition data for status notification method requirement (see below)	unsigned char	6	Set/ Get			
Communications definition for fire detection status	0xB1	Communications definition data for status notification method requirement (see below)	unsigned char	6	Set/ Get			
Communications definition for malfunction status	0x88	Communications definition data for status notification method requirement (see below)	unsigned char	6	Set/ Get			
Communications definition for malfunction content	0x89	Communications definition data for status notification method requirement (see below)	unsigned char	6	Set/ Get			

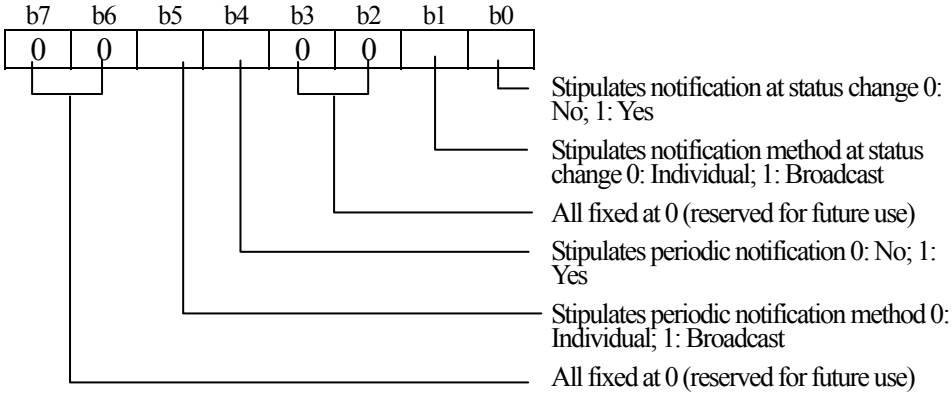
The size of each property is 6 bytes. The property configuration is shown below:

1-byte	2-byte	2-byte	1-byte
Notification trigger data	Status change notification destination EA data	Periodic notification destination EA data	Notification interval data

Details are given below:

① Notification trigger data

Uses a 1-byte bitmap to retain information about an associated object property having the same EPC code. The information retained in this manner indicates whether or not to send a notification in the event of a status change, the method of such status change notification, whether or not to send a periodic notification, and the method of such periodic notification. The figure below shows the meanings of the individual bits:



② Status change notification destination EA data

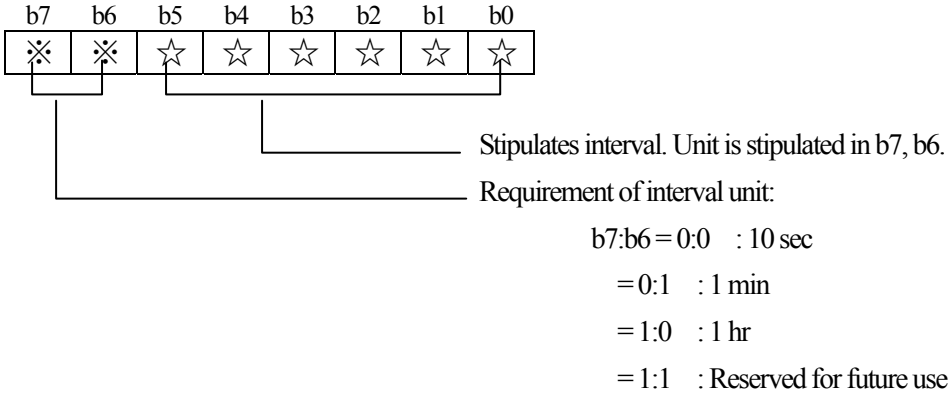
When bit b0 of the notification trigger data indicates that a status change notification will be made, two bytes are used to retain the ECHONET address of the notification destination or a set of the broadcast type requirement code and broadcast target requirement code. When bit b1 of the notification trigger data is "individual", the ECHONET address of the notification destination is retained. When bit b1 of the notification trigger data is "broadcast", the set of broadcast type and broadcast target requirement codes is retained.

③ Periodic notification destination EA data

When bit b4 of the notification trigger data indicates that a periodic notification will be made, two bytes are used to retain the ECHONET address of the notification destination or a set of the broadcast type requirement code and broadcast target requirement code. When bit b5 of the notification trigger data is "individual", the ECHONET address of the notification destination is retained. When bit b5 of the notification trigger data is "broadcast", the set of broadcast type and broadcast target requirement codes is retained.

④ Notification interval data

When bit b4 of the notification trigger data indicates that a periodic notification will be made, one byte is used to retain the periodic notification interval data. The data is retained in the following format:



The combination of bits b6 and b7 indicates the unit of notification interval value. Three different units are selectable: 10 seconds, 1 minute, and 1 hour. Bits b0 to b5 indicate the coefficient for the selected unit value. If, for instance, b7:b6 = 0:1 and bits b0 to b5 are 000101b (notification interval data = 0x45), the notification will be made at 5-minute intervals.

9.14 Requirements for the Set Control Acceptance Method Specification Communication Definition Class Group

This section provides detailed specifications for ECHONET objects belonging to the Set control reception method requirement communications definition class group (class group code X1 = 0x20 to 0x2F). The objects belonging to this class group retain the following Set control (property value rewrite) reception method settings for all the properties of the associated objects (objects existing in the same node and equal in the class group code X1 four low-order bits, class code X2, and instance code X3). When a remote party whose Set control is rendered acceptable is stipulated for an object by the Set control reception method requirement communications definition object, the former object must not accept the request for a change in an associated property value if the request is not issued from the stipulated party.

- Maximum registration number of nodes that can be allowed to perform a property rewrite (10 nodes maximum).
- Number of nodes that are allowed to perform a rewrite
- ECHONET addresses of nodes that are allowed to perform a rewrite

The above settings are retained in the unit of a property. The Set control reception method for a certain property is retained by an associated Set control reception method requirement object property having the same ECHONET property code. The property to be targeted for the Set control reception method requirement is not specified because it is a matter of system design. (Properties for which it is allowed to "Get" only cannot exist as a Set control reception method requirement object property. Furthermore, the Set control reception method cannot be stipulated for the property map properties (0x9B to 0x9F)). There is no need to set a Set control reception method for all the properties of associated objects.

The "fire sensor class Set control reception method requirement communications definition object" is

described on the next page as an example for giving the details of properties retained by the objects of the Set control reception method requirement communications definition class group. The fire sensor class has a class group code X1 of 0x00 and a class code X2 of 0x19. Therefore, the associated Set control reception method requirement object has a class group code X1 of 0x20, a class code X2 of 0x19, and the same instance code X3 value as fire sensor object X3.

Fire Sensor Class Communications Definition Objects for Set Control Reception Method Requirement: Detailed Specifications

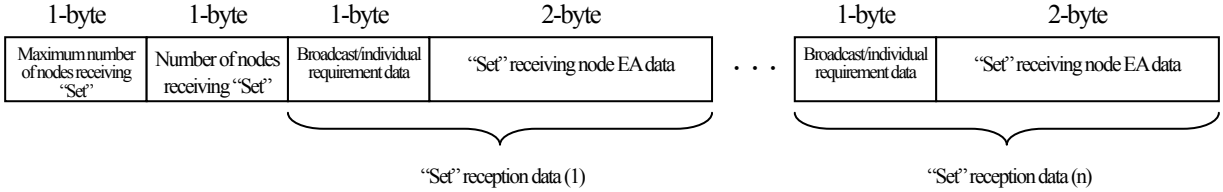
Class group code: 0x20

Class code: 0x19

Instance code: 0x01–0x7F (0x00: all instance requirement code)

Property Name	EPC	Contents of Property Value range (decimal notation)	Data Type	Data Size (Bytes)	Access Rule	Mandatory	Announce Status Change	Remarks
Communications definition for operating status	0x80	Communications definition data for set control reception method requirement (see below)	unsigned char	Max32	Set/ Get			
Communications definition for detection threshold level	0xB0	Communications definition data for set control reception method requirement (see below)	unsigned char	Max 32	Set/ Get			
Communications definition for fire detection status	0xB1	Communications definition data for set control reception method requirement (see below)	unsigned char	Max 32	Set/ Get			
Communications definition for malfunction status	0x88	Communications definition data for set control reception method requirement (see below)	unsigned char	Max 32	Set/ Get			
Communications definition for malfunction content	0x89	Communications definition data for set control reception method requirement (see below)	unsigned char	Max 32	Set/ Get			

The size of each property is variable up to 32 bytes. The property configuration is shown below:



Details are given below:

① Maximum number of "Set" receiving nodes

This is the first byte of the property. The size is 1 byte. It retains the maximum number of nodes that can be allowed to "Set" in relation to associated object properties having the same EPC code. For this data, only

referencing is valid; a write to this data can be ignored. The acceptable setting ranges from 1 to 10.

② Number of nodes receiving "Set"

This is the second byte of the property. The size is 1 byte. It retains the number of nodes that are allowed to "Set" in relation to associated object properties having the same EPC code.

③ Set reception data

This data consists of the third and subsequent bytes. Its size is 3 bytes. It retains information about nodes that are allowed to "Set" in relation to associated object properties having the same EPC code. Contiguous Set reception data can be retained. The maximum number of contiguously retained Set reception data is equal to the value that is retained as the maximum number of "Set" receiving nodes. Furthermore, the actually retained number indicates the number of "Set" receiving nodes.

The Set reception data consists of two elements: broadcast/individual requirement data and Set receiving node EA data. The size of broadcast/individual requirement data is 1 byte. This data retains the information that indicates whether or not the rewrite request message from a node designated by the Set receiving node EA data retained in the same Set reception data permits a rewrite only when an individual address is selected, permits a rewrite only when a broadcast address is selected, or permits a rewrite regardless of whether an individual or broadcast address is selected. The relationship between permissions and property values is as indicated below:

0x41: Permitted only when an individual address is selected.
0x42: Permitted only when a broadcast address is selected.
0x43: Permitted no matter which address is selected.

The Set receiving node EA data consists of 2 bytes. It retains the ECHONET address of a node that is allowed to "Set".

9.15 Specifications for Linkage (Action) Setting Communications Definition Class Group

This section provides detailed specifications for ECHONET objects that belong to the linkage (action) setting communications definition class group (X1 = 0x30 to 0x3F). The objects belonging to this class group must transmit a specified message (this message transmission operation is called an action) when specified conditions are met by the associated property value of an associated object (object existing in the same node and equal in the class group code X1 four low-order bits, class code X2, and instance code X3). The objects (action setting target object) belonging to this class group retain the conditions (linked startup conditions and linked startup condition values) specified for a property value (action setting target property) and the configuration of an outgoing message (action message configuration data). When the conditions specified for a property value are met, it means the following:

Specified condition for cases in which the action setting target property is a non-array property

The condition for a linked startup must be satisfied between the value of the action setting target property of the action setting target object and the linked startup condition value.

Array properties can also be targeted. When the conditions specified for an array property value are met, it means the following:

Specified condition for cases in which the action setting target property is an array property

The condition for a linked startup must be satisfied between the linked startup condition value and the values of all array elements that are specified by the pairs of array element number mask values and masked array element numbers in the action setting target property of the action setting target object.

The following procedure is performed for array element selection based on the array element number mask value and masked array element number:

Array element selection method

The array element numbers of the associated property are ANDed with the array element number mask value. The values derived from ANDing are compared against the masked array element number. The linked startup conditions are then applied to the match.

The linked startup conditions are roughly divided into the following six types:

- The property value is equal to the linked startup condition value.
- The property value is greater than the linked startup condition value.
- The property value is less than the linked startup condition value.
- The property value is greater than or equal to the linked startup condition value.
- The property value is less than or equal to the linked startup condition value.
- The property value is not equal to the linked startup condition value.

When the linked startup conditions are met, a message in a designated configuration must be transmitted. The outgoing message configuration is retained in the outgoing message configuration data on an individual property basis.

The “fire sensor class linkage (action) setting communications definition object” is described on the next page as an example for giving the details of properties retained by the objects of the linkage (action) setting communications definition class group. The fire sensor class has a class group code X1 of 0x00 and a class code X2 of 0x19. Therefore, the associated linkage (action) setting communications definition object has a class group code X1 of 0x30, a class code X2 of 0x19, and the same instance code X3 value as fire sensor object X3.

Note that this Version does not support the transmission of secure messages and compound messages.

In principle, ECHONET objects that belong to this class group must have a nonvolatile memory.

Part X, Section 3.12 specifies the method for building network systems using linkage setting object-based

linkage settings.

Fire Sensor Class Communications Definition Objects for Linked (Action) Settings: Detailed Specifications

Class group code: 0x30

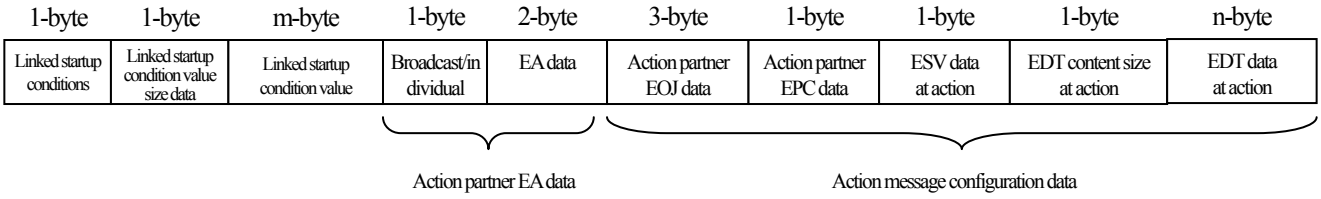
Class code: 0x19

Instance code: 0x01–0x7F (0x00: all instance requirement code)

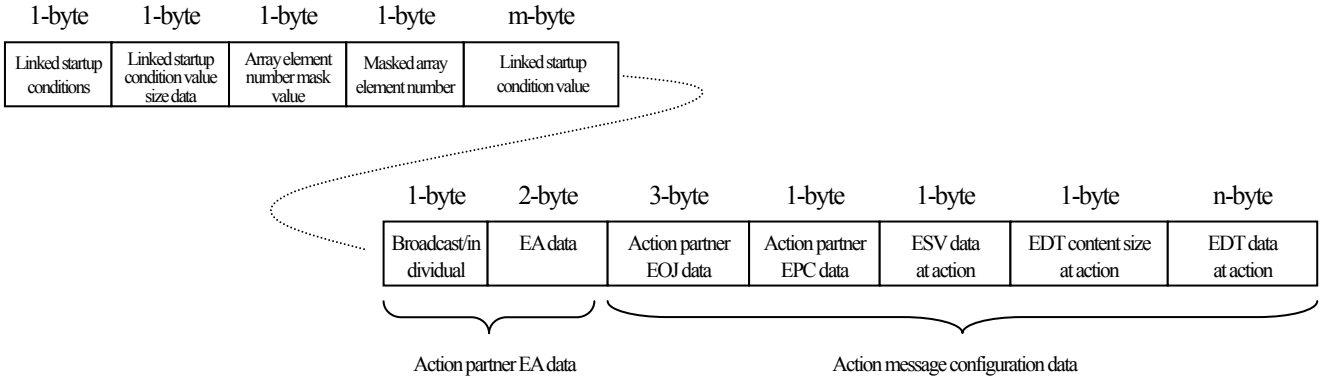
Property Name	EPC	Contents of Property Value range (decimal notation)	Data Type	Data Size (Byte)	Access Rule	Mandatory	Announce Status Change	Remarks
Communications definition for operating status	0x80	Communications definition data for linked setting (action setting) (see below)	unsigned char	Max 247	Set/ Get			
Communications definition for detection threshold level	0xB0	Communications definition data for linked setting (action setting) (see below)	unsigned char	Max 247	Set/ Get			
Communications definition for fire detection status	0xB1	Communications definition data for linked setting (action setting) (see below)	unsigned char	Max 247	Set/ Get			
Communications definition for malfunction status	0x88	Communications definition data for linked setting (action setting) (see below)	unsigned char	Max 247	Set/ Get			
Communications definition for malfunction content	0x89	Communications definition data for linked setting (action setting) (see below)	unsigned char	Max 247	Set/ Get			

The size of each property is variable up to 245 bytes. The property configuration is shown below:

When the associated property is of a non-array type



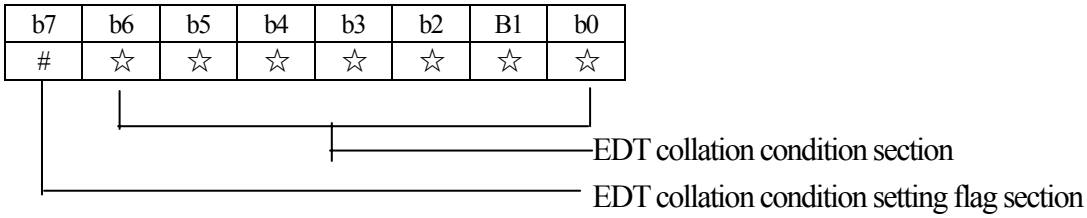
When the associated property is of an array type



The details of the individual configuration elements are given below:

① Linked startup conditions

1-byte information indicating the relationship between the value of the associated property of the associated object and the linked startup condition value. The linkage (action) setting communications definition object monitors the value of the associated property of the associated object or the property's internal array element stipulated by a set of an array element number mask value and masked array element number. When the linked startup conditions are satisfied by the monitored value and linked startup condition value, a message transmission takes place in accordance with the action message configuration data. The following codes are assigned to the linked startup conditions:



Linked startup condition setting flag section (b7)

If b7 = 0, transmits the message that is created based on the action message composition data when the condition is satisfied (linked startup condition (b6 to b0)). When b7 is 1, no message is transmitted regardless of the linked startup condition.

0x00: No linkage.
0x01: The property value is equal to the linked startup condition value. (=)
0x02: The property value is greater than the linked startup condition value. (>)
0x03: The property value is less than the linked startup condition value. (<)

0x04: The property value is greater than or equal to the linked startup condition value. (≧)

0x05: The property value is less than or equal to the linked startup condition value. (≦)

0x06: The property value is not equal to the linked startup condition value. (≠)

② Linked startup condition value size data

When the action setting target property is a non-array type property, this shall be the size of the linked startup condition value. When the action setting target property is an array type property, this shall be the sum of the sizes of the array element number mask value, masked array element number and linked startup condition value.

③ Array element number mask value

This value is present when the action setting target property is an array type property. This value is always used in combination with the masked array element number. For details, see "④ Masked array element number" below.

④ Masked array element number

This value is present when the action setting target property is an array type property. This value is always used in combination with the array element number mask value, to determine which array element of the action setting target property should be monitored. The array element numbers of the action setting target property are ANDed with the array element number mask value and the values derived from the ANDing are compared with the masked array element number, with the matched ones treated as the subjects of the monitoring and of the linked startup condition.

⑤ Linked startup condition value

The value of the monitored property or the value compared against a stipulated array element value under the linked startup conditions. Its size is indicated by the linked startup condition size data.

⑥ Action partner EA data

3-byte information for stipulating the destination node to which an action message is to be transmitted when the linked startup conditions are met. It consists of 1-byte broadcast/individual requirement data and 2-byte EA data.

- Broadcast/individual requirement data

  This data specifies whether the ECHONET address presented by the EA data is a broadcast address or individual address. The value to be taken is 0x42 for a broadcast address or 0x41 for an individual address.
- EA data

  This data indicates the ECHONET address of a node that is to be requested to perform an action.

  When a broadcast address is selected by the broadcast/individual requirement data, the broadcast type requirement code is placed in the first byte position with the broadcast target

requirement code in the second byte position. The codes to be used and their meanings must comply with the specifications set forth in Section 4.2.2 “Source/Destination ECHONET Address (SEA/DEA)”.

When an individual address is selected by the broadcast/individual requirement data, the Net ID is placed in the first byte position with the Node ID in the second byte position.

⑦ Action partner EOJ data

3-byte information for stipulating the destination ECHONET object to which an action message is to be transmitted when the linked startup conditions are met. The class group code X1 is placed in the first byte position, with the class code X2 in the second byte position and the instance code X3 in the third byte position. The codes to be used and their meanings must comply with the specifications set forth in Section 4.2.6 “ECHONET Objects (EOJ)”.

⑧ Action partner EPC data

1-byte information for stipulating the ECHONET property on which the action message operates. The code to be used and its meaning must comply with the specifications for the target object.

⑨ ESV data at action

1-byte information for stipulating a code that defines the operation request to be issued to the property stipulated by the action partner EPC data that is owned by the ECHONET object stipulated by the action partner EA data and action partner EOJ data. The code to be used and its meaning must comply with the specifications set forth in Section 4.2.8 “ECHONET Service (ESV)”.

⑩ EDT content size at action

When the operation stipulated by the ESV data at action is a write or an operation on an array, the content to be written or the target array element number is required as an EDT. The EDT size data at action is 1-byte information for indicating the number of bytes used for the EDT data at action, which indicates the EDT. When the ESV data at action indicates an ESV that does not require an EDT, the value 0x00 must be taken.

⑪ EDT data at action

Being variable in length, this data indicates the contents of the EDT. The size of this data is indicated by the EDT content size at action.

9.16 Specifications for Linkage (Trigger) Setting Communications Definition Class Group

This section provides detailed specifications for ECHONET objects belonging to the linkage (trigger) setting communications definition class group (X1 = 0x40 to 0x4F). The objects belonging to this class group must rewrite the contents of a stipulated property of an associated object (object existing in the same node and equal in the class group code X1 four low-order bits, class code X2, and instance code X3) with a stipulated value when a message received by the communication middleware is found to be in compliance

with stipulated conditions.

The linkage (trigger) setting communications definition objects function in relation to a property having the same ECHONET property code as the object property targeted for linkage setup. They retain the configuration of a trigger message (trigger message configuration data) and the information about the values (property setting size and property setting) to be written into the associated property when the linkage trigger conditions are met. The trigger message configuration data consists of the following elements:

- Partner EA data
- EA mask data
- Partner EOJ data
- Instance code mask data
- EPC data
- ESV data
- EDT size data
- EDT content
- EDT comparison data

The linkage trigger conditions are met when a message satisfying all the following conditions is received:

Condition 1:

The source ECHONET address (SEA) in the message must match the partner EA data. However, the use of the EA mask data makes it possible to register two or more source ECHONET addresses as the match target.

Condition 2:

The source ECHONET object (SDEOJ) in the message must match the partner EOJ data. However, the use of the instance code mask data makes it possible to register two or more source ECHONET objects of the same type as the match target.

Condition 3:

The ECHONET property (EPC) in the message must match the EPC data.

Condition 4:

The ECHONET service (ESV) in the message must match the ESV data.

Condition 5:

The relationship between the EDT value and EDT data in the message must comply with the EDT comparison data. For an ECHONET property code whose EPC data is an array property, the relationship among all the array elements stipulated by the EDT data must comply with the EDT comparison data.

The EDT comparison data is classified into six types as indicated below:

- The EDT value in the message is equal to the EDT data.
- The EDT value in the message is greater than the EDT data.
- The EDT value in the message is less than the EDT data.
- The EDT value in the message is greater than or equal to the EDT data.
- The EDT value in the message is less than or equal to the EDT data.
- The EDT value in the message is not equal to the EDT data.

The "fire sensor class linkage (trigger) setting communications definition object" is described on the next page as an example for giving the details of properties retained by the objects of the linkage (trigger) setting communications definition class group. The fire sensor class has a class group code X1 of 0x00 and a class code X2 of 0x19. Therefore, the associated Set control reception method requirement object has a class group code X1 of 0x40, a class code X2 of 0x19, and the same instance code X3 value as fire sensor object X3.

Note that this Version does not support linkage based on compound messages.

In principle, ECHONET objects that belong to this class group must have a nonvolatile memory.

Part X, Section 3.12 specifies the method for building network systems using linkage setting object-based linkage settings.

Fire Sensor Class Communications Definition Objects for Linked (Trigger) Settings: Detailed Specifications

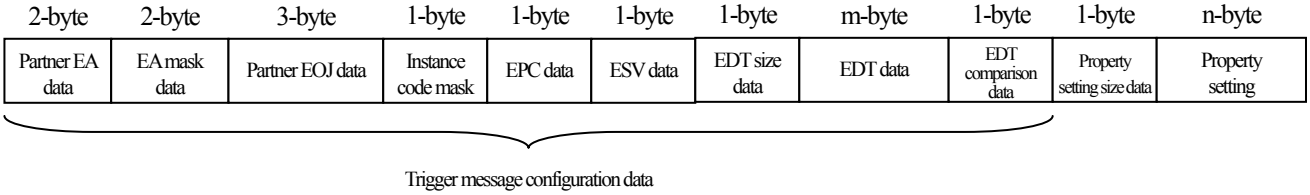
Class group code: 0x40

Class code: 0x19

Instance code: 0x01–0xFF (0x00: all instance requirement code)

Property Name	EPC	Contents of Property Value range (decimal notation)	Data Type	Data Size (Bytes)	Access Rule	Mandatory	Announce Status Change	Remarks
Communications definition for operating status	0x80	Communications definition data for linked setting (trigger setting) (see below)	unsigned char	Max 247	Set/ Get			
Communications definition for detection threshold level	0xB0	Communications definition data for linked setting (trigger setting) (see below)	unsigned char	Max 247	Set/ Get			
Communications definition for fire detection status	0xB1	Communications definition data for linked setting (trigger setting) (see below)	unsigned char	Max 247	Set/ Get			
Communications definition for malfunction status	0x88	Communications definition data for linked setting (trigger setting) (see below)	unsigned char	Max 247	Set/ Get			
Communications definition for malfunction content	0x89	Communications definition data for linked setting (trigger setting) (see below)	unsigned char	Max 247	Set/ Get			

The size of each property is variable up to 247 bytes. The property configuration is shown below:



Details of the individual configuration elements are given below:

① Partner EA data

This data consists of 2 bytes and retains the ECHONET address for an "individual" transmission. It is always used in conjunction with the EA mask data and is compared against the source ECHONET address (SEA) of a received message. The details are given under "2) EA mask data".

② EA mask data

This data consists of 2 bytes. It is used in conjunction with the partner EA data to check whether or not

Condition 1 is met. Condition 1 is met when the AND of the source ECHONET address (SEA) of the received message and the EA mask data matches the partner EA data.

③ Partner EOJ data

This data consists of 3 bytes and retains an ECHONET object code. It is always used in conjunction with the instance code mask data and compared against the source ECHONET object code (SEOJ) of the received message. The details are given under "4) Instance code mask data".

④ Instance code mask data

This is a 1-byte piece of data that is used in combination with the other side's EOJ data to check whether or not Condition 2 is satisfied. Condition 2 is satisfied if the value derived by ANDing the third byte (instance code X3) of the sender's ECHONET object code (SEOJ) of the received message with the instance code mask data matches the other side's EOJ data.

⑤ EPC data

This is a 1-byte piece of data that contains the ECHONET property code. Condition 3 is satisfied if the ECHONET property code (EPC) of the received message matches the EPC data.

⑥ ESV data

This is a 1-byte piece of data that contains the ECHONET service code. Condition 4 is satisfied if the ECHONET service code (ESV) of the received message matches the ESV data.

⑦ EDT size data

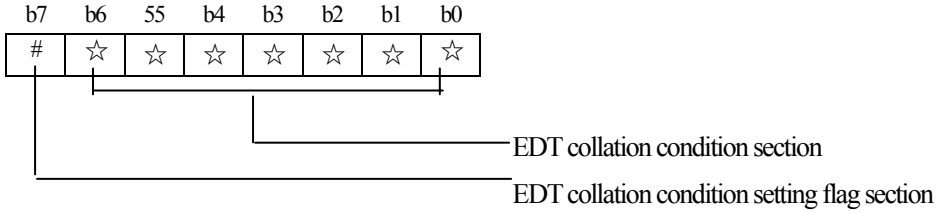
This is a 1-byte piece of data that contains the size (the number of bytes) of the EDT size data.

⑧ EDT content

The EDT content retains ECHONET data. The size of the EDT content is variable and indicated by the EDT size data. When the EDT content concerns an array element property, the first two bytes represent an array element number. The EDT content is always used in conjunction with the EDT comparison data and compared against the ECHONET data (EDT) in the received message. When the EDT content relates to a non-array property, the entire region is subjected to comparison. When it relates to an array property, on the other hand, the region excluding the first two bytes of the array element number data is subjected to comparison. The details are given under "9) EDT comparison data".

⑨ EDT collation data

Contains a 1-byte piece of data that indicates the method for collating the ECHONET data (EDT) of the received message with the EDT content. The EDT collation data is defined as follows:



EDT collation condition setting flag section (b7)

When b7 is 0, Condition 5 is satisfied if the EDT collation condition is satisfied (EDT collation condition section (b6 to b0)).When b7 is 1, Condition 5 is not satisfied regardless of whether or not the EDT collation condition is satisfied.

EDT collation condition section (b6 to b0)

The EDT collation condition section uses the following coding:

0x00: No linkage.
0x01: The EDT value in the message is equal to the EDT data. (=)
0x02: The EDT value in the message is greater than the EDT data. (>)
0x03: The EDT value in the message is less than the EDT data. (<)
0x04: The EDT value in the message is greater than or equal to the EDT data. (≧)
0x05: The EDT value in the message is less than or equal to the EDT data. (≦)
0x06: The EDT value in the message is not equal to the EDT data. (≠)

Condition 5 is met when the above conditions are satisfied within the region targeted for comparison between the EDT value in the message and the EDT data.

⑩ Property setting size data

This data consists of 1 byte. It retains the size (in bytes) of the information (property setting) to be written into a stipulated property of the associated object when the linkage trigger conditions (five individual conditions) are met.

⑪ Property setting

This data retains the information to be written into a stipulated property of the associated object when the linkage trigger conditions (five individual conditions) are met. The size of this data is variable and indicated by the property setting size data.

9.17 Requirements for the Local Alteration Limit Setting Communication Definition Class Group

This section provides the detailed requirements for ECHONET objects that belong to the local alteration

limit setting communication definition class group (class group code X1 = 0x60 to 0x6F). Objects that belong to this class group shall make the property alteration limit settings described below for each property of the corresponding object (i.e. the object that resides in the same node and whose class group code X1 (the 4 lowest-order bits, class code X2 and instance code X3 match).

* Setting for prohibiting the acceptance of alterations on the equipment side – properties having the same EPC in the corresponding object

* Setting to specify the conditions for acceptance of alterations by the equipment side – properties having the same EPC in the corresponding object

It is not necessary to permit alteration limit settings for all properties of the corresponding object.

The size of the properties shall be determined by the size of the property specified by the same EPC of the corresponding object. If the size of the property specified by the same EPC of the corresponding object is m bytes, the size of n is calculated as follows:

$$n = m \times 2 + 1 \text{ (bytes)}$$

The first (highest-order) byte of the property is "unsigned char" type data that specifies whether or not the corresponding property permits alterations on the equipment side and whether or not the corresponding property imposes limits on alterations on the equipment side (limit status data). The second byte and the succeeding bytes are a continuum of data of the same type and size as those of the corresponding property. The first data indicates the upper limit of the range and the second data indicates the lower limit. The composition of this property is as shown below.

1 byte	m bytes	m bytes
limit status data	upper limit	lower limit

The relationship between the value of the limit status data and the limit status shall be as follows:

Limit status data	Limit status
0x00	No limit is imposed.
0x01	No alteration is permitted.
0x02	Alteration limits are imposed (the range is defined by an upper and lower limit)

When the value of the limit status data is 0x00 (no limit) or 0x01 (alterations prohibited), the upper and

lower limit values shall be ignored.

Detailed explanation on properties held by objects that belong to this class group is given on the next page using the "local alteration limit setting communication definition object for the passage sensor class" as an example. The class group code X1 and class code X2 of the passage sensor class are 0x00 and 0x27, respectively, which means that the corresponding local alteration limit setting communication definition object class group code X1 and class code X2 are 0x60 and 0x27, respectively, and the instance code X3 is the same value as that of the passage sensor object X3.

Detailed Requirements for Property Value Alteration Limit Setting Communication Definition Objects

Class group code: 0x60

Class code: 0x27

Instance code: 0x01 to 0x7F (0x00: all instance designation)

Property Name	EPC	Property Content Value range (decimal notation)	Data Type	Size (bytes)	Access Rule	○ = Mandatory	Announce Status Change	Remarks
Communication definition – operation status	0x80	Communication definition data to specify the status notification method See below.	unsigned char x 3	3	Set/Get			
Communication definition – detection threshold level	0xB0	Communication definition data to specify the status notification method See below.	unsigned char x 1 unsigned short x 2	5	Set/Get			
Communication definition – passage detection hold time	0xBE	Communication definition data to specify the status notification method	unsigned char x 3	3	Set/Get			

The sizes of the properties are determined by the size of the property specified by the same EPC of the corresponding object. For example, the contents of the properties when

- the operation status is "alterations on the equipment side prohibited,"
- no limit is imposed on the detection threshold level, and
- the alteration range for the passage detection hold time between 500 and 5,000 msec are as follows:

Communication definition property – operation status

1 byte	1 byte	1 byte
0x01	don't care	don't care

Communication definition property – detection threshold level

1 byte	1 byte	1 byte
0x00	don't care	don't care

Communication definition property – passage detection hold time

1 byte	2 bytes	2 bytes
0x02	0x01F4 (500)	0x1388 (5,000)

9.18 Requirements for the Network Control Limit Status Display Communication Definition Class Group

This section provides the detailed requirements for ECHONET objects that belong to the network control limit status display communication definition class group (class group code X1 = 0x70 to 0x7F). Objects that belong to this class group permit the displaying of the local control prohibitions described below for each property of the corresponding object (i.e. the object that resides in the same node and whose class group code X1 (the 4 lowest-order bits), class code X2 and instance code X3 match).

- Displaying of the prohibition of acceptance of alterations by another node – properties having the same EPC in the corresponding object
- Displaying of the conditions for acceptance of alterations by another node – properties having the same EPC in the corresponding object

It is not necessary to permit the displaying of the alteration limits imposed for all properties of the corresponding object.

The sizes of the properties shall be determined by the size of the property specified by the same EPC of the corresponding object. If the size of the property specified by the same EPC of the corresponding object is m bytes, the size of n is calculated as follows:

$$n = m \times 2 + 1 \text{ (bytes)}$$

The first (highest-order) byte of the property is "unsigned char" type data that specifies whether or not the

corresponding property can accept alterations by another node and whether or not a limit is imposed on alterations from the network (limit status data).

The second byte and the succeeding bytes are a continuum of data of the same type and size as those of the corresponding property. The first data indicates the upper limit of the range and the second data indicates the lower limit. The composition of this property is as shown below.

1 byte	m bytes	m bytes
limit status data	upper limit	lower limit

The relationship between the value of the limit status data and the limit status shall be as follows:

Limit status data	Limit status
0x00	No limit is imposed.
0x01	No alteration is permitted.
0x02	Alteration limits are imposed (The range is defined by an upper and lower limit)

When the value of the limit status data is 0x00 (no limit) or 0x01 (alterations prohibited), the upper and lower limit values shall be ignored.

Detailed explanation on properties held by objects that belong to this class group is given on the next page using the "network control limit status display communication definition object for the passage sensor class" as an example. The class group code X1 and class code X2 of the passage sensor class are 0x00 and 0x27, respectively, which means that the corresponding network control limit status display communication definition object class group code X1 and class code X2 are 0x70 and 0x27, respectively, and the instance code X3 is the same value as that of the passage sensor object X3.

Detailed Requirements for Network Control Limit Status Display Communication Definition Objects

Class group code: 0x70

Class code: 0x27

Instance code: 0x01 to 0x7F (0x00: all instance designation)

Property Name	EPC	Property Content Value range (decimal notation)	Data Type	Size (bytes)	Access Rule	○ = Mandatory	Announce Status change	Remarks
Communication definition – operation status	0x80	Communication definition data to specify the status notification method See below.	unsigned char x 3	3	Get			

Communication definition – detection threshold level	0xB0	Communication definition data to specify the status notification method See below.	unsigned char x 1 unsigned short x 2	5	Get			
Communication definition – passage detection hold time	0xBE	Communication definition data to specify the status notification method	unsigned char x 3	3	Get			

The sizes of the properties are determined by the size of the property specified by the same EPC of the corresponding object. For example, the contents of the properties when

- the operation status is "network alterations prohibited,"
- no limit is imposed on the detection threshold level, and
- the alteration range for the passage detection hold time between 500 and 5,000 msec is as follows:

Communication definition property – operation status

1 byte	1 byte	1 byte
0x01	don't care	don't care

Communication definition property – detection threshold level

1 byte	1 byte	1 byte
0x00	don't care	don't care

Communication definition property – passage detection hold time

1 byte	2 bytes	2 bytes
0x02	0x01F4 (500)	0x1388 (5,000)

9.19 Specifications for Secure Communication Access Property Setup Class Group

The use of the "write service" for a write into the properties of the secure communication access property setup object in a User-Key-based authentication/enciphered message format makes it possible to set accessible properties in accordance with the authentication levels for the properties of the device objects, service objects, profile objects, and communications definition objects.

The class group codes and class codes indicate a secure communication access property setup object for device objects, profile objects, and service objects whose underlined* portions of codes are in agreement. Furthermore, accessible properties of a communications definition object agree with the settings for the

accessible properties of a device objects, profile object, or service object associated with the communications definition object.

Class group code: 0x5*

Class code: 0x**

Instance code: 0x01–0x7F (0x00: all instance requirement code)

Property Name	EPC	Contents of Property Value range (decimal notation)	Data Type	Data Size (Byte)	Access Rule	Mandatory	Announce Status Change	Remarks
SetM property map setting (Anonymous Level)	0xCB	Sets a property that can be subjected to SetM at anonymous level. See Supplement 2.	unsigned char x Max 17	Max. 17	Set/ Get			(1)
GetM property map setting (Anonymous Level)	0xCC	Sets a property that can be subjected to GetM at anonymous level. See Supplement 2.	unsigned char x Max 17	Max. 17	Set/ Get			(2)
Set property map setting (Anonymous Level)	0xCE	Sets a property that can be subjected to Set at anonymous level. See Supplement 2.	unsigned char x Max 17	Max. 17	Set/ Get			(3)
Get property map setting (Anonymous Level)	0xCF	Sets a property that can be subjected to Get at anonymous level. See Supplement 2.	unsigned char x Max 17	Max. 17	Set/ Get			(4)
SetM property map setting (Service Provider Level)	0xDB	Sets a property that can be subjected to SetM at service provider level. See Supplement 2.	unsigned char x Max 17	Max. 17	Set/ Get			(5)
GetM property map setting (Service Provider Level)	0xDC	Sets a property that can be subjected to GetM at service provider level. See Supplement 2.	unsigned char x Max 17	Max. 17	Set/ Get			(6)
Set property map setting (Service Provider Level)	0xDE	Sets a property that can be subjected to Set at service provider level. See Supplement 2.	unsigned char x Max 17	Max. 17	Set/ Get			(7)
Get property map setting (Service Provider Level)	0xDF	Sets a property that can be subjected to Get at service provider level. See Supplement 2.	unsigned char x Max 17	Max.17	Set/ Get			(8)

(1) SetM property map setting (Anonymous Level)

The "write service" (ESV = 0x60, 0x61) is used in a User-Key-based authentication/enciphered message format to set properties that can be subjected to SetM at the anonymous level for device objects, profile objects, and service objects having the secure communication access property setup class group code and class code whose underlined* portions are in agreement.

The property value must be in the property map description format defined in Part 2, Supplement 2 "Property Map Description Format".

When the "read service" (ESV = 0x62) is performed in the User-Key-based authentication/enciphered message format in relation to the captioned property, the map of properties that can be subjected to SetM at the anonymous level is obtained

(2) GetM property map setting (Anonymous Level)

The "write service" (ESV = 0x60, 0x61) is used in a User-Key-based authentication/enciphered message format to set properties that can be subjected to GetM at the anonymous level for device objects, profile objects, and service objects having the secure communication access property setup class group code and class code whose underlined* portions are in agreement.

The property value must be in the property map description format defined in Part 2, Supplement 2 "Property Map Description Format".

When the "read service" (ESV = 0x62) is performed in the User-Key-based authentication/enciphered message format in relation to the captioned property, the map of properties that can be subjected to GetM at the anonymous level is obtained

(3) Set property map setting (Anonymous Level)

The "write service" (ESV = 0x60, 0x61) is used in a User-Key-based authentication/enciphered message format to set properties that can be subjected to Set at the anonymous level for device objects, profile objects, and service objects having the secure communication access property setup class group code and class code whose underlined* portions are in agreement.

The property value must be in the property map description format defined in Part 2, Supplement 2 "Property Map Description Format".

When the "read service" (ESV = 0x62) is performed in the User-Key-based authentication/enciphered message format in relation to the captioned property, the map of properties that can be subjected to Set at the anonymous level is obtained

(4) Get property map setting (Anonymous Level)

The "write service" (ESV = 0x60, 0x61) is used in a User-Key-based authentication/enciphered message format to set properties that can be subjected to Get at the anonymous level for device objects, profile objects, and service objects having the secure communication access property setup class group code and class code whose underlined* portions are in agreement.

The property value must be in the property map description format defined in Part 2, Supplement 2 "Property Map Description Format".

When the "read service" (ESV = 0x62) is performed in the User-Key-based authentication/enciphered message format in relation to the captioned property, the map of properties that can be subjected to Get at the anonymous level is obtained

(5) SetM property map setting (Service Provider Level)

The "write service" (ESV = 0x60, 0x61) is used in a User-Key-based authentication/enciphered message format to set properties that can be subjected to SetM at the service provider level for device objects, profile objects, and service objects having the secure communication access property setup class group code and class code whose underlined* portions are in agreement.

The property value must be in the property map description format defined in Part 2, Supplement 2 "Property Map Description Format".

When the "read service" (ESV = 0x62) is performed in the User-Key-based authentication/enciphered message format in relation to the captioned property, the map of properties that can be subjected to SetM at the service provider level is obtained

(6) GetM property map setting (Service Provider Level)

The "write service" (ESV = 0x60, 0x61) is used in a User-Key-based authentication/enciphered message format to set properties that can be subjected to GetM at the service provider level for device objects, profile objects, and service objects having the secure communication access property setup class group code and class code whose underlined* portions are in agreement.

The property value must be in the property map description format defined in Part 2, Supplement 2 "Property Map Description Format".

When the "read service" (ESV = 0x62) is performed in the User-Key-based authentication/enciphered message format in relation to the captioned property, the map of properties that can be subjected to GetM at the service provider level is obtained

(7) Set property map setting (Service Provider Level)

The "write service" (ESV = 0x60, 0x61) is used in a User-Key-based authentication/enciphered message format to set properties that can be subjected to Set at the service provider level for device objects, profile objects, and service objects having the secure communication access property setup class group code and class code whose underlined* portions are in agreement.

The property value must be in the property map description format defined in Part 2, Supplement 2 "Property Map Description Format".

When the "read service" (ESV = 0x62) is performed in the User-Key-based authentication/enciphered message format in relation to the captioned property, the map of properties that can be subjected to Set at the service provider level is obtained

(8) Get property map setting (Service Provider Level)

The "write service" (ESV = 0x60, 0x61) is used in a User-Key-based authentication/enciphered message format to set properties that can be subjected to Get at the service provider level for device objects, profile objects, and service objects having the secure communication access property setup class group code and class code whose underlined* portions are in agreement.

The property value must be in the property map description format defined in Part 2, Supplement 2 "Property Map Description Format".

When the "read service" (ESV = 0x62) is performed in the User-Key-based authentication/enciphered message format in relation to the captioned property, the map of properties that can be subjected to Get at the service provider level is obtained.

9.20 Requirements for the Property Status Referencing Request Compliability Communication Definition Class Group

This section defines the detailed requirements for ECHONET objects that belong to the property status

referencing request compliability communication definition class group (class group code X1 = 0x80 to 0x8F). Objects that belong to this class group permit the displaying (for other nodes) and alteration (by other nodes) of the status described below of each of the properties of the applicable objects (i.e. the objects which reside in the same node and whose class group code X1 (the 4 lowest-order bits), class code X2 and instance code X3 match). However, whether or not alteration requests from other nodes are actually accepted shall be implementation-dependent.

It is not required to display the status of every property of the applicable objects as to whether it is ready to accept referencing requests from other nodes, and it is not required to provide a function that allows other nodes to alter the status setting for every property of the applicable objects. This class shall not allow the referencing of the status of a property map property (as to whether the property is ready to accept referencing requests from other nodes) to be prohibited.

・Status of each of the properties having the same EPC of the applicable objects as to whether the property is ready to accept referencing requests from other nodes (displaying and alteration of the status)

The size of each property shall be 1 byte, and b0 shall indicate whether or not referencing is possible.

If b0 is 0, the corresponding property shall operate as usual (i.e. If the property is a referenceable property, the value will be returned). If b0 is 1, referencing for the corresponding property shall not be allowed and a "response not possible" message shall be returned upon a referencing request.

Detailed requirements for properties of property status referencing request compliability communication definition class group objects are shown on the next page using the "passage sensor class property status referencing request compliability communication definition object" as an example. The class group code and class code for the passage sensor class is X1 = 0x00 and X2 = 0x27, respectively. Therefore, the class group code X1 and class code X2 for the corresponding property status referencing request compliability communication definition object are 0x80 and 0x27, respectively, and the instance code X3 is the same as the X3 for passage sensor objects.

Detailed Requirements for Property Status Referencing Request Compliability Communication Definition Objects

Class group code: 0x80

Class code: 0x27

Instance code: 0x01 to 0x7F (0x00: all instances)

Property Name	EPC	Content of Property Value range (decimal notation)	Data Type	Size (Bytes)	Access Rule	Mandatory	Status Change Announcement	Remark

Communication definition for operating status	0x80	Communication definition data for specifying status notification method	unsigned char	1	Set/ Get			
Communication definition for detection threshold level	0xB0	Communication definition data for specifying status notification method	unsigned char	1	Set/ Get			
Communication definition for passage detection hold time	0xBE	Communication definition data for specifying status notification method	unsigned char	1	Set/ Get			

9.21 Requirements for the Property Status Notifiability Communication Definition Class Group

This section defines the detailed requirements for ECHONET objects that belong to the property status notifiability communication definition class group (class group code X1 = 0x90 to 0x9F). Objects that belong to this class group permit the displaying (for other nodes) and alteration (by other nodes) of the status described below of each of the properties of the applicable objects (i.e. the objects which reside in the same node and whose class group code X1 (the 4 lowest-order bits), class code X2 and instance code X3 match). However, whether or not alteration requests from other nodes are actually accepted shall be implementation-dependent.

It is not required to display the status of every property of the applicable objects as to whether it is ready to provide notifications, and it is not required to provide a function that allows other nodes to alter the status setting for every property of the applicable objects.

・Status of each of the properties having the same EPC of the applicable objects as to whether the property is ready to provide notifications (displaying and alteration of the status)

The size of each property shall be 1 byte, and b0 shall indicate whether or not notification is possible.

If b0 is 0, the corresponding property shall operate as usual (i.e. If the conditions for notification are met, notification will be made). If b0 is 1, notifications by the corresponding property shall not be allowed under any conditions.

Detailed requirements for properties of property status notifiability communication definition class group objects are shown on the next page using the "passage sensor class property status notifiability communication definition object" as an example. The class group code and class code for the passage sensor class is X1 = 0x00 and X2 = 0x27, respectively. Therefore, the class group code X1 and class code X2 for the corresponding property status notifiability communication definition object are 0x90 and 0x27, respectively, and the instance code X3 is the same as the X3 for passage sensor objects.

Detailed Requirements for Property Status Notifiability Communication Definition Objects

Class group code: 0x90

Class code: 0x27

Instance code: 0x01 to 0x7F (0x00: all instances)

Property Name	EPC	Content of Property Value range (decimal notation)	Data Type	Size (Bytes)	Access Rule	Mandatory	Status Change Announcement	Remark
Communication definition for operating status	0x80	Communication definition data for specifying status notification method	unsigned char	1	Set/ Get			
Communication definition for detection threshold level	0xB0	Communication definition data for specifying status notification method	unsigned char	1	Set/ Get			
Communication definition for passage detection hold time	0xBE	Communication definition data for specifying status notification method	unsigned char	1	Set/ Get			

Chapter 10 ECHONET Security Communication Specification

10.1 ECHONET Security Problems

- Power lines, radio, etc. are vulnerable to attack by hackers.
- Impersonation and falsification are prevented. Falsification is detected.
- Access is restricted by authentication to cope with illegal access from outside the network.
- Wiretap is prevented by enciphering.

10.2 ECHONET Security Policy

It is necessary to clarify the extent to which ECHONET can be observed. The following three actions shall be performed to provide secure ECHONET communication.

①	To prevent wiretap	Common key system enciphering
②	To detect falsification	Hash signature
③	To prevent impersonation	Authentication header

10.3 Encryption Method for ECHONET Secure Communication

- The encryption method shall be AES-CBC.
- The key size shall be 128 bits.

10.4 Positioning of ECHONET in the Protocol Stack

- Realization of secure communication independent from media
- Possible authentication between nodes having different media

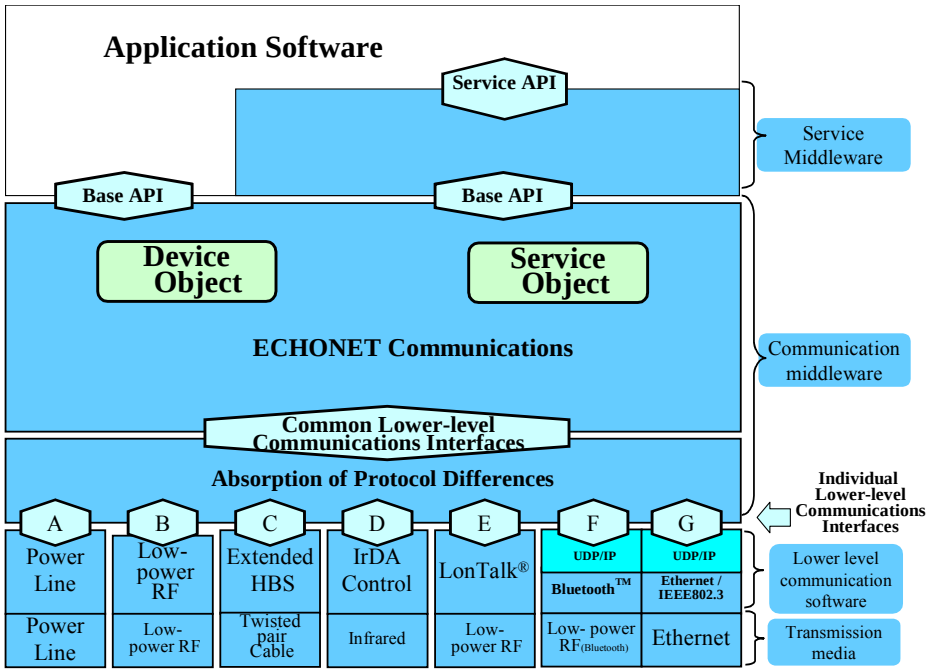


Fig. 10.1 Secure Communication in ECHONET

10.5 Configuration of Secure Communication Messages in ECHONET

10.5.1 ECHONET Secure Message Format

Refer to "Fig. 4.1-2 ECHONET Frame for Secure Messages" in "4.2 Message Structure". Detailed specifications for each Message element are provided below.

10.5.2 ECHONET Header (EHD)

Refer to "4.2.1 ECHONET Header (EHD)".

10.5.3 ECHONET Byte Counter (EBC)

This is a 1-byte piece of data that indicates the size of the EDATA section explained in Fig. 4.1-2 "ECHONET Frame for Secure Messages." When the specified message type is the secure type, a value between 22 and 256 (0x16 – 0xFF, 0x00; 0x00 indicates 256) can be used to specify the size of the EDATA section. A message must be at least 22 bytes long (8 bytes for SHD, 1 byte for PBC, 6 bytes for PEDATA, 1 byte for BCC and 6 bytes for PDG) if it is to be treated as a secure message.

10.5.4 ECHONET Secure Header (SHD)

Fig. 10.2 shows the format for the ECHONET secure header (SHD).

TID	SKH	SNF
TID: Transaction ID (2 bytes) SKH: Secure key header (2 bytes) SNF: Sequence number field (4 bytes)		
SHD		

SHD: secure header (8 bytes)

Fig. 10.2 ECHONET Secure Header

10.5.4.1 Transaction ID (TID)

The size shall be 2 bytes. A transaction ID is defined to bind a request message and a response message. Any value can be stored when a request message is transmitted. The same value as the received request message shall be stored when a response message is transmitted.

10.5.4.2 Secure Key Head (SKH)

The size shall be 2 bytes. It denotes authentication, encryption, secure level, distinction of authentication request / response, and authentication status. Four kinds of secure message formats are defined by the combination of b6 and b7.

	The 1st byte								The 2nd byte							
	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
	#	#	#	#	0	0	0	#	#	#	0	0	#	#	#	#

b3:b2:b1:b0	**Secure User Level** b3:b2:b1:b0=0:0:0:0 Serial Key Index b3:b2:b1:b0=0:0:0:1 User Secure Key Index b3:b2:b1:b0=0:0:1:0 Maker Secure Key Index Others Service Provider Secure Key Index
b5:b4	Reserved for future use
b6	**Authenticate or not** 0:Authenticated 1:not authenticated
b7	**Encrypt or not** 0:encrypted 1:not encrypted
b8	**Authentication request / response** 0:request 1:response
b9~b11	Reserved for future use
b15:b14:b13:b12	**Authentication succeeded / failed** b15:b14:b13:b12 = 0:0:0:0: succeeded b15:b14:b13:b12 = 0:0:0:1: SNF inconformity b15:b14:b13:b12 = 0:0:1:0: authentication signature inconformity b15:b14:b13:b12 = 0:1:0:0: no access right others: reserved for future use

Fig. 10.3 Secure Key Head (SKH)

When b6 is 1 (with authentication) and b7 is 0 (with encryption), it shall be presumed that the message type is "encryption with authentication" and encrypted communication shall be used for ECHONET secure messages. Figure 10.4 shows the frame format.

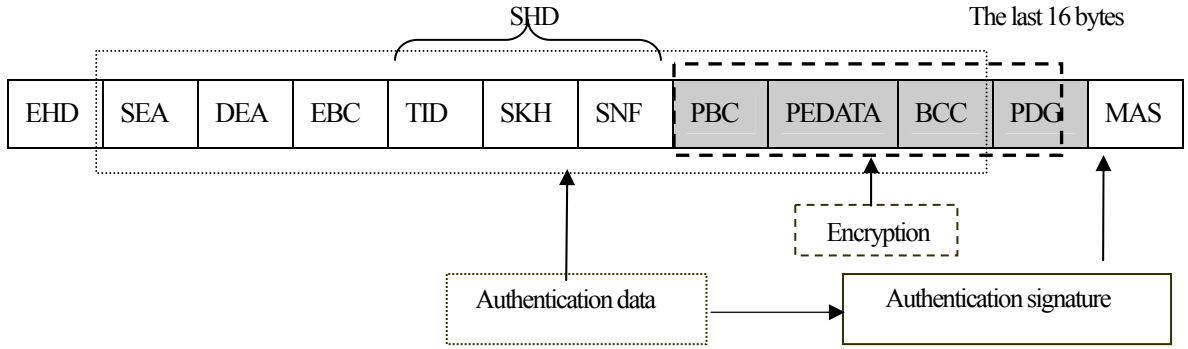


Fig. 10.4 Frame Format When Both Encryption and Authentication Are Used

When b6 is 1 (without authentication) and b7 is 0 (with encryption), it shall be presumed that the message type is "encryption without authentication" and encrypted communication shall be used for ECHONET secure messages. Figure 10.5 shows the frame format. There is no requirement with regard to the value for b8 of SKH. Service assuming entities shall not perform SNF-based management of service requesting entities. The service requesting entity shall not retain the SNF value that is available to it at the time of reception of the response. However, the SNF value shall be used as the initial vector when encryption/decryption is performed.

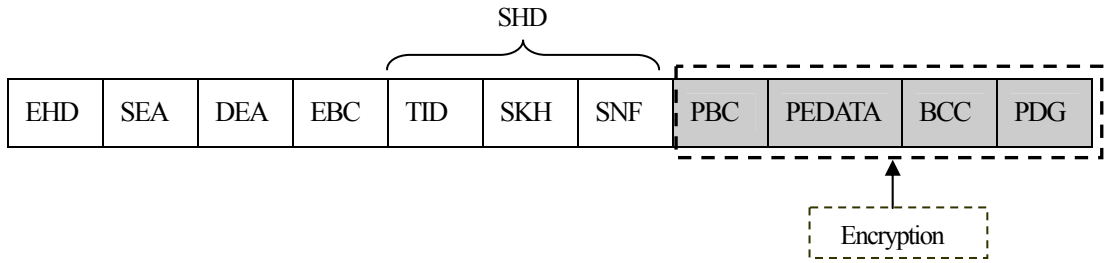


Fig. 10.5 Frame Format When Encryption Is Used Without Authentication

When b6 is 0 (with authentication) and b7 is 1 (without encryption), it shall be presumed that the message type is "authentication without encryption" and authentication shall be used for ECHONET secure messages. Figure 10.6 shows the frame format.

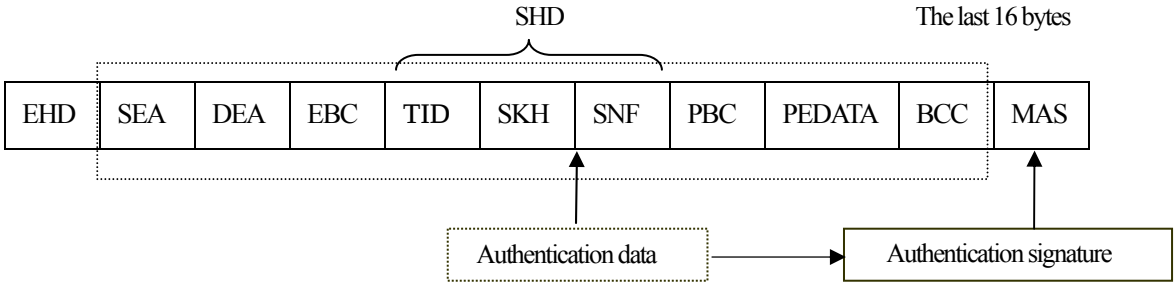


Fig. 10.6 Frame Format When Authentication Is Used Without Encryption

When b6 is 0 (without authentication) and b7 is 0 (with encryption), it shall be presumed that the message type is "without authentication and encryption" and encrypted communication shall be used for ECHONET secure messages. Figure 10.7 shows the frame format. b0, b1, b2, b3 and b8 of SKH are not specified. Service assuming entities shall not perform SNF-based management of service requesting entities.

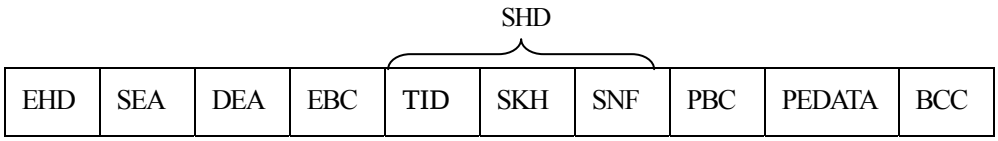


Fig. 10.7 Frame Format When Neither Encryption Nor Authentication Is Used

The Secure User Level corresponds to the authentication and enciphering level in ECHONET secure communication. The Secure User Level shown in b3:b2:b1:b0 is described below.

・ Supervisor Level

Coding and decoding, or authentication is performed by the Serial Key of the ECHONET device. The Serial Key is set during manufacturing of the ECHONET device, and is displayed on the case of the device. The party who supervises the access rights to the device uses this Serial Key during the initial setting of the common key (User Secure Key, Service Provider Key) to the ECHONET device.

・ User Level

Coding and decoding, or authentication is performed by the User Secure Key. The common key is supervised by a supervisor of the domain who sets one common key for one domain. This is used in manipulating information that the inhabitant does not wish to disclose to others.

・ Maker Level

Coding and decoding, or authentication is performed by the Maker Secure Key, which is supervised by a manufacturer of the device. This is used in manipulating information that the manufacturer does not wish to disclose to others.

- Service Provider Level

Coding and decoding, or authentication, is performed by the Service Provider Secure Key. The Service Provider Secure Key transfers rights to a third party when the supervision of stipulated devices is entrusted to the third party by the owner of the device. It is used in manipulating information that is not meant to be seen by persons other than the third party in question.

In ECHONET secure communication (authentication), b8 shall be used to differentiate authentication requests from authentication responses. However, when the value of b6 specifies "without authentication" (b6 = 1), the value of b8 shall have no meaning.
In ECHONET secure communication (authentication), b12, b13, b14 and b15 shall indicate, in an authentication response, the result (passed / not passed) of the authentication.

The result shall be "not passed" if:

- the SNF of the received message does not match the home SNF;
- the received message's authentication signature does not match the authentication signature created from the received message; or
- no authorization is granted for access.

However, the values of b12, b13, b14 and b15 shall have no meaning unless b8 specifies "authentication response" (b8 = 1) and b6 specifies "with authentication" (b6 = 0).

10.5.4.3 Sequence Number Field (SNF)

The size shall be 4 bytes. The initial value shall be determined using a random number generation algorithm. Sequence numbers shall be managed by service assuming entities for each service requesting node. When a node cold-starts, the initial value of the sequence number shall be determined using a random number generation algorithm. When a node warm-starts, the sequence number shall be determined either by reading the value stored in the nonvolatile memory or using a random number generation algorithm. If the result of the authentication is "passed," the service assuming entity shall increase the sequence number by one and store that value.
The service requesting entity shall use the sequence number it received in the previous authentication response from the service assuming entity.
In the case of encrypted communication, this shall be used only as the initial vector for encryption/decryption. In the case of encrypted communication and encrypted plain text communication, service assuming entities shall not perform SNF-based management of service requesting entities.

10.5.5 Certification Signature (MAS)

The size shall be 16 bytes. In the case of the "encryption and authentication" type, the encryption signature shall be calculated using the encryption function from the SEA, DEA, EBC, SHD, encrypted PBC, PEDATA, BCC and PDG (+ zero padding) of the secure message frame shown in Fig. 4.1.2. The calculation of an authentication signature that uses the encryption function shall use the secure key.

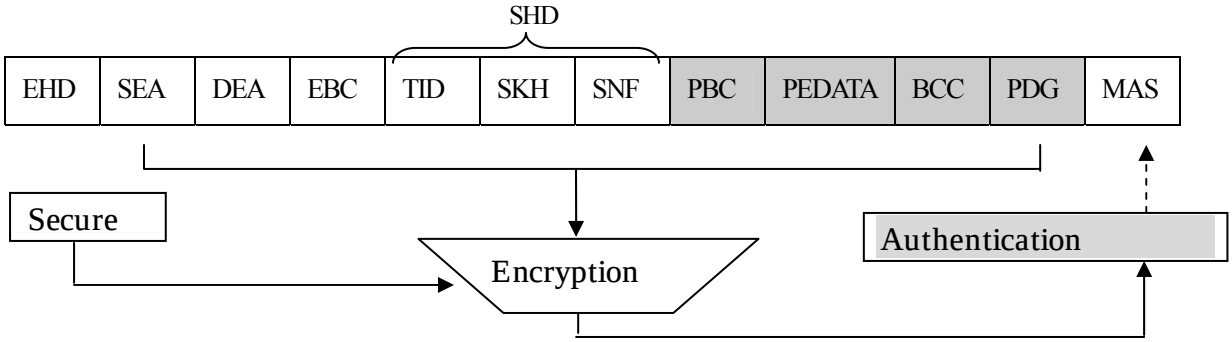


Fig. 10.8 Authentication Signature ("encryption and authentication" type)

In the case of the "authentication" type, the encryption signature shall be calculated using the encryption function from the SEA, DEA, EBC, SHD, PBC, PEDATA and BCC (+ zero padding) of the secure message frame shown in Fig. 4.1.2. The calculation of an authentication signature that uses the encryption function shall use the secure key.

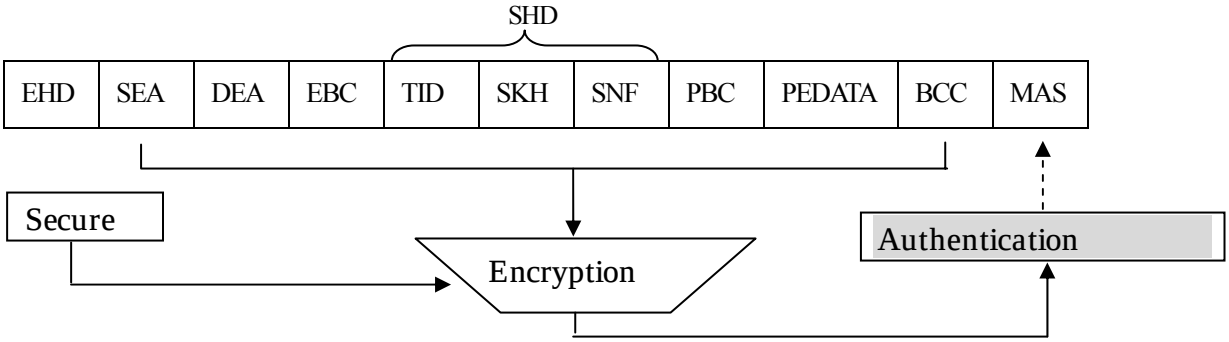


Fig. 10.9 Authentication Signature ("authentication without encryption" type)

The calculation of authentication signatures (MAS) shall use the Cipher Block Chaining mode. The initial register vector shall be calculated from the SNF. Figure 10.12 shows the initial vector and Fig. 10.10 shows the method used to calculate the authentication signature (MAS) using the Cipher Block Chaining mode. The unit of processing shall be the block encryption input/output size (128 bits) for all arrows shown in Fig. 10.10. When the b7 and b6 data of the secure key header (SKH) shown in Fig. 10.3 indicates that the type is "encryption and authentication", the input block shall be composed from the zero padding to achieve 128-bit alignment in the continuous field for the SEA, DEA, EBC, SHD, encrypted PBC, PEDATA, BCC and PDG and the corresponding output block shall be treated as the authentication signature (MAS).

When the b7 and b6 data of the secure key header (SKH) shown in Fig. 10.3 indicates that the type is "authentication without encryption", the input block shall be composed from the zero padding to achieve 128-bit alignment in the continuous field for the SEA, DEA, EBC, SHD, PBC, PEDATA and BCC and the corresponding output block shall be treated as the authentication signature (MAS).

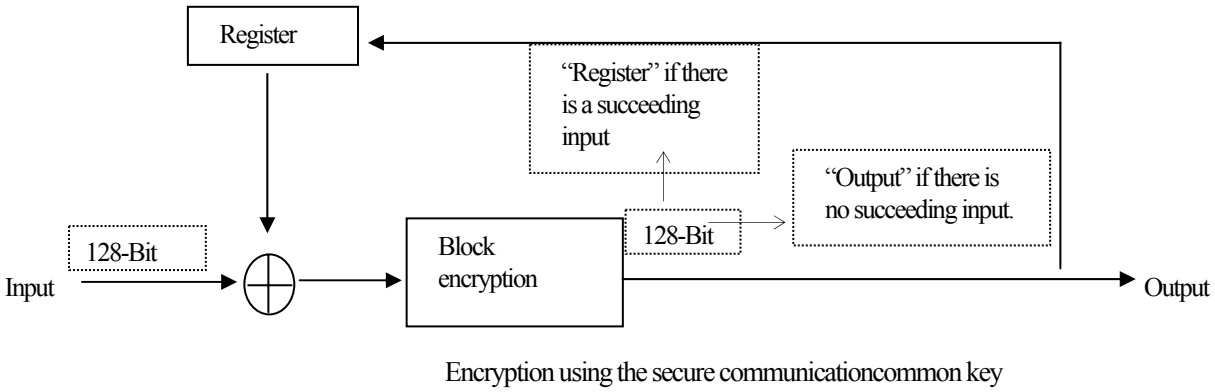


Fig. 10.10 Calculation of Authentication Signature (MAS) Using the Cipher Block Chaining Mode

10.5.6 Plain Text ECHONET Data Section Byte Counter (PBC)

The size shall be 1 byte.The plain text ECHONET data section byte counter (PBC) indicates the size of the plain text ECHONET data (PEDATA).

10.5.7 Plain Text ECHONET Data (PEDATA)

The plain text ECHONET data (PEDATA) shall take the form shown in Fig. 10.11 (Message type I) when b2, b1 and b0 of the ECHONET header (EHD) are 1, 0 and 0, respectively.

When b6 of the SKH is 0 (= "with authentication"), the size of the plain text ECHONET data (PEDATA) shall be a value between 6 and 222 bytes.

When b6 of the SKH is 1 (= "without authentication"), the size of the plain text ECHONET data (PEDATA) shall be a value between 6 and 238 bytes.

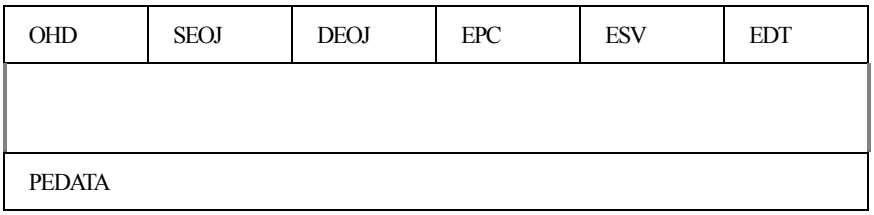


Fig. 10.11Plain Text ECHONET Data (PEDATA)

10.5.8 Block Check Code (BCC)

The size shall be 1 byte. It indicates the block check code ("horizontal parity") for the SEA, DEA, EBC, SHD, PBC and PEDATA of the secure message frame (plain text) explained in Fig. 4.1-2.

10.5.9 Padding (PDG)

The size shall be 0 to 15 bytes. When making an ECHONET secure communication frame from a secure message (plain text) frame having the format shown in Fig. 4.1-2 using the method explained in "10.6.2 Common Key Block Encryption," padding using 0x00 shall be made to adjust the size of the plain text for 128-bit encryption.

10.6 Encryption

The encryption of secure message frames having the format shown in Fig. 4.1-2 shall be common key-based encryption. The sender and recipient shall have the same private key.

10.6.1 Initial Vector

The SNF shall be used as the source of the initial vector for encrypting and decrypting a message and for creating an authentication signature (MAS). Figure 10.12 shows the relationship between the initial vector and SNF.

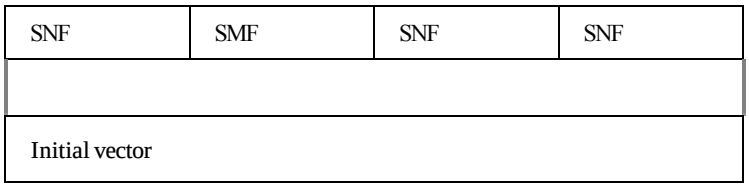


Fig. 10.12 Initial Vector

10.6.2 Common Key Block Encryption

A block encryption algorithm shall be used that converts 128-bit plain text blocks into 128-bit encrypted blocks using a common key. The PBC, PEDATA, BCC and PDG of the secure message frame shown in Fig. 4.1-2 shall be encrypted to be used as the secure message EDATA section that comprises the ECHONET secure communication frame.

The initial vector for encryption and decryption shall be calculated from the SNF (see Fig. 10.12).

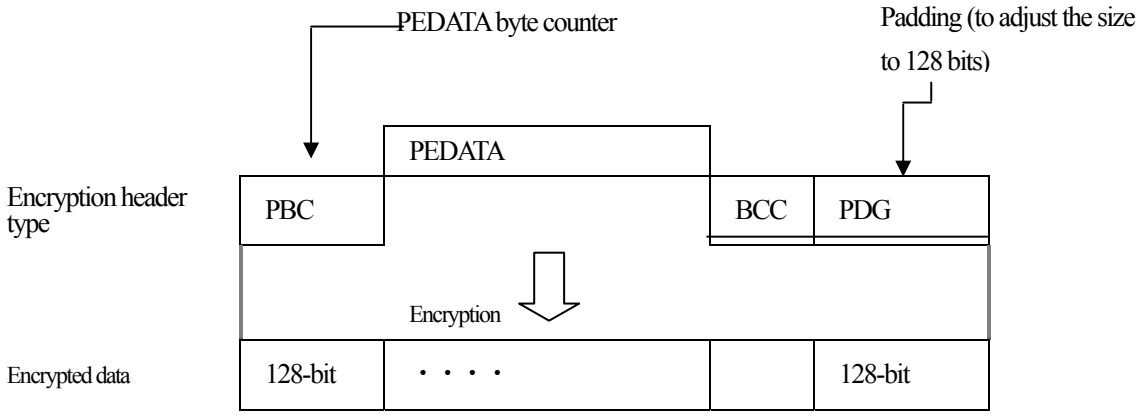


Fig. 10.13 Encrypted Part of a Secure Message

After decrypting the received message, the BCC (Fig. 10.13) shall be checked, and if different, the received message shall be discarded.

10.7 Certification Sequence

10.7.1 Certification Sequence

When the type is "authentication and encryption" or "authentication without encryption", the DEA of the service requesting entity's authentication request message shall specify "individual." If it specifies "broadcast," the service assuming entity must discard the message. Sequence numbers (SNF) shall be managed by service assuming entities for each service requesting node.

The authentication sequence shall not use instance broadcasting (the lowest-order byte of EOJ = 0x00). If instance broadcasting is used, the service assuming entity shall discard the message.

Figure 10.14 shows the ECHONET secure frame and authentication sequence for the "authentication and encryption" type and Fig. 10.15 shows the ECHONET secure frame and authentication sequence for the "authentication without encryption" type.

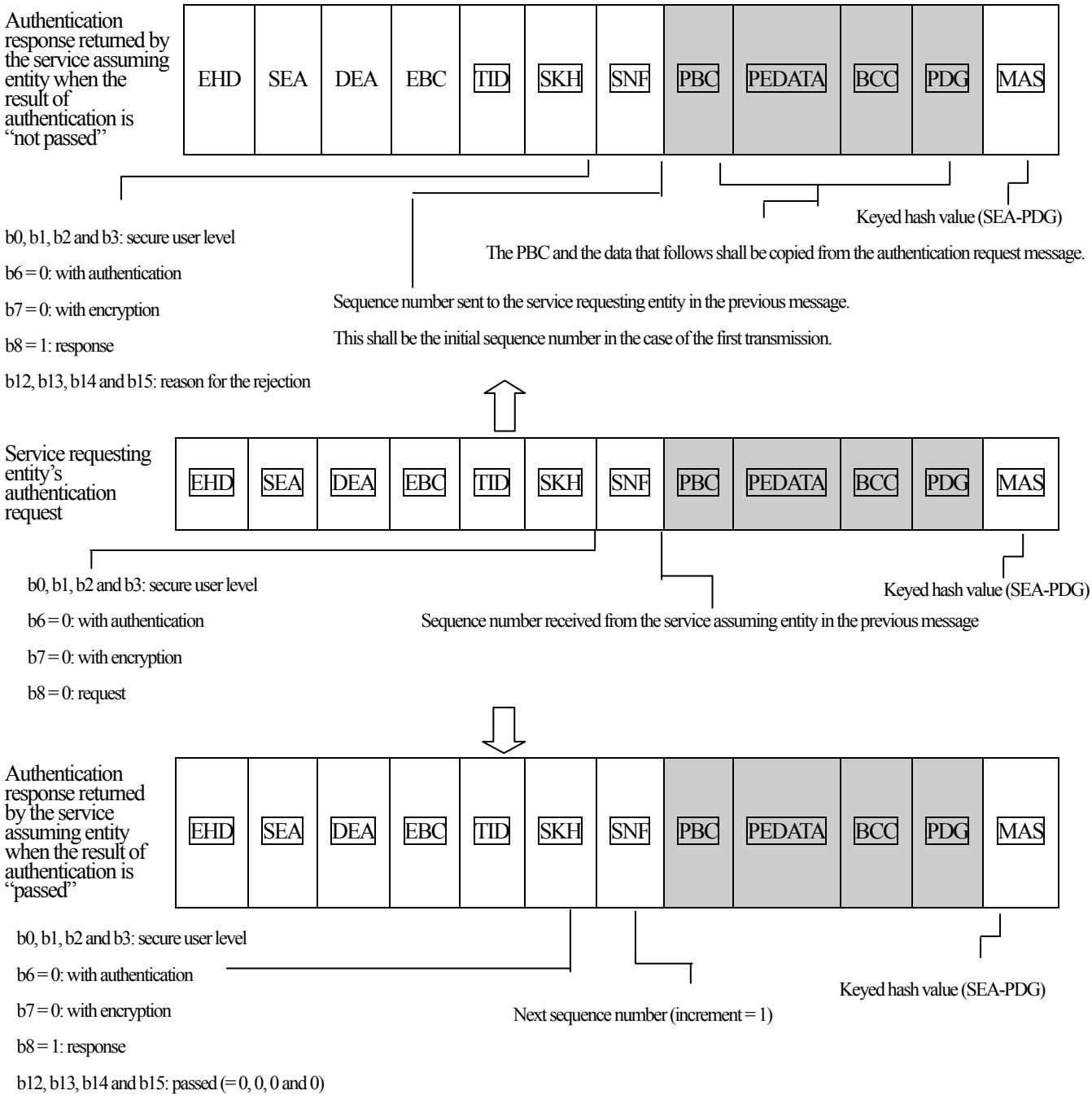


Fig. 10.14 ECHONET Secure Frame and Authentication Sequence ("authentication and encryption")

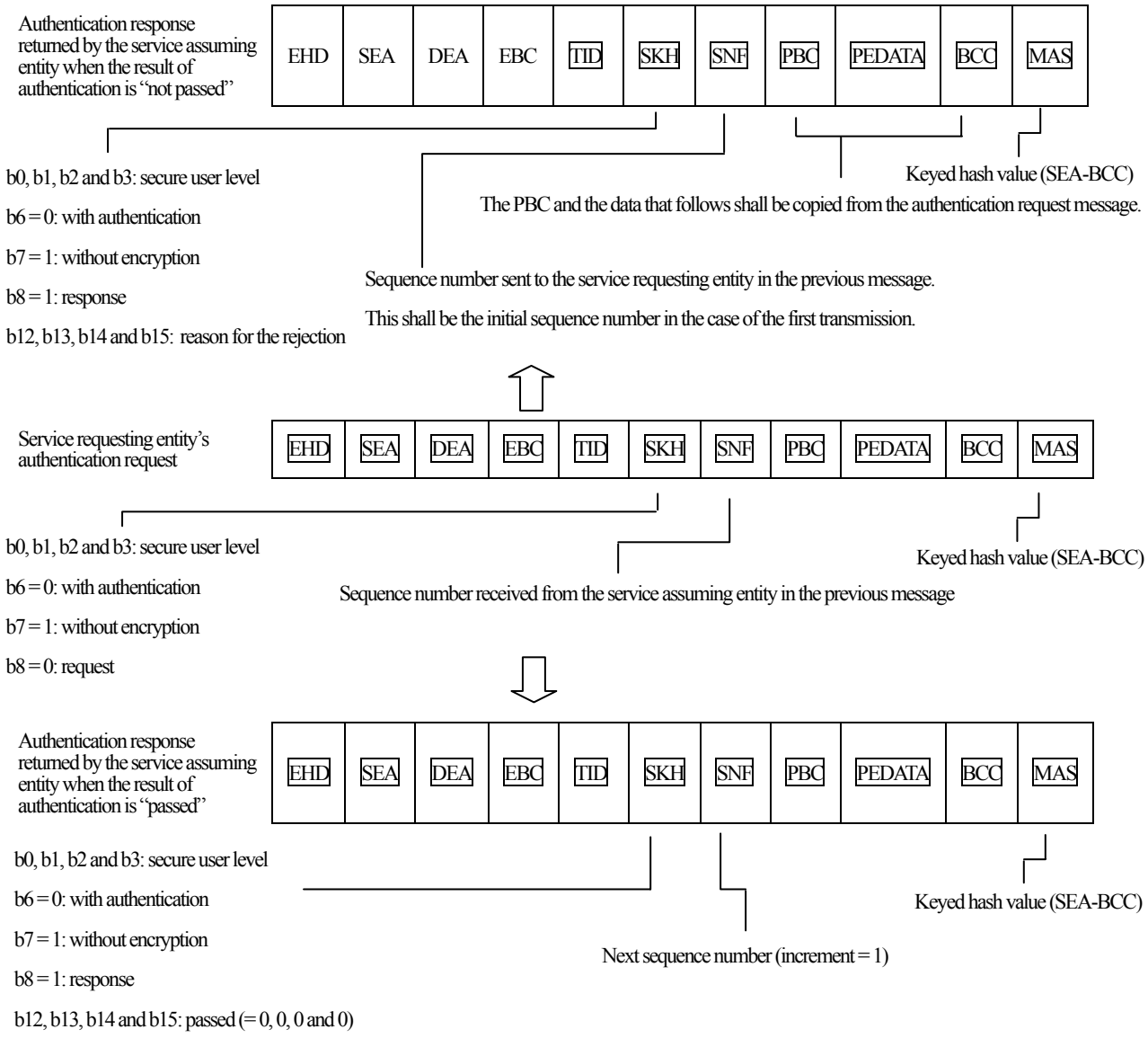


Fig. 10.15 ECHONET Secure Frame and Authentication Sequence ("authentication and encryption")

Figure 10.16 shows the authentication sequence. The service requesting entity shall create a MAS from the common key, SEA, DEA, EBC, TID, SKH, PBC, PEDATA, BCC, PDG and SNF received from the service assuming entity at the time of the previous authentication and send it to the service assuming entity.

The service assuming entity shall confirm whether or not the received SNF matches the last SNF that was sent to the client. It shall also confirm whether or not the received MAS matches the MAS calculated from the SNF and the received SEA, DEA, EBC, TID, SKH, PBC, PEDATA, BCC, PDG and common key.

If both the received SNF and MAS match the values described above, the requested action specified by the PEDATA shall be performed. The sequence number (SNF) is increased by 1, a MAS is created from the SNF, SEA, DEA, EBC, TID, SKH, PBC, PEDATA, BCC, PDG and common key, and then an authentication response containing the SNF and MAS is transmitted.

If either the received SNF or MAS does not match the corresponding value described above (which means

that "not passed" is to be returned), a MAS shall be created from the common key, SEA, DEA, EBC, TID, SKH, PBC, PEDATA, BCC, PDG and SNF sent to the service requesting entity in the previous message, a message shall be created that contains the MAS and SNF sent to the service requesting entity in the previous message and uses the PBC and the succeeding data of the service assuming entity's authentication request message as its PBC and succeeding data, and the message shall be sent as a "not passed" authentication response message.

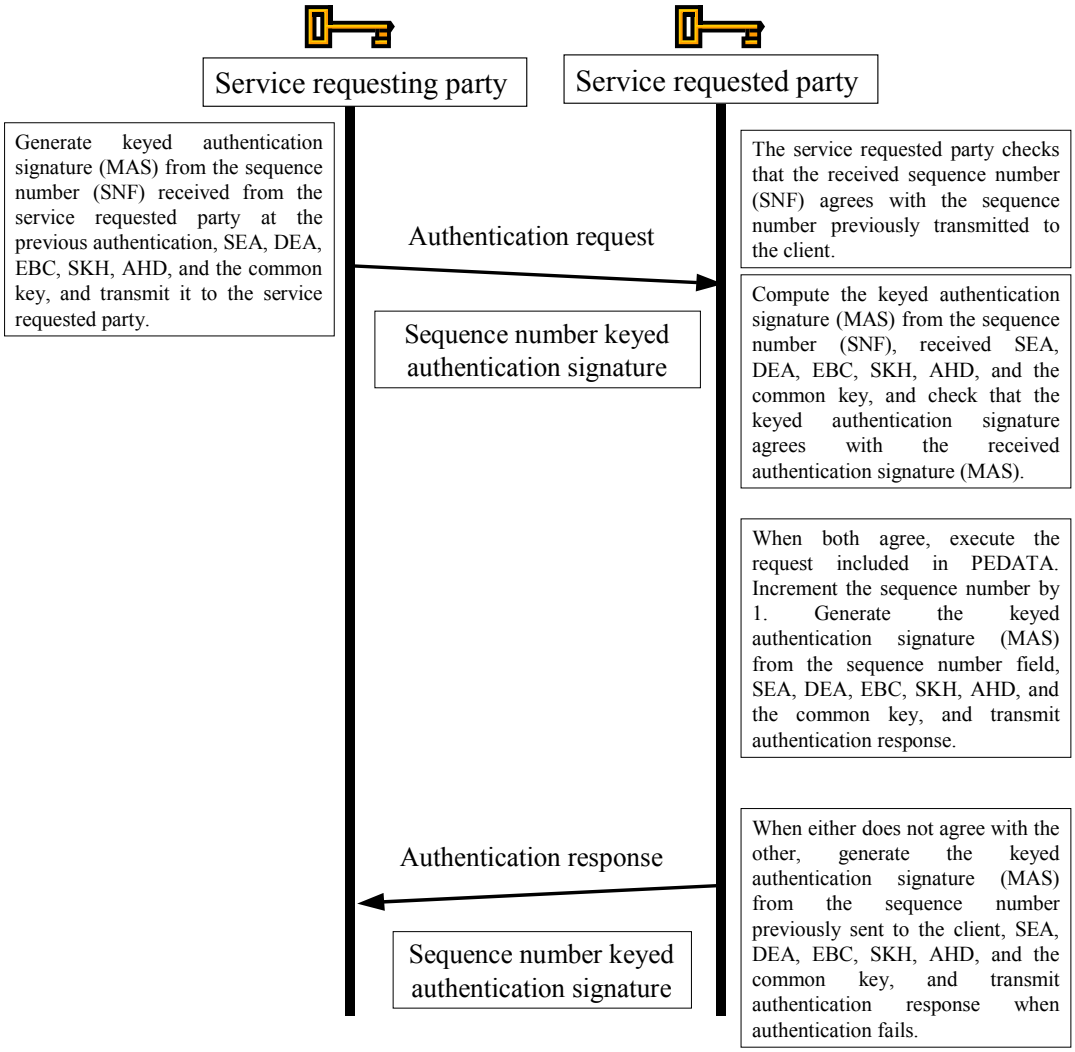


Fig. 10.16 Authentication Sequence

Figure 10.17 shows the authentication sequence in the first authentication request from the node of the service requesting party to the node of the service requested party. Because the service requesting party has not previously received a sequence number from the service requested party, the service requesting party transmits the authentication request to the service requested party and includes an arbitrary sequence number in the sequence number field (SNF).

Because the received sequence number is different from the controlled sequence number, the service requested party transmits an "authentication failed" response, including the controlled sequence number, to the service requesting party. The service requesting party acquires the normal sequence number from the

sequence number field (SNF) of the "authentication failed" response and transmits the authentication request, including the acquired sequence number in the sequence number field (SNF), to the service requested party.

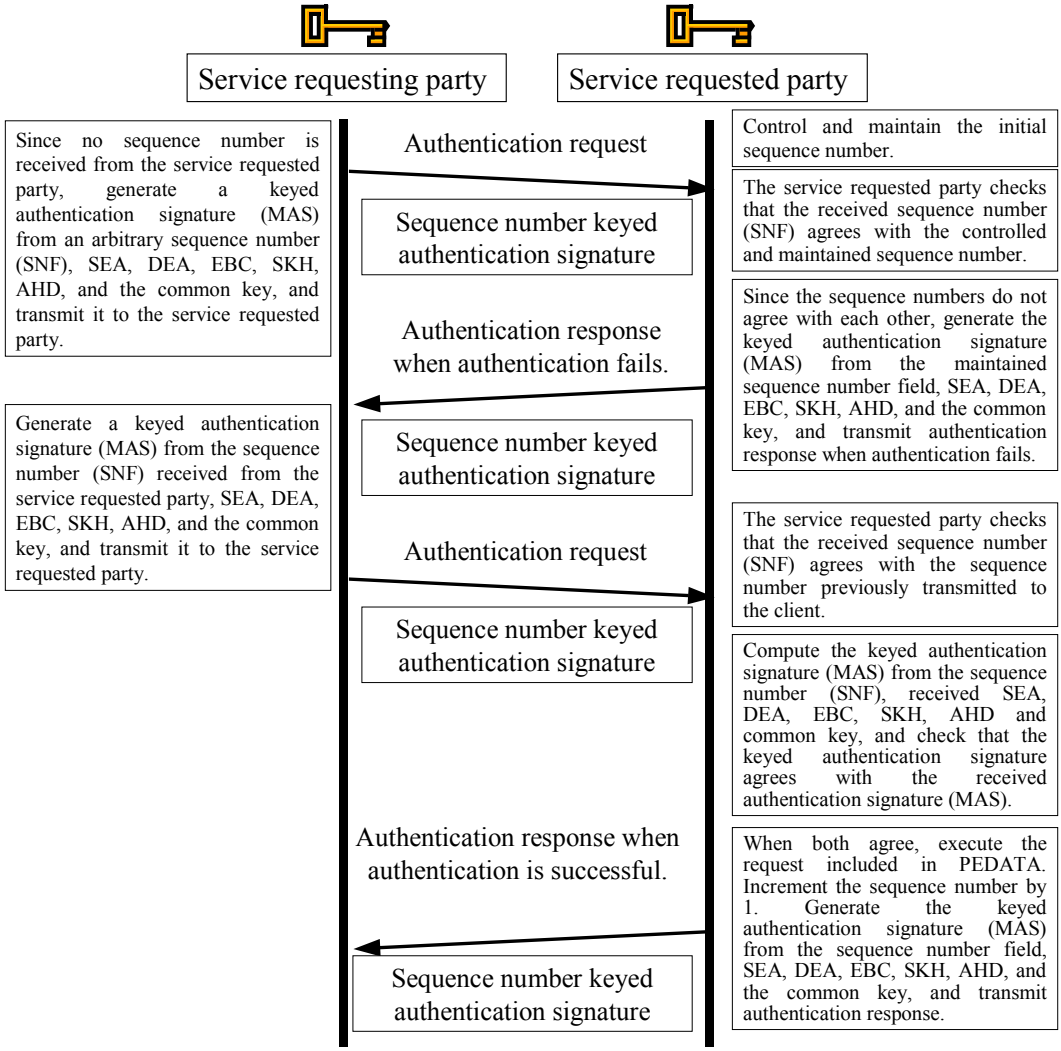


Fig. 10.17 Initial Authentication Sequence

10.8 Secure Communication Common Key Management

10.8.1 Detailed Requirements for the Secure Communication Common Key Setting Class

See "9.9.1 Detailed Requirements for the Secure Communication Common Key Setting Node Class."

10.8.2 Secure Communication Common Key Setting Method

Different initial setting methods and management methods are used for the 3 key types (User Key, Service Provider Key and Maker Key).

The initial User Key (secure communication common key) setting shall be made by making an offline input, into the node with the key setting functions, of the Serial Key for the new registration equipment, encrypting the User Key (secure communication common key) using the Serial Key, and then sending it to the new registration equipment using an "authentication and encryption" type message. The initial User Key (secure communication common key) setting method is as specified in Part X, Chapter 5.

The initial Service Provider Key (secure communication common key) setting shall be made by encrypting the Service Provider Key (secure communication common key) using the User Key (secure communication common key) and sending it to the new registration equipment using an "authentication and encryption" type message.

User Keys and Service Provider Keys (secure communication common keys) shall be periodically updated using the existing common keys to encrypt the new common keys.

In principle, the Maker Key (secure communication common key) setting shall be made (built into the nodes) by the manufacturers before shipment, with secure communication common keys managed by manufacturers (manufacturer key index (MKI)-specific management) in such a manner that secure communication common keys are not disclosed. Maker Keys (secure communication common keys) shall not be updated.

The ECHONET secure communication specifications in this version of the ECHONET Specifications presuppose that there is only one node with the key setting functions in each domain.

10.8.3 Secure Communication Common Key (User Key) Setting Sequence

Figure 10.19 shows the secure communication common key (User Key) setting sequence. The initial User Key (secure communication common key) setting shall be made after shifting the new registration equipment to the common key initial setting mode. The new registration equipment shall respond to "authentication and encryption" type authentication requests (using its Serial Key) only when it is in the common key initial setting mode.

Figure 10.18 shows the secure communication common key (User Key) setting sequence and ECHONET secure frame. The secure communication common key (User Key) setting shall use supervisor-level authentication.

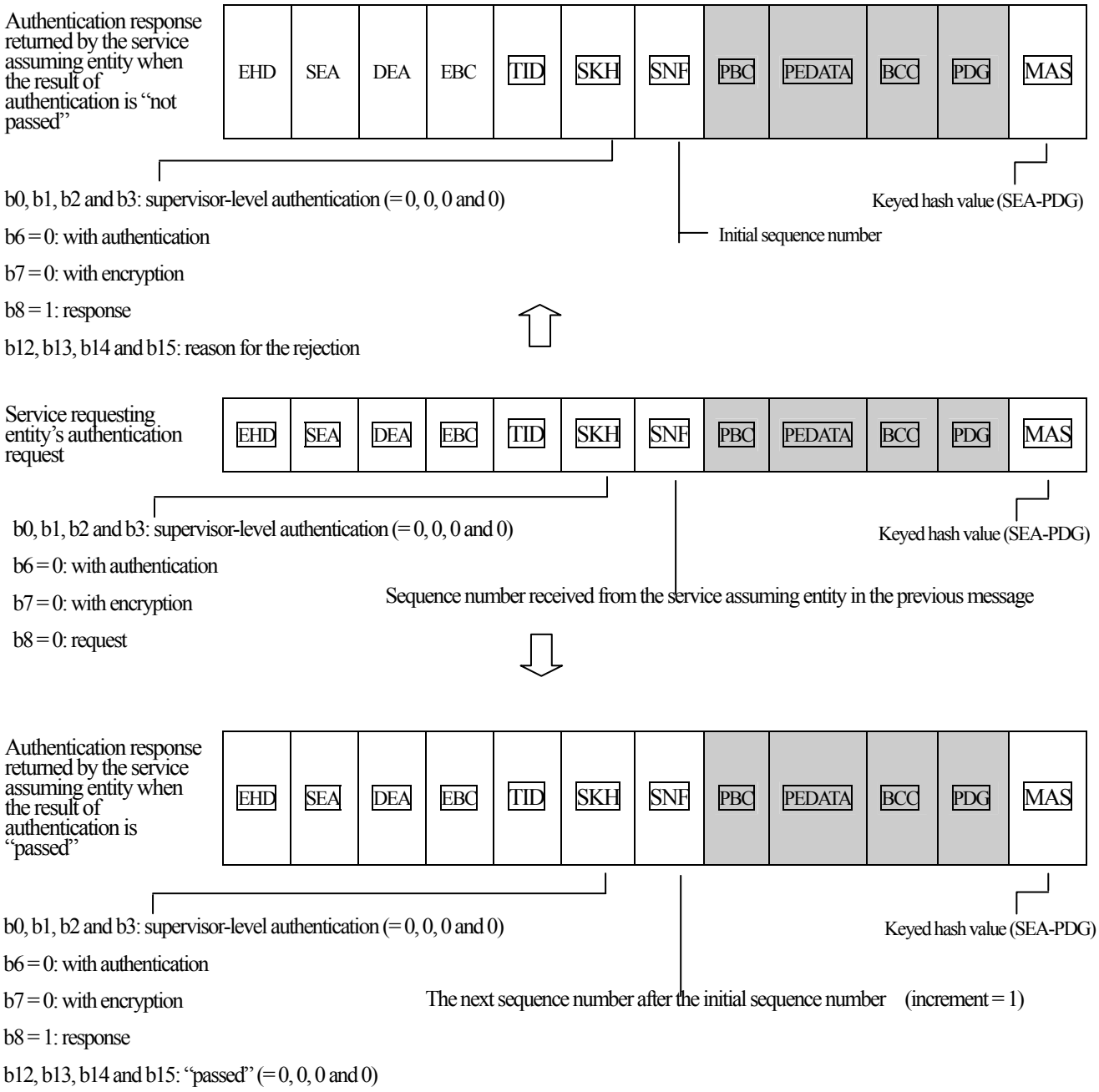


Fig. 10.18 ECHONET Secure Frame Used for Secure Communication Common Key (User Key) Setting

A new registration device shall determine the initial sequence number using a random number generation algorithm when it cold-starts. The node with the key setting functions shall perform a read from the node's secure communication common key setting (Serial Key) property for an encryption method that uses a Serial Key.

The node with the key setting functions shall create a common key and write it into the secure communication common key setting (User Key) property of the new registration equipment using a Serial

Key-based encryption and authentication type message.

The new registration equipment shall perform the authentication processing using its Serial Key.

If the result of the authentication is "passed," the new registration equipment shall acquire the common key (User Key) by decrypting the common key encrypted using its Serial Key. If the result of the authentication is "passed," the new registration equipment shall increase the sequence number (SNF) by 1, create an authentication response message using its Serial Key, and send it to the node that has the key setting functions.

If the result of the authentication is "not passed," the new registration equipment shall create a MAS from the SEA, DEA, EBC, TID, SKH, PBC, PEDATA, BCC, PDG, common key and the initial value of SNF and send a "not passed" authentication response that contains the MAS and the initial value of SNF.

Upon receiving the "not passed" authentication response, the node with the key setting functions shall create a MAS from the received SNF, SEA, DEA, EBC, TID, SKH, PBC, PEDATA, BCC, PDG and common key and send an "encryption and authentication" type message that contains the MAS and SNF received to the new registration equipment.

If no authentication response is received, the node with the key setting functions shall send the new registration equipment an "encryption and authentication" type message containing the MAS and SNF it sent to the new registration equipment in the previous message.

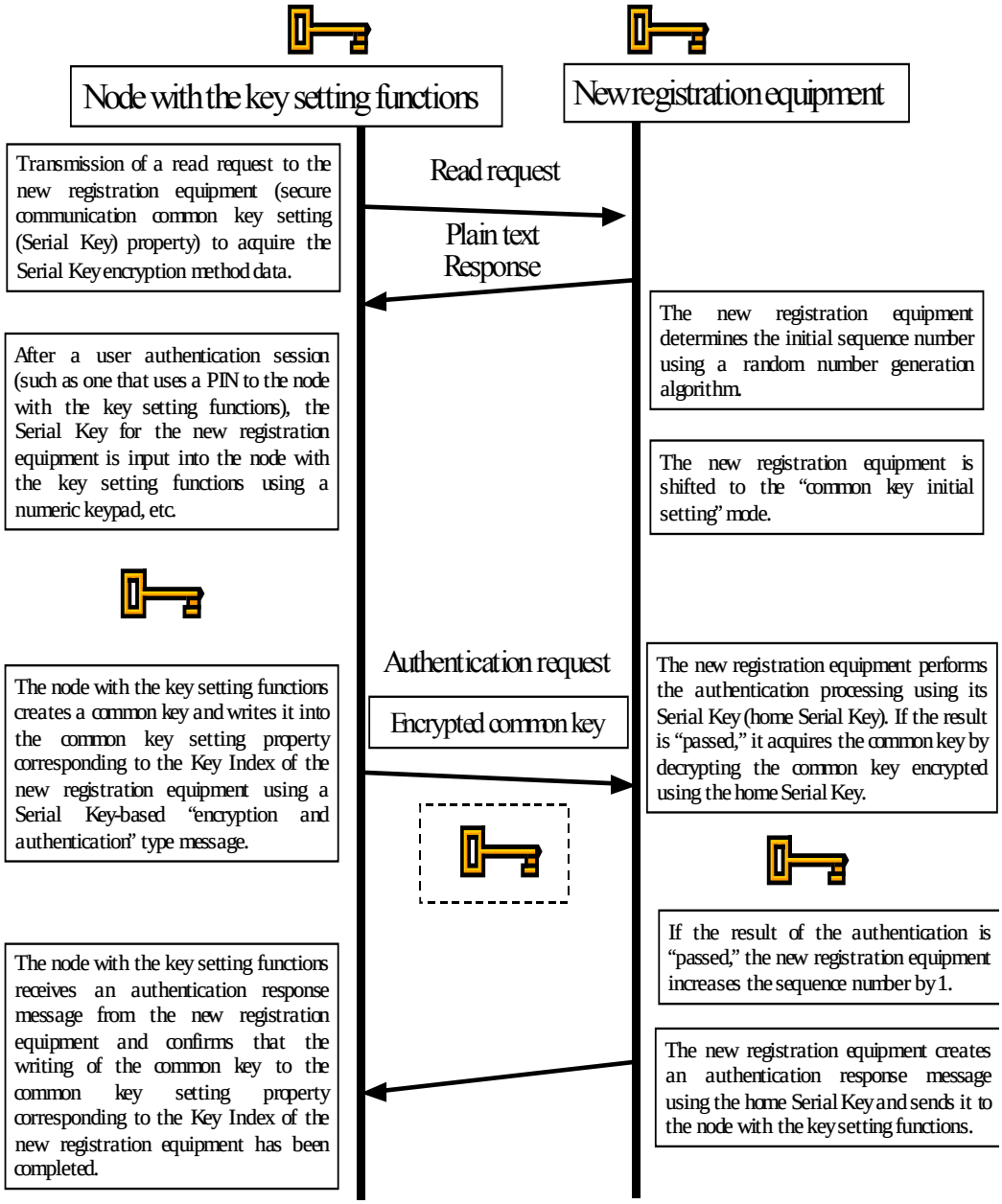


Fig. 10.19 Secure Communication Common Key (User Key) Setting Sequence

10.8.4 Secure Communication Common Key (Service Provider Key) Setting Sequence

Figure 10.21 shows the secure communication common key (Service Provider Key) setting sequence. The node with the key setting functions shall write the secure communication common key (Service Provider Key) into the secure communication common key setting (Service Provider Key) property of the node profile object implemented in the new registration equipment by sending an "authentication and

encryption" type message using the User Key.

Figure 10.20 shows the secure communication common key (Service Provider Key) setting sequence and ECHONET secure frame. The secure communication common key (Service Provider Key) setting shall use user-level authentication.

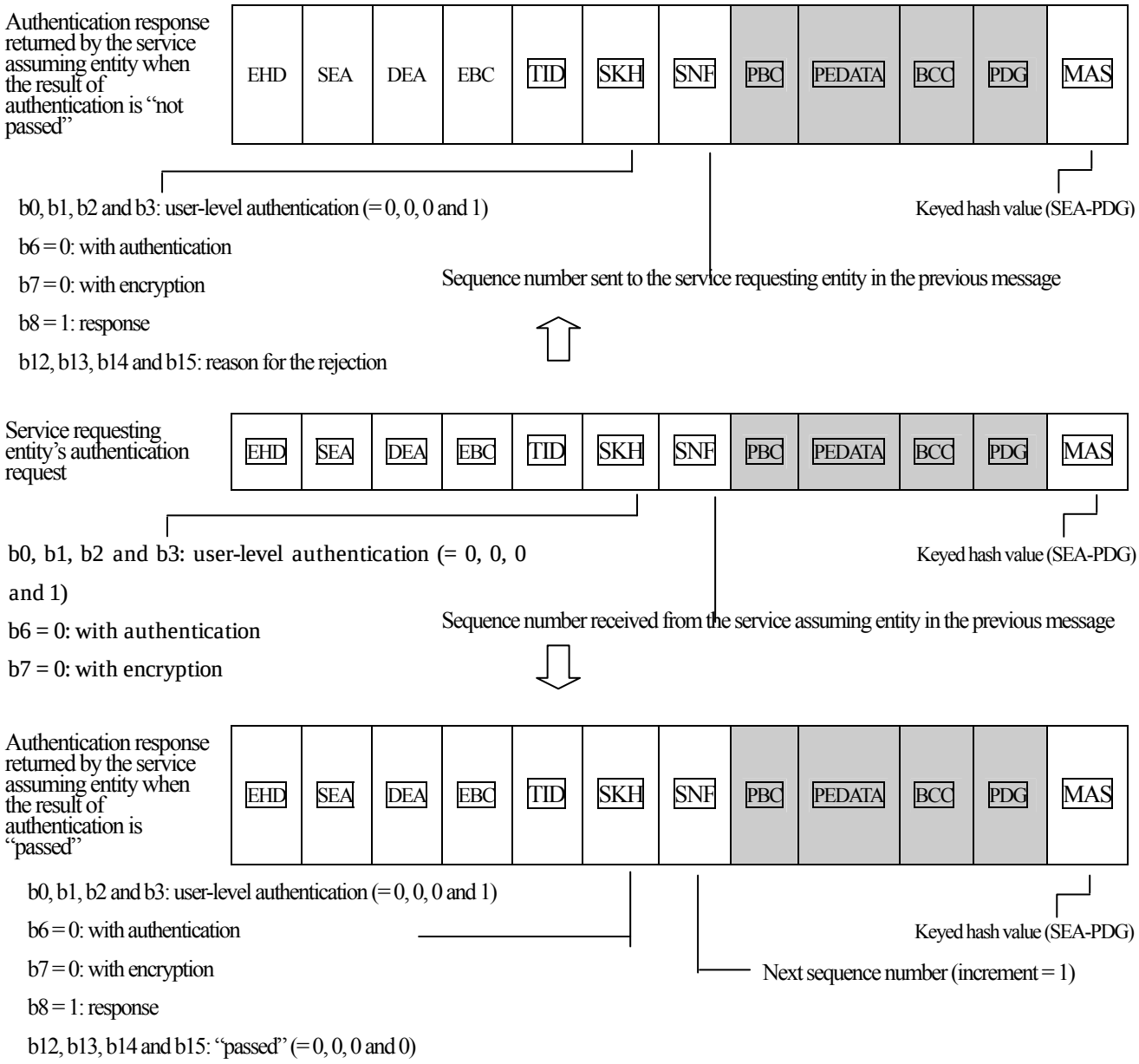


Fig. 10.20 ECHONET Secure Frame Used for Secure Communication Common Key (Service Provider Key) Setting

The node with the key setting functions shall create a common key (Service Provider Key) and write it into the new registration equipment's secure communication common key setting (Service Provider Key) property by sending an "encryption and authentication" type message using the User Key.

The new registration equipment performs the authentication processing using the User Key.

If the result of the authentication is "passed," the new registration equipment shall acquire the secure communication common key (Service Provider Key) by decrypting the secure communication common key encrypted using the User Key. The new registration equipment shall increase the sequence number (SNF) by 1, create an authentication response message using the User Key, and send it to the node with the key setting functions.

If the result of the authentication is "not passed," the new registration equipment shall create a MAS from the common key, DEA, EBC, TID, SKH, PBC, PEDATA, BCC, PDG, SEA and SNF sent to the node with the key setting functions in the previous message and send a "not passed" authentication response that contains the MAS and SNF sent to the node with the key setting functions in the previous message.

Upon receiving the "not passed" authentication response, the node with the key setting functions shall create a MAS from the received SNF, SEA, DEA, EBC, TID, SKH, PBC, PEDATA, BCC, PDG and common key and send an "encryption and authentication" type message that contains the MAS and SNF received to the new registration equipment.

If no authentication response is received, the node with the key setting functions shall send the new registration equipment an "encryption and authentication" type message containing the MAS and SNF it sent to the new registration equipment in the previous message.

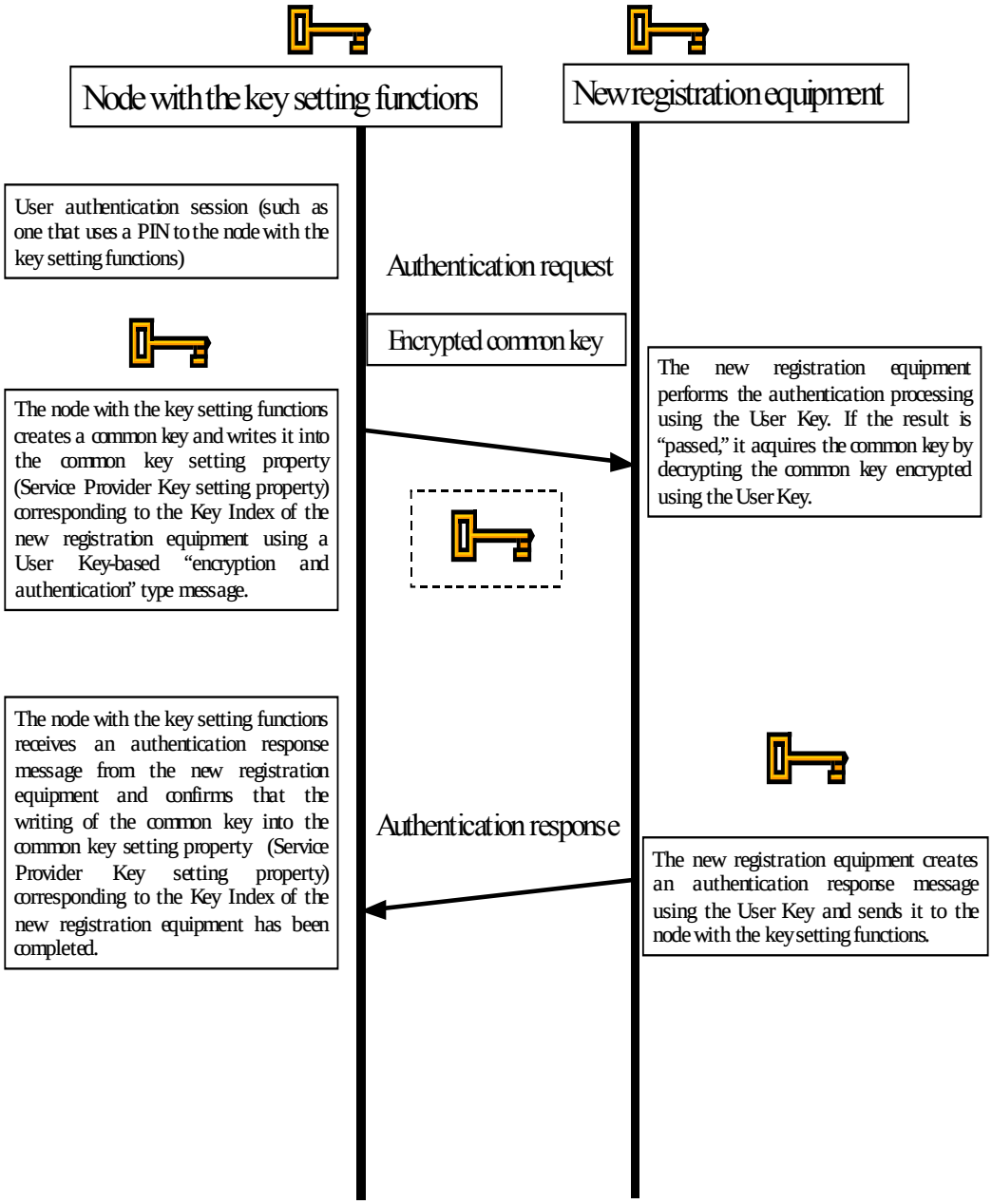


Fig. 10.21 Secure Communication Common Key (Service Provider Key) Setting Sequence

10.8.5 Secure Communication Common Key (Maker Key) Setting

In principle, the Maker Key (secure communication common key) setting shall be made (built into the nodes) by the manufacturers before shipment, with secure communication common keys managed by the manufacturers in such a manner that they are not disclosed.

The secure communication common key (Maker Key) shall be held by the application software at the

controller that manages and controls the nodes (devices) and be used to achieve secure communication common key (Maker Key)-based secure communication accesses to nodes (devices).

The method used to input secure communication common key (Maker Key) settings into nodes is not defined herein.

10.8.6 Common Key Delivery Method

Secure communication common keys shall be periodically delivered using the existing common keys to encrypt the new common keys. Figure 10.23 shows the secure communication common key delivery sequence and Fig. 10.22 shows the ECHONET secure frame and authentication sequence.

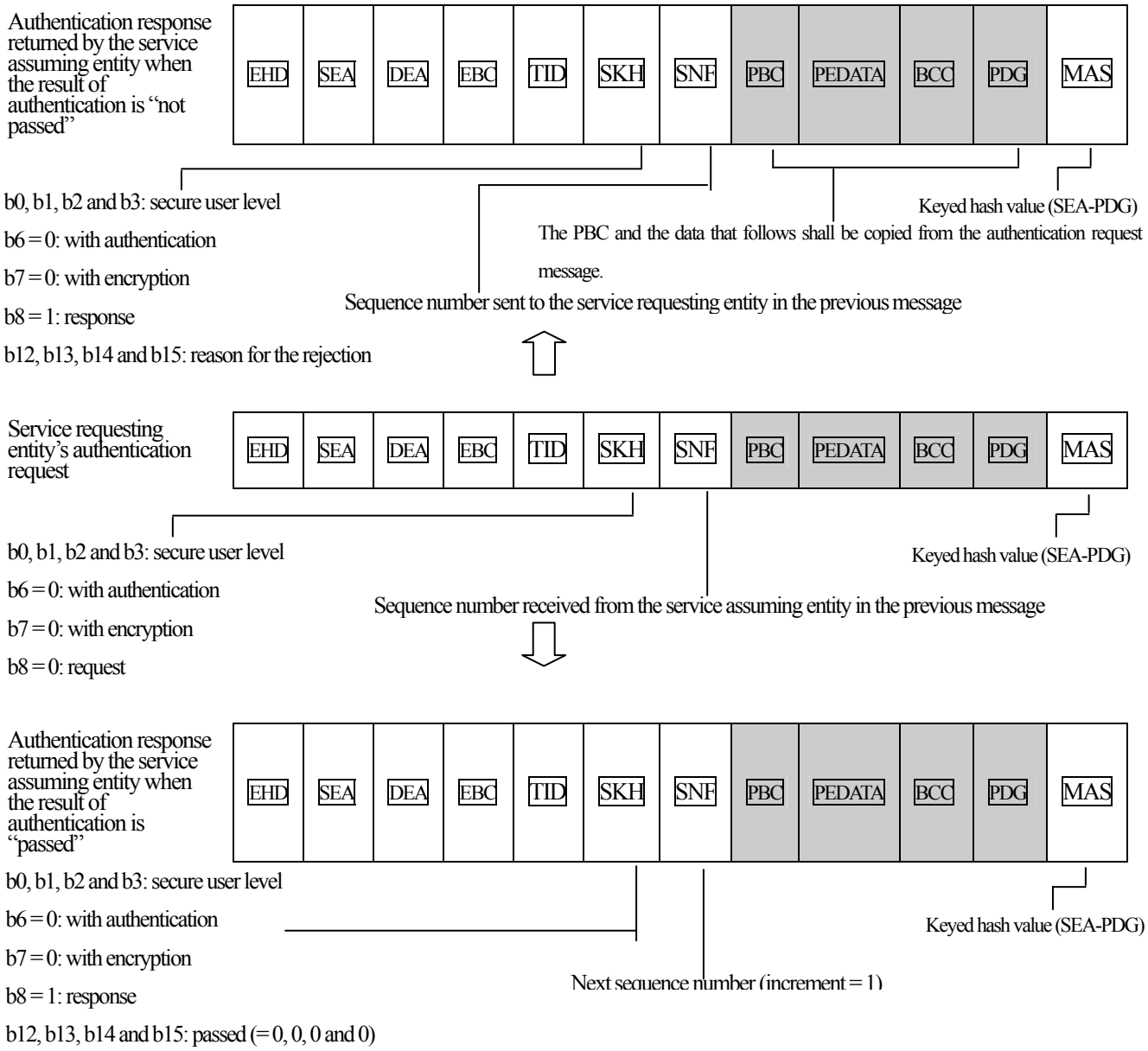


Fig. 10.22 ECHONET Secure Frame Used to Deliver Common Keys

The node with the key setting functions shall create a new common key (New Master Key) and write it into the equipment's common key setting property by sending an "encryption and authentication" type message using the common key (Pre Master Key).

The equipment shall perform the authentication processing using the common key (Pre Master Key).

If the result of the authentication is "passed," the equipment shall acquire the new common key by decrypting the new common key (New Master Key) using the Pre Master Key (common key). It shall send a "passed" authentication response to the service requesting entity, with the sequence number (SNF) (the one sent to the service requesting entity in the previous message) increased by 1 and stored (the response shall be in the form of an "authentication and encryption" type message using the common key (Pre Master Key)).

If the result of the authentication is "not passed," the equipment shall create a MAS from the common key (Pre Master Key), DEA, EBC, SKH, PBC, PEDATA, BCC, PDG, SEA and SNF sent to the node with the key setting functions in the previous message and send the node with the key setting functions a common key (Pre Master Key)-based "authentication and encryption" type "not passed" authentication response message containing the MAS and SNF sent to the node with the key setting functions in the previous message.

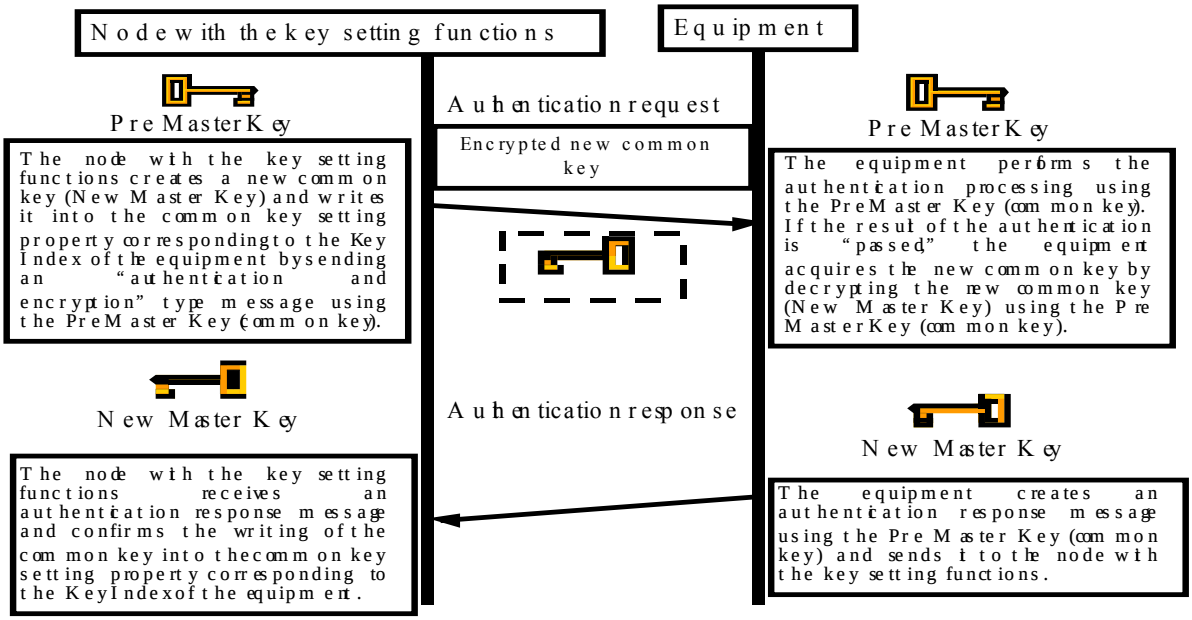


Fig. 10.23 Common Key Delivery Method

10.8.7 Common Key Update Synchronization

Even in a case where the node with the key setting functions periodically updates the secure communication common key, the timing of acquisition of the new common key (New Master Key) differs between devices in the domain, because the node with the key setting functions sends the secure communication common key to individual devices by means of individually transmitted "authentication and encryption" type messages.

For this reason, "shift in progress" and "update completed" status indication writes to the common key change property of the node profile object implemented in the equipment shall be made (using "authentication and encryption" type messages) when the shift from Pre Master Key to New Master Key starts and when the updating of Pre Master Key to New Master Key is completed, respectively, as a means of synchronizing among the nodes in the domain the common key update from Pre Master Key to New Master Key, in addition to the write of the common key to the common key setting property of the node profile object implemented in the equipment explained in "10.7.3 Updating Common Keys."

The node with the key setting functions shall send the new secure communication common key (New Master Key) to the ordinary nodes it manages after encrypting it using the Pre Master key. If an ordinary node receives a new secure communication common key (New Master Key) setting request, the common key change setting property shall shift to "delivery complete."

After sending the secure communication common key (New Master Key) to all ordinary nodes that it manages, the node with the key setting functions shall set the EDT of "secure communication common key change setting" to "shift in progress," encrypt it using the New Master Key and send it to the ordinary nodes that have successfully completed the secure communication common key (New Master Key) setting. If an ordinary node receives a secure communication common key change setting message, the common key change setting property shall shift to "shift in progress."

If the new secure communication common key setting completes successfully in all ordinary nodes managed by the node with the key setting functions, the node with the key setting functions shall set the EDT of "secure communication common key change setting" to "update complete" for all ordinary nodes that it manages, encrypt it using the New Master Key and send it to the applicable ordinary nodes.

The sequence is as shown in Fig. 10.24.

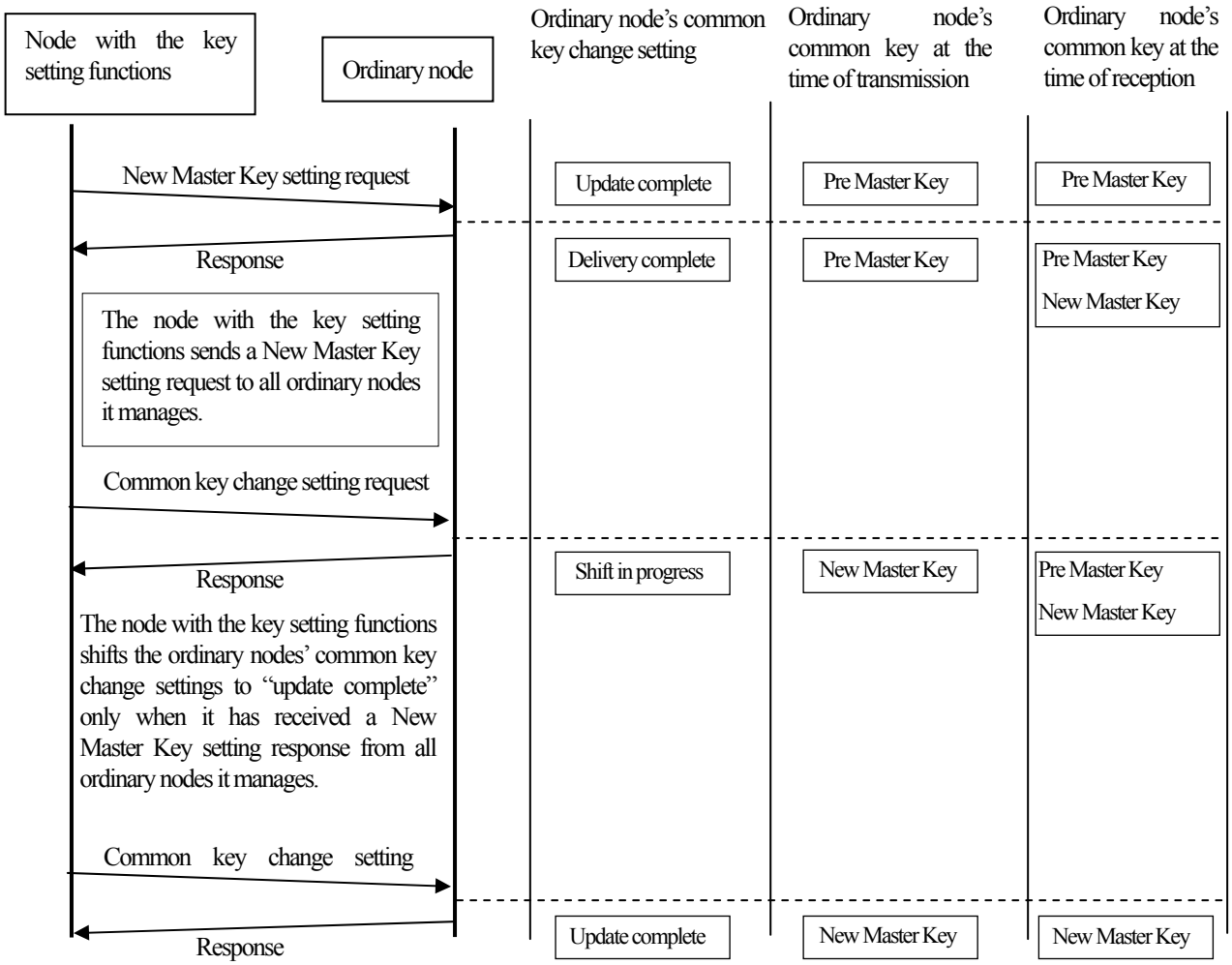


Fig. 10.24 Common Key Update Synchronization

10.8.8 Method for Ensuring Updated Common Keys of All Devices Unplugged During a Common Key Update

Attempts are made sequentially in the order of the Serial Key list maintained by the node with the key setting functions. If a device is left unplugged for a long period of time, a common key generation change may occur during that period.

Therefore, it is necessary to manage the devices' common keys with respect to their generations and set common keys appropriately in the devices having common keys of different generations by encrypting new common keys with the common keys of the appropriate generations for such devices and sending them to such devices in accordance with the common key delivery method explained in Fig. 10.23.

A device shall perform the common key update sequence immediately after it warm-starts. A common key setting request property write request shall be encrypted using the Pre Master Key and be sent as an "authentication and encryption" type message to the common key delivery request property of the class having the key setting functions.

If the node with the key setting functions receives a write request to the common key delivery request object, it shall perform the secure communication common key delivery sequence explained in "10.8.6

Common Key Delivery Method."

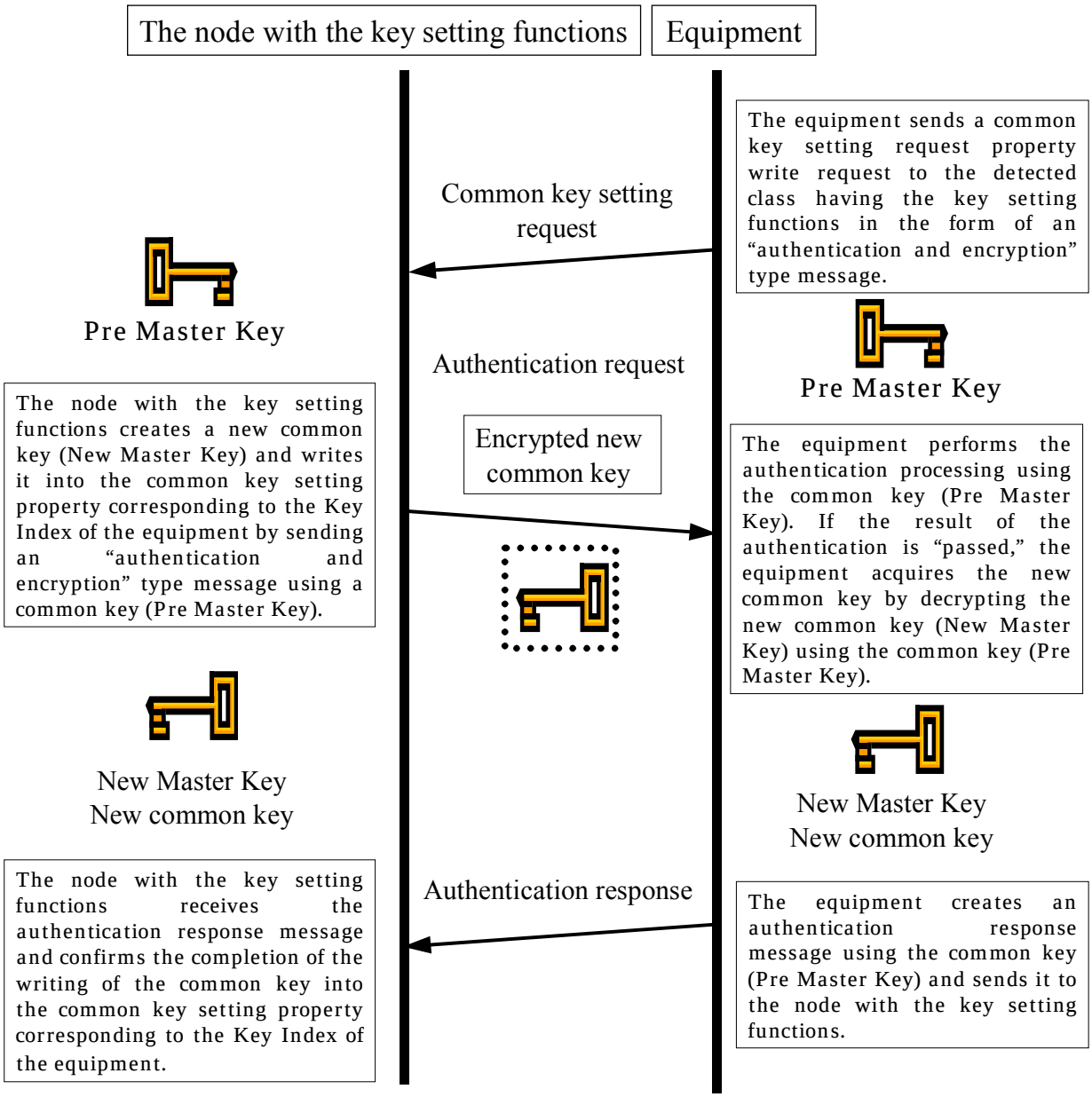


Fig. 10.25 Method for Ensuring Updated Common Keys of All Devices Unplugged During a Common Key Update

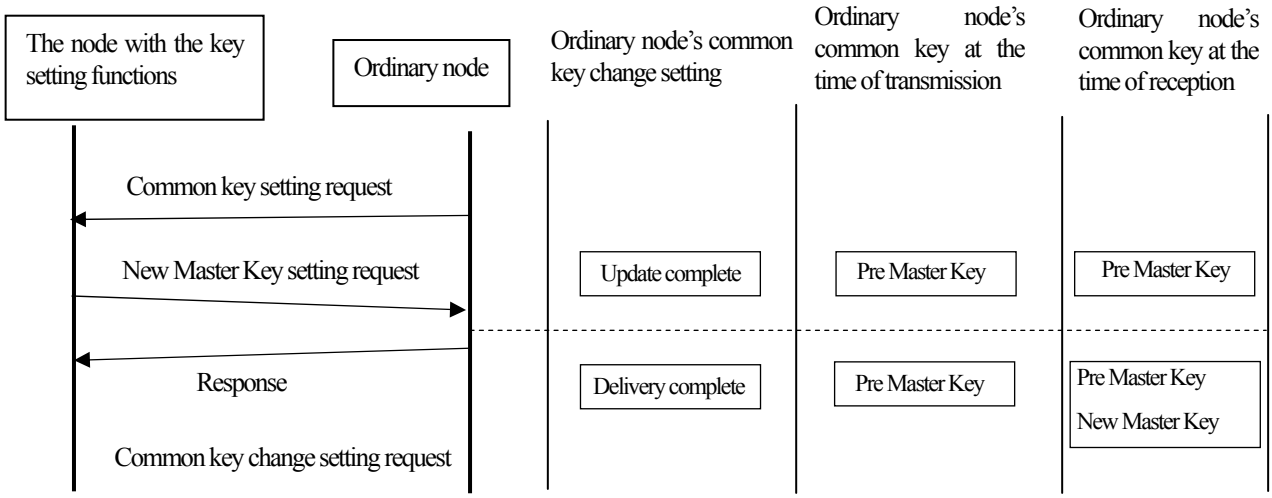


Fig. 10.26 Sequence for Ensuring Updated Common Keys of All Devices Unplugged During a Common Key Update

To ensure that the common keys of all devices that were unplugged during a common key update are updated, ordinary nodes shall send, at the time of the startup, a common key setting request to the node with the key setting functions after encrypting it using the Pre Master Key.

The node with the key setting functions shall send a New Master Key to the ordinary nodes from which it received common key setting requests encrypted with the Pre Master Key. The following sequence is the same as that shown in Fig. 10.26.

10.9 Node Profile Property Requirement for ECHONET Secure Communication

The "common key for secure communication" and "transfer of common key for secure communication" setting properties are stipulated in the node profile class to be used for initial setting and updating of the ECHONET secure communication common key. These properties are essential for mounting secure communication.

For details of these properties, refer to "9.11.1 Detailed Specifications of Node Profile Class".

10.10 Access Limitation

In ECHONET secure communication, access to the properties of the ECHONET object of the requested party is limited based on the authentication level of the ECHONET object of the requesting party. In the node of the requesting party, access is limited in a different manner by the ECHONET object of the requested party.

There are four levels of authentication:

- Supervisor authentication
- User Level authentication

- Maker Level authentication
- Service Provider Level authentication

ECHONET secure communication includes the following five access limit levels corresponding to the four authentication levels listed above and cases of no authentication. Not all access limit levels need be supported in the mounting mode.

- Access limit level when inhabitant changes access rules for ECHONET device (Supervisor Level): Access is permitted only to objects with Supervisor authentication.
- Access limit level to device used by inhabitant (User Level): Access is permitted only to objects with User Level authentication.
- Access limit level to device maker (Maker Level): Access is permitted only to objects with Maker Level authentication.
- Access limit level to application user entrusted by the inhabitant (Service Provider Level): Access is permitted only to objects with Service Provider Level authentication.
- Access limit level without authentication (Anonymous Level): Access is permitted from any object, without authentication.

Access rules corresponding to each access limit level are determined and set when the ECHONET device is developed or when system operation is designed during the installation of each ECHONET object mounted on the ECHONET node.

Accessible properties (based on the authentication level of each device object property), the service object, the profile object, and the communication definition object shall be set using “9.17 Secure Communication Access Property Setting Class”.

Thus, access to the ECHONET object side of the requested party by the ECHONET object of the requesting party is limited by the self-requesting authentication level. This results in a different view of the ECHONET object of the requested party. Figure 10.23 shows an example of mounting a device object under four access limit levels.

The present version does not stipulate access rules with individual ECHONET objects.

The ECHONET node on the request-receiving side must be authenticated by separately controlling the key by access limit level and by authentication key index. This means that, for example, if a number of service providers are controlled, the ECHONET node must separately control and authenticate each service provider.

In addition, authentication must be performed individually by the ECHONET object of the requesting party.

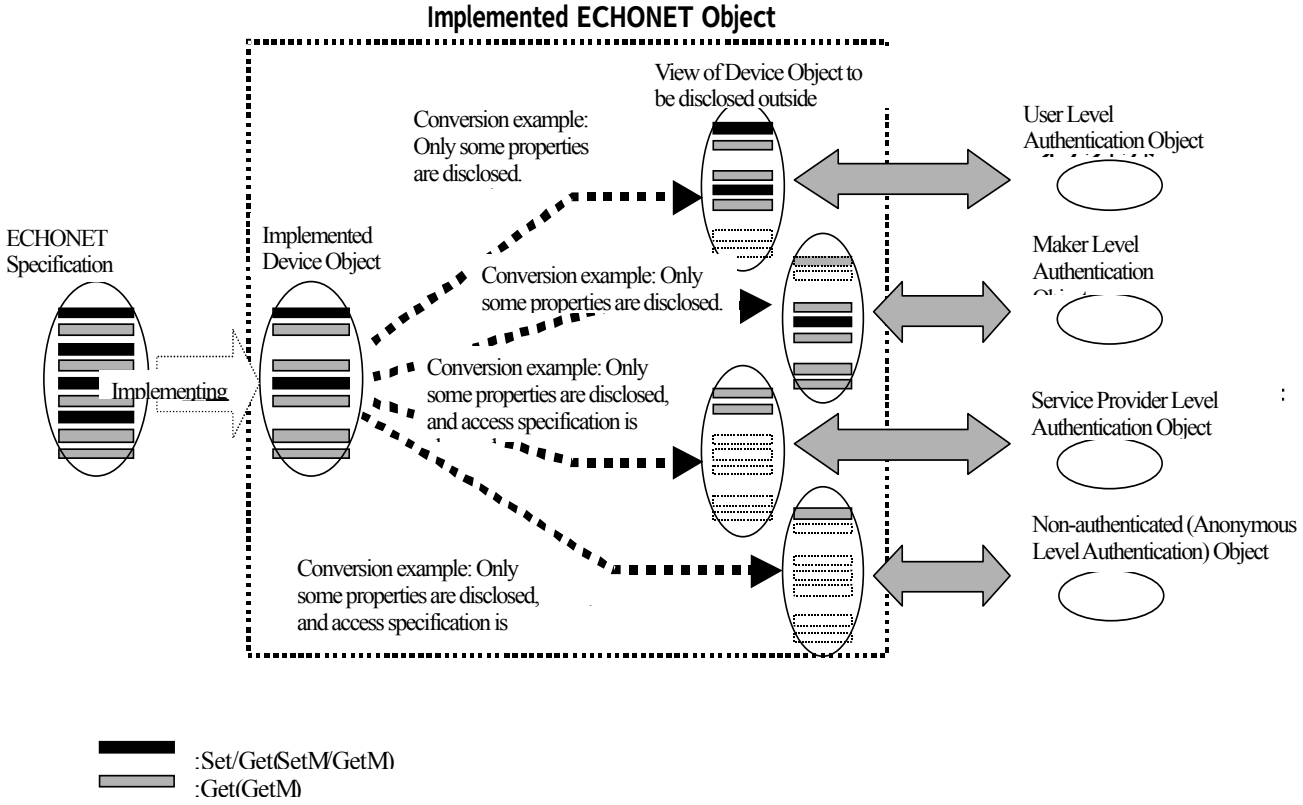


Fig. 10.27 Mounting a Device Object Under Four Access Limitation Levels (Example)

As described above, the ECHONET object of the requesting party is viewed differently by the ECHONET object of the requested party based on the self-requested authentication level. The following items are stipulated for the property map specified as a property of the ECHONET object in order to understand which access rules are stipulated in the ECHONET object of the requested party. The following requirements shall be applied if secure communication is supported.

When the Get service request is received for Set Property Map, Get Property Map, SetM Property Map, and GetM Property Map, a list of the properties applicable to Set, Get, SetM, and GetM is prepared on the authentication level of the Get service request message and returned as the property map. The format of the responding property shall follow the descriptive format of the property map described in the appendix to Part 2.

Specific examples are described below.

It is assumed that the device object shown in Fig. 10.24 is mounted on the device. The individual mounting the device designs the access rules shown in Fig. 10.25. However, note that the Set and Get rules shown in the figure are only examples.

It is also assumed that the object access rules follow Fig. 10.25. When the service request is actually received, the service requesting party object is authenticated, and it is determined whether or not the service request has been received according to the access rules for the given authentication level.

In addition, access rules and the property map vary with the authentication level. This means that if the reading service is requested for Set Property Map, Get Property Map, SetM Property Map, and GetM Property Map based on the authentication level, the property map for the given authentication level is returned as a response. Figure 10.27 shows the content of the response in the above example.

Example of Device Object — (Some are different from the actual EPC.)

	Property code	Property content
Place of installation	0x81	Living room (0x09)
Maker code	0x8A	0x000000
Present transmission (w)	0xE8	0x004F
Alarm threshold transmission (w)	0xE9	0x00FF
Watt-hour (kWh)	0xE0	0x11223344
Error code for maintenance	0xF0	0x0000
Set Property Map	0x9E	Described below
Get Property Map	0x9F	Described below

Property value to be returned varies with the authentication level at which access is made.

Fig.10.28 Device Object (Example)

		Support Access Specification			
Property Code	Access Specifi-cation	Anonymous Level	User Level	Maker Level	Service Provider Level
0X81	Get	Get	Get	Get	Get
0X8A	Get	–	Get	Get	–
0XE8	Get	–	Get	Get	Get
0XE9	Set/Get	–	Set/Get	–	Get
0XE0	Get	–	Get	–	Get
0XF0	Get	–	–	Get	–
0X9E	Get	Get	Get	Get	Get
0X9F	Get	Get	Get	Get	Get

The UserLevel/MakerLevel access rule is embedded in the device.

Fig. 10.29 Mounting Access Specification (Example)

Authentication Level and Response Set Property Map

Anonymous Level	0x00
User Level	0x01,{0xE9}
Maker Level	0x00
Service Provider Level	0x00

Authentication Level and Response Get Property Map

Anonymous Level	0x03,{0x81,0xpE,0x9F}
User Level	0x07,{0x81,0x8A,0xE8,0xE9,0xE0,0x9E,0x9F}
Maker Level	0x06,{ 0x81,0x8A,0xE8,0xF0,0x9E,0x9F }
Service Provider Level	0x06,{ 0x81,0xE8, 0xE9,0xE0,0x9E,0x9F }

Fig. 10.30 Response Property Map Content Based on Authentication Level

10.11 Secure Communication Access Property Setting Class Group

See "9.17 Requirements for the Secure Communication Access Property Setting Class Group."

Appendix 1 References

(1) *EIAJ ET-2101 Home Bus System*, Electronic Industries Association of Japan

Technical Division
Electronic Industries Association of Japan
Tel: +81-3-3213-1075

(2) *EIAJ ET-2101 Home Bus System (Addendum)*, Electronic Industries Association of Japan

Technical Division
Electronic Industries Association of Japan
Tel: +81-3-3213-1075

(3) *EIAJ RC-5202 Data Outlet for Home Bus System*, Electronic Industries Association of Japan

Technical Division
Electronic Industries Association of Japan
Tel: +81-3-3213-1075

(4) *JEM 1439 Housekeeping Command Code Assignment for Use in Home Bus System*, Electronic Industries Association of Japan

General Affairs Division
Electronic Industries Association of Japan
Tel: +81-3-3581-4841

Appendix 2 Property Map Description Format

When there are fewer than 16 properties, Description Format (1) below is followed; when there are 16 or more, Description Format (2) is followed.

Description Format (1)

Byte 1 : Number of properties. Displayed in binary.

Byte 2 and higher : List of property codes (1-byte code).

Description Format (2)

Byte 1 : Number of properties. Displayed in binary.

Bytes 2–17 : In the 16-byte table below, the bit location showing existing property codes is set to 1, and properties are listed in order starting with Byte 2.

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 2	80	90	A0	B0	C0	**D**0	**E**0	F0
Byte 3	81	91	A1	B1	C1	**D**1	**E**1	F1
Byte 4	82	92	A2	B2	C2	**D**2	**E**2	F2
Byte 5	83	93	A3	B3	C3	**D**3	**E**3	F3
Byte 6	84	94	A4	B4	C4	**D**4	**E**4	F4
Byte 7	85	95	A5	B5	C5	**D**5	**E**5	F5
Byte 8	86	96	A6	B6	C6	**D**6	**E**6	F6
Byte 9	87	97	A7	B7	C7	**D**7	**E**7	F7
Byte 10	88	98	A8	B8	C8	**D**8	**E**8	F8
Byte 11	89	99	A9	B9	C9	**D**9	**E**9	F9
Byte 12	8A	9A	AA	BA	CA	**D**A	**E**A	FA
Byte 13	8B	9B	AB	BB	CB	**D**B	**E**B	FB
Byte 14	8C	9C	AC	BC	CC	**D**C	**E**C	FC
Byte 15	8**D**	9**D**	A**D**	B**D**	C**D**	**D D**	**E D**	F**D**
Byte 16	8**E**	9**E**	A**E**	B**E**	C**E**	**D E**	**E E**	F**E**
Byte 17	8F	9F	AF	BF	CF	**D**F	**E**F	FF

Note: For each bit, 0 = no property; 1 = property exists.

The explanations below show how Property Map Description Format (2) is used, using an ECHONET node equipped with home air conditioner class (0x0130) as an example.

No.	Property name	EPC	Corresponding property map bit
1	Operating status	0x80	Bit 0 of the second byte
2	Installation location	0x81	Bit 0 of the third byte
3	Specification version information	0x82	Bit 0 of the fourth byte
4	Node identification number	0x83	Bit 0 of the fifth byte
5	Current limit setting	0x87	Bit 0 of the ninth byte
6	Fault status	0x88	Bit 0 of the tenth byte
7	Error description	0x89	Bit 0 of the eleventh byte
8	Manufacturer code	0x8A	Bit 0 of the twelfth byte
9	Factory code	0x8B	Bit 0 of the thirteenth byte
10	Product code	0x8C	Bit 0 of the fourteenth byte
11	Serial number	0x8D	Bit 0 of the fifteenth byte
12	Date of manufacture	0x8E	Bit 0 of the sixteenth byte
13	Power-saving operation setting	0x8F	Bit 0 of the seventeenth byte
14	On-timer reservation setting	0x90	Bit 1 of the second byte
15	Cumulative operation hours	0x9A	Bit 1 of the twelfth byte
16	SetM property map	0x9B	Bit 1 of the thirteenth byte
17	GetM property map	0x9C	Bit 1 of the fourteenth byte
18	Status change announcement property map	0x9D	Bit 1 of the fifteenth byte
19	Set property map	0x9E	Bit 1 of the sixteenth byte
20	Get property map	0x9F	Bit 1 of the seventeenth byte
21	Operation mode setting	0xB0	Bit 3 of the second byte
22	Temperature setting	0xB3	Bit 3 of the fifth byte

If the above-mentioned properties are published in the ECHONET node:

The first byte value is 0x16 since the number of properties is 22. The second byte value is 0x0B = b'00001011' since the 0x80, 0x90 and 0xB0 properties are published and the corresponding bits are bit 0, bit 1 and bit 3. The third, fourth, ninth, tenth and eleventh byte values are 0x01 since the 0x81, 0x82, 0x87, 0x88 and 0x89 properties are published and the corresponding bit is bit 0. The fifth byte value is 0x09 = b'00001001' since the 0x83 and 0xB3 properties are published and the corresponding bits are bit 0 and bit3. The twelfth to seventeenth byte values are 0x03 = b'00000011' since the 0x8A, 0x9A, 0x8B, 0x9B, 0x8C, 0x9C, 0x8D, 0x9D, 0x8E, 0x9E, 0x8F and 0x9F properties are published and the corresponding bits are bit 0 and bit 1.

Therefore, the property map description is "0x16, 0x0B, 0x01, 0x01, 0x09, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03."

Appendix 3 All Router Data Description Format

Byte 1 : Number of routers

Byte 2 and higher : The following router data set exists for all routers.

(Router data Byte 1: Router ID

Byte 2: Number of connected subnets (n)

Byte 3–[(2 * n)+2]: Held EA data (for n cases))

Appendix 4 Instance List Description Format

Relevant instance code location bits are set to 1, and non-relevant bits to 0. The relevant class code (most significant 2 bytes of EOJ) is stipulated as an array element number.

Self-node instance list page 1 (EPC = 0xD0) is for disclosing data for instance numbers 0x00–0x7F.

- Self-node instance list page 1 (EPC = 0xD0) description format
-

First byte: Total number of the instances of the class specified using the element-designation function (binary notation)

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 2	00	01	02	03	04	05	06	07
Byte 3	08	09	0A	0B	0C	0D	0E	0F
Byte 4	10	11	12	13	14	15	16	17
Byte 5	18	19	1A	1B	1C	1D	1E	1F
Byte 6	20	21	22	23	24	25	26	27
Byte 7	28	29	2A	2B	2C	2D	2E	2F
Byte 8	30	31	32	33	34	35	36	37
Byte 9	38	39	3A	3B	3C	3D	3E	3F
Byte 10	40	41	42	43	44	45	46	47
Byte 11	48	49	4A	4B	4C	4D	4E	4F
Byte 12	50	51	52	53	54	55	56	57
Byte 13	58	59	5A	5B	5C	5D	5E	5F
Byte 14	60	61	62	63	64	65	66	67
Byte 15	68	39	3A	3B	3C	3D	3E	3F
Byte 16	70	71	72	73	74	75	76	77
Byte 17	78	79	7A	7B	7C	7D	7E	7F

Note: For each bit, 0 = no property; 1 = property exists.

Bit 0 of the second byte that corresponds to all instance designation code (0x00) shall always be 0.

The explanations below show how the instance list description format is used, using as an example an ECHONET node equipped with 0x01, 0x05 and 0x10 (instance numbers) of the temperature sensor class (0x0011).

No.	Object name	Instance number	Corresponding instance list map bit
1	Temperature sensor (0x0011)	0x01	Bit 1 of the second byte
2	Temperature sensor (0x0011)	0x05	Bit 5 of the second byte
3	Temperature sensor (0x0011)	0x10	Bit 0 of the fourth byte

The instance list description for a response in the case where 0x0011 (temperature sensor class) is specified

using the element designation function for the above-mentioned ECHONET node is as follows:

The first byte value is 0x03 since the total number of the instances of the class specified using the element designation function is 3. The second byte value is 0x22 = b'00100010' since the bits that correspond to the instance numbers 0x01 and 0x05 are bit 1 and bit 5. The fourth byte value is 0x01 = b'00000001' since the bit that corresponds to the instance number 0x10 is bit 0.

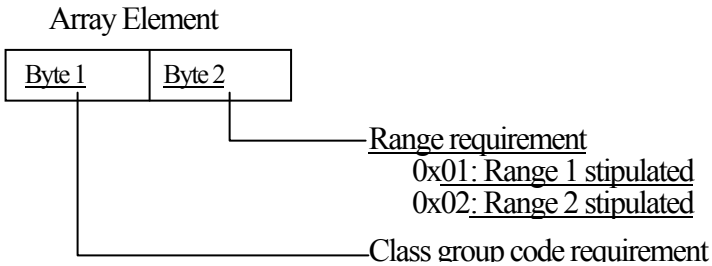
Therefore, the EDT in the above example for the case where 0x0011 (temperature sensor class) is specified using the element designation function is "0x03, 0x22, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00."

Appendix 5 Class List Description Format

The relevant class code location bits of EOJ Byte 2 are set to 1, and the non-relevant bits are set to 0.

When Range 1 is stipulated in the element, the format shown in (2) below is used.

The relevant class group code (most significant byte of EOJ) is stipulated as the most significant byte of the array element number, and the aforementioned range is stipulated by the least significant byte (see diagram below).



Format when Range 1 is stipulated

Byte 1: Number of classes belonging to stipulated class group

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 2	00	01	02	03	04	05	06	07
Byte 3	08	09	0A	0B	0C	0D	0E	0F
Byte 4	10	11	12	13	14	15	16	17
Byte 5	18	19	1A	1B	1C	1D	1E	1F
Byte 6	20	21	22	23	24	25	26	27
Byte 7	28	29	2A	2B	2C	2D	2E	2F
Byte 8	30	31	32	33	34	35	36	37
Byte 9	38	39	3A	3B	3C	3D	3E	3F
Byte 10	40	41	42	43	44	45	46	47
Byte 11	48	49	4A	4B	4C	4D	4E	4F
Byte 12	50	51	52	53	54	55	56	57
Byte 13	58	59	5A	5B	5C	5D	5E	5F
Byte 14	60	61	62	63	64	65	66	67
Byte 15	68	39	3A	3B	3C	3D	3E	3F
Byte 16	70	71	72	73	74	75	76	77
Byte 17	78	79	7A	7B	7C	7D	7E	7F

Note: For each bit, 0 = no instance; 1 = instance exists.

Format when Range 2 is stipulated

Byte 1: Number of classes belonging to stipulated class group

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 2	**80**	**81**	**82**	**83**	**84**	**85**	**86**	**87**
Byte 3	**88**	**89**	**8**A	**8**B	**8**C	**8**D	**8**E	**8**F
Byte 4	**90**	**91**	**92**	**93**	**94**	**95**	**96**	**97**
Byte 5	**98**	**99**	**9**A	**9**B	**9**C	**9**D	**9**E	**9**F

Byte 6	A0	A1	A2	A3	A4	A5	A6	A7
Byte 7	A8	A9	AA	AB	AC	AD	AE	AF
Byte 8	B0	B1	B2	B3	B4	B5	B6	B7
Byte 9	B8	B9	BA	BB	BC	BD	BE	BF
Byte 10	C0	C1	C2	C3	C4	C5	C6	C7
Byte 11	C8	C9	CA	CB	CC	CD	CE	CF
Byte 12	D0	D1	D2	D3	D4	D5	D6	D7
Byte 13	D8	D9	DA	DB	DC	DD	DE	DF
Byte 14	E0	E1	E2	E3	E4	E5	E6	E7
Byte 15	E8	E9	EA	EB	EC	ED	EE	EF
Byte 16	F0	F1	F2	F3	F4	F5	F6	F7
Byte 17	F8	F9	FA	FB	FC	FD	FE	FF

Note: For each bit, 0 = no instance; 1 = instance exists.

The explanations below show how the class list description format is used, using as an example an ECHONET node equipped with four ECHONET objects; "human body detection sensor class" (0x0007), "illumination sensor class" (0x000D), "temperature sensor class " (0x0011) and "humidity sensor class" (0x0012).

No.	Object name	Class group	Field	Corresponding class list map bit
1	Human body detection sensor class (0x0007)	0x00	0x01	Bit 7 of the second byte
2	Illumination sensor class (0x000D)	0x00	0x01	Bit 5 of the third byte
3	Temperature sensor class (0x0011)	0x00	0x01	Bit 1 of the fourth byte
4	Humidity sensor class (0x0012)	0x00	0x01	Bit 2 of the fourth byte

The class list description in the case where the above class list is implemented is as follows:

EDT of a response in the case where 0x0001 (field 1 (class group 0x00)) is specified using the element designation function:
The first byte value is 0x04 since the total number in the class group (0x00) specified using the element designation function is 4. The second byte value is 0x80 = b'10000000' since the bit 7 value is 1 as a result of the implementation of "human body detection sensor" (0x0007). The third byte value is 0x20 = b'00100000' since the bit 5 value is 1 as a result of the implementation of "illumination sensor" (0x000D). The fourth byte value is 0x06 = b'00000110' since the bit 1 and bit 2 values are 1 as a result of the implementation of "temperature sensor" (0x0011) and "humidity sensor" (0x0012).

Therefore, the EDT of a response in the case where 0x0001 (field 1 (class group 0x00)) is specified using the element designation function is "0x04, 0x80, 0x20, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,0x00, 0x00, 0x00."

Appendix 6 NetID Server Startup Sequence

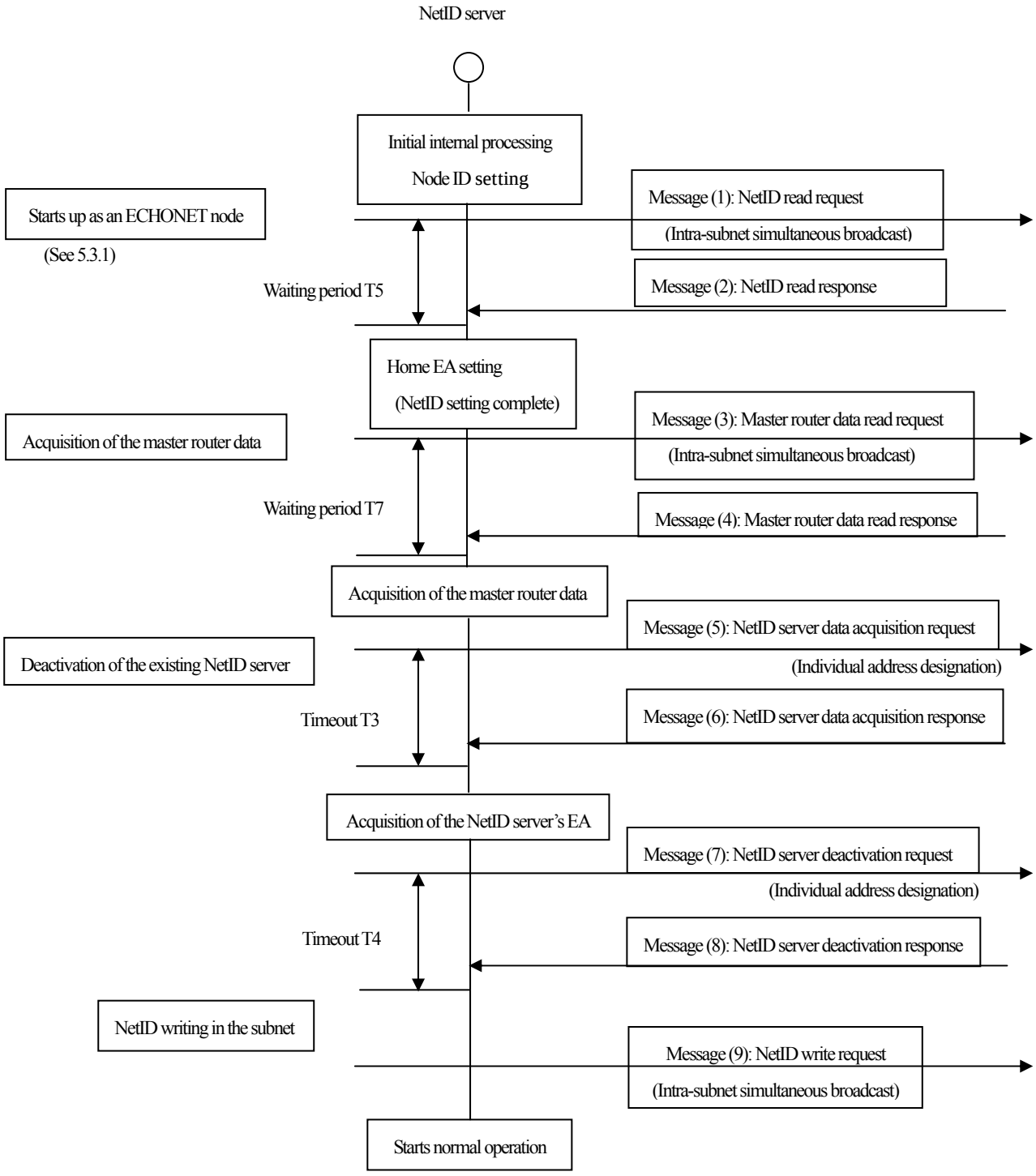


Fig. A6-1 Basic Sequence for NetID Server Cold Start

Table A6-1 Summary of Message Contents Used in the Basic Sequence for NetID Server Cold Start

Message (1)	NetID read request * SEA value setting (home EA's NetID → 0x00) * Setting for intra-subnet simultaneous broadcast (0x01FF)(DEA) * Setting for node profile object (0x0EF001)(DEOJ) * Setting for NetID property (0xE1)(EPC) * Setting for read request (0x62)(ESV)
Message (2)	Response to Message (1) Waits for the reception of Message (2) for the T5 period. If Message (2) is not received within the T5 period, the NetID server shall set a new NetID in the NetID assignment code area and shift to normal operation.
Message (3)	Master router data read request * Setting of the home EA value as seen from the subnet (SEA) * Setting for intra-subnet simultaneous broadcast (0x01FF)(DEA) * Setting for router profile object (0x0EF101)(DEOJ) * Setting for master router data property (0xE6)(EPC) * Setting for read request (0x62)(ESV)
Message (4)	Response to Message (3) Waits for the reception of Message (4) for the T7 period. If Message (4) is not received within the T7 period, it shall be presumed that no master router exists in the subnet and Messages (5) and (7) shall not be transmitted.
Message (5)	NetID server data acquisition request * Setting of the home EA value as seen from the subnet (SEA) * Setting of the master router's EA value acquired from Message (4) (DEA). * Setting for router profile object (0x0EF101)(DEOJ) * Setting for NetID server data (0xE3)(EPC) * Setting for read request (0x62)(ESV)
Message (6)	Response to Message (5) If Message (6) is received within the T3 period, Message (7) shall be transmitted immediately. If Message (6) is not received within the T3 period, Message (7) shall not be transmitted.
Message (7)	NetID server deactivation request * Setting of the home EA value as seen from the subnet (SEA) * Setting of the existing NetID server's EA acquired from Message (6) (DEA) * Setting for NetID server profile object (0x0EF501)(DEOJ) * Setting for operation status (0x80)(EPC) * Setting for write request requiring a response (0x61)(ESV) * Setting for "deactivated" (0x31)(EDT)
Message (8)	Response to Message (7) If Message (8) is received within the T4 period, Message (9) shall be transmitted immediately. There is no requirement for the processing in a case where Message (8) is not received within the T4 period.
Message (9)	NetID write request * SEA value setting (home EA's NetID → 0x00) * Setting for intra-subnet simultaneous broadcast (0x01FF)(DEA) * Setting for node profile object (0x0EF001)(DEOJ) * Setting for NetID property (0xE1)(EPC)

	* Setting for write request (0x60)(ESV) * NetID setting (EDT)
T3	Timeout period for waiting to receive a response from a router or NetID server in the subnet (60 sec. - Design Guidelines).
T4	Timeout period for waiting to receive a response from a router or NetID server located outside the subnet (60 sec. - Design Guidelines).
T5	Period to wait for a response from a node in the subnet (60 sec. - Design Guidelines).
T7	Period to wait for a response from a node located outside the subnet (T5 < T7).

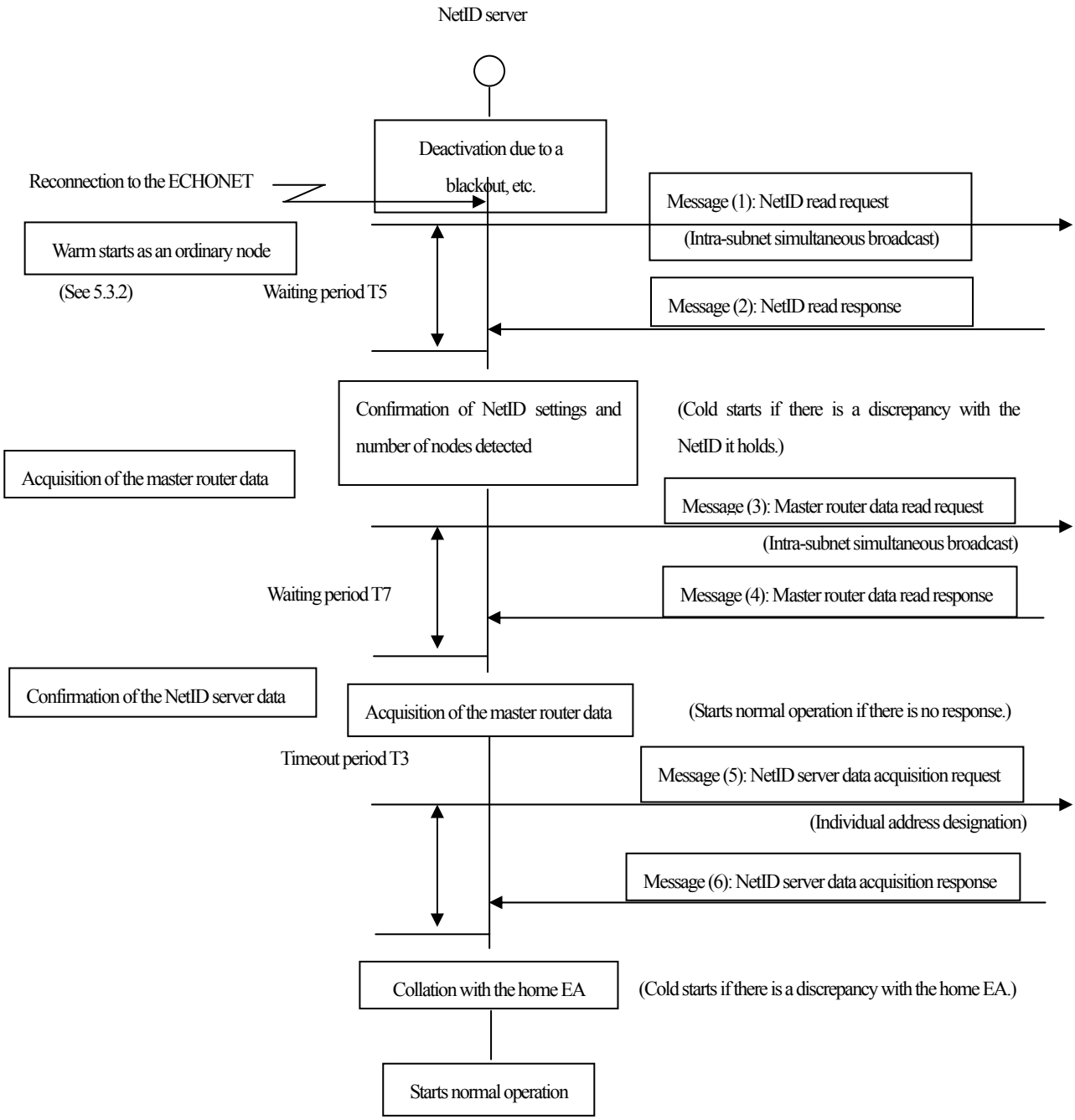


Fig. A6-2 Basic Sequence for NetID Server Warm Start

Table A6-2 Summary of Message Contents Used in the Basic Sequence for NetID Server Warm Start

Message (1)	NetID read request * SEA value setting (home EA's NetID -> 0x00) * Setting for intra-subnet simultaneous broadcast (0x01FF)(DEA) * Setting for node profile object (0x0EF001)(DEOJ) * Setting for NetID property (0xE1)(EPC) * Setting for read request (0x62)(ESV)
Message (2)	Response to Message (1) Waits for the reception of Message (2) for the T5 period. If Message (2) is not received within the T5 period or the NetID acquired from Message (2) is different from the home NetID, a shift shall be made to the cold start sequence.
Message (3)	Master router data read request * Setting of the home EA value as seen from the subnet (SEA) * Setting for intra-subnet simultaneous broadcast (0x01FF)(DEA) * Setting for router profile object (0x0EF101)(DEOJ) * Setting for master router data property (0xE6)(EPC) * Setting for read request (0x62)(ESV)
Message (4)	Response to Message (3) Waits for the reception of Message (4) for the T7 period. If Message (4) is not received within the T7 period, it shall be presumed that no master router exists in the subnet and Message (5) shall not be transmitted.
Message (5)	NetID server data acquisition request * Setting of the home EA value as seen from the subnet (SEA) * Setting of the master router's EA value acquired from Message (4) (DEA). * Setting for router profile object (0x0EF101)(DEOJ) * Setting for NetID server data (0xE3)(EPC) * Setting for read request (0x62)(ESV)
Message (6)	Response to Message (5) If Message (6) is received within the T3 period, a shift to the EA confirmation sequence shall be made immediately. If no response to Message (5) is received within the T3 period, it shall be presumed that the attempt to acquire the NetID server data has failed and a shift shall be made to the cold start sequence. If the existing NetID server's EA acquired from Message (6) is different from the home EA, a shift shall be made to the cold start sequence.
T3	Timeout period for waiting to receive a response from a router or NetID server in the subnet (60 sec. - Design Guidelines).
T5	Period to wait for a response from a node in the subnet (60 sec. - Design Guidelines).
T7	Period to wait for a response from a node located outside the subnet (T5 < T7).

Appendix 7 Processing to Be Performed upon Receipt of a Message Containing an Error

When a received ECHONET message has one of the errors listed in the table below, the respective processing specified in the table shall be performed.

Error name		Definition	Processing
DEOJ error		The DEOJ code specified in the received ECHONET message does not match the EOJ code of the ECHONET object implemented in the home ECHONET node. Or the DEOJ instance code specified in the received ECHONET message is 0x00 and the code does not match the combination of the EOJ class group code and class code of the ECHONET object implemented in the ECHONET node.	In all cases: Discard the received message.
EPC error		The received ECHONET message does not have a DEOJ error, but the EPC specified in the message does not match the EPC of the object implemented in the home ECHONET node.	If ESV = 0x60 to 0x63: Return a "response not possible" message (without EDT). If ESV = 0x64 to 0x6E: Return a "response not possible" message (*). If ESV = 0x74 or 0x78: Return a "response (processing request accepted)" message. If ESV and CpESV are other than b7:b6=0:1: Discard the received message. If b7 of the EPC is 0: Discard the received message.
ESV error	Non-array type	The received ECHONET message does not have a DEOJ or EPC error and the EPC specified in the message matches the EPC of the object implemented in the home ECHONET node, but an ESV value that does not conform to the access rule is specified and the type is the non-array type.	
	Array type	The received ECHONET message does not have a DEOJ or EPC error and the EPC specified in the message matches the EPC of the object implemented in the home ECHONET node, but an ESV value that does not conform to the access rule is specified and the type is the array type.	
Element number error	Array type	The received ECHONET message does not have a DEOJ, EPC or ESV error, the EPC specified in the message matches the EPC of the object implemented in the home ECHONET node and the ESV value conforms to the access rule, but the array element number does not match (when the type is the array type).	If ESV = 0x64 to 0x6B or 0x6E: Return a "response not possible" message (with element number). If ESV = 0x6D: Return a "response not possible" message (no EDT). If ESV = 0x6C, 0x74 or 0x78: Return a "response (processing request accepted)" message.
EDT size error		The received ECHONET message does not have a DEOJ, EPC, ESV or element number error, but the EDT size calculated from the EBC of the message is different from the actual EDT size. Or the size of the EDT of the received ECHONET message is different from the EDT size expected under the ECHONET Specification. The term "the EDT size expected under the ECHONET Specification" shall mean the size of each property specified in Appendices and the ECHONET frame specified in 4.2.8 of Part 2.	If ESV = 0x60 or 0x61: Discard the received message or return a "response not possible" message (no EDT). If ESV = 0x64 or 0x6D: Discard the received message or return a "response not possible" message (*). If ESV = 0x6E: Discard the received message.

		V = 0x74 or 0x78: Discard the received message or return a "response (processing request accepted)" message (if the EDT size is 0). Return a "response (processing request accepted)" message (if the EDT size is not 0).

(*) Whether or not the "response not possible" message contains the array element number shall be implementation-dependent.