Part II

ECHONET Lite

Communication Middleware Specifications

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.

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

The original language of The ECHONET Lite 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 Lite 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 to 4.

The ECHONET Lite Communication Middleware (hereinafter, simply referred to as "Communication Middleware") specifications were designed primarily to enable concealment of differences in Lower-Layer Transmission Medium from the perspective of the application layer.

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.

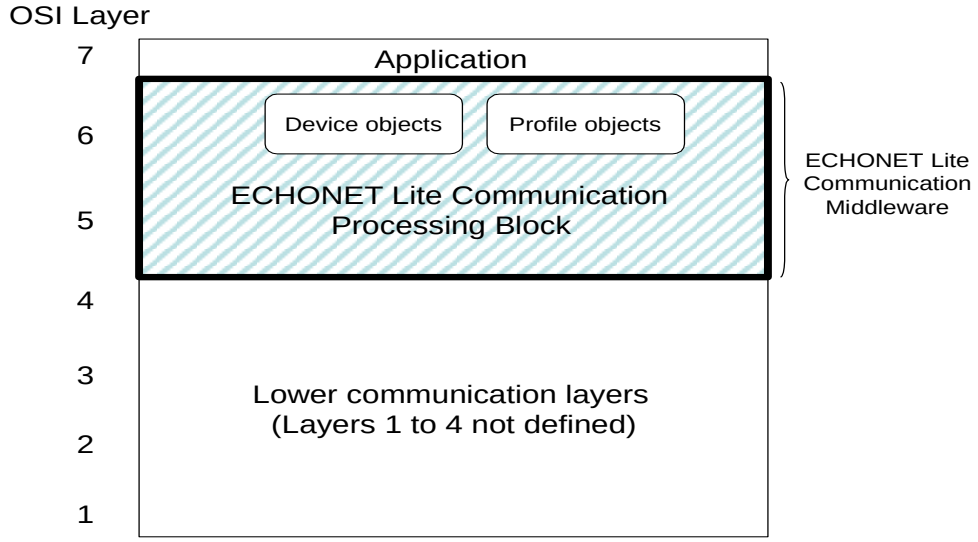


Fig. 1.1 Communication middleware

As Fig. 1.1 shows, the Communication Middleware Block specified in this document (Part 2) consists of ECHONET Lite Communication Processing Block. The ECHONET Lite Communication Processing Block is specified as a function not dependent on Layers 1 to 4. The ECHONET Lite Communications Processing Block transmits and receives the ECHONET Lite frames specified in Chapter 3. There are two types of transmission mode, namely individual transmission and broadcast transmission. With individual transmission, destinations inside the ECHONET Lite subnet are stipulated using an address in Layer 4 or lower, and ECHONET Lite frames are transmitted to specific ECHONET Lite nodes. With broadcast transmission, destinations

inside the ECHONET Lite subnet are stipulated using an address in Layer 4 or lower, and ECHONET Lite frames are transmitted to all ECHONET Lite nodes inside the subnet. If a lower communication layer (Layer 4 or lower) does not support multicast or broadcast, broadcast transmission by ECHONET Lite may be achieved in unicast by transmitting to ECHONET Lite devices connected inside the subnet. However, neither unicast destinations nor the method of setting them is specified in this Specification; these shall be determined individually for each lower communication layer used.

Security is not specified in the ECHONET Lite Communication Processing Block. By applying the existing security standard technologies to Layer 4 or lower as required, security of ECHONET Lite is ensured. See “ECHONET Lite System Design Guidelines” for details.
When using the following protocol in Layer 4 or lower, it is mandatory to support specified addresses and ports.

(1) Using UDP(User Datagram Protocol) in Layer 4 and Internet Protocol (IP) in Layer 3

Each ECHONET Lite node has its own IP address. The IP address range and acquisition method are not specified. One ECHONET Lite frame is transferred by a single UDP packet. The destination port number of a UDP packet is always 3610, irrespective of the type, such as request, response, or notification. The source port number is not specified. For broadcast (simultaneous transmission), ECHONET Lite frames are mapped on IP multicast packets and transferred.

For IPv4, the destination multicast address value shall be 224.0.23.0. IP broadcasts may be used in combination when broadcasting node discovery messages to discover ECHONET Lite nodes in the network. However, broadcasts shall always be used in combination with multicasts. For IPv6, ff02::1 (all-node multicast address) shall be used. But in both cases of IPv4 and IPv6, if OSI reference model Layer 4 or lower specifications conform to those stipulated by another standard-specification body, the multicast address stipulated by that standard-specification body shall be used.

An ECHONET Lite node waits for UDP unicast and multicast packets at port 3610. It is recommended to wait for UDP broadcast packets at port 3610 to receive and process messages.

If security is necessary in Layer 4 (UDP) and Layer 3 (IP), RFC5191 shall be used for node authentication, DTLS for encryption and tampering prevention in Layer 4 (UDP), and IPSec, etc. for encryption and tampering prevention in Layer 3 (IP).

(2) Using Transmission Control Protocol (TCP) in Layer 4 and Internet Protocol (IP) in Layer 3

Each ECHONET Lite node has its own IP address. The IP address range and acquisition method are not specified. When establishing a connection, the destination port number of a TCP packet shall always be 3610. After the establishment of a connection, no destination port number is specified. The source port number is not specified either. A response message to a request message shall be sent through the same connection.

For a general broadcast (simultaneous send), an ECHONET Lite frame is mapped to an IP multicast packet by using UDP in Layer 4. The destination multicast address is 224.0.23.0 for IPv4. IP broadcasts may be used in combination when broadcasting node discovery messages to discover ECHONET Lite nodes in the network. However, broadcasts shall always be used in combination with multicasts. For IPv6, ff02::1 (all-node multicast address) shall be used. But in both cases of IPv4 and IPv6, if OSI reference model Layer 4 or lower specifications conform to those stipulated by another standard-specification body, the multicast address stipulated by that standard-specification body shall be used.

An ECHONET Lite node supporting TCP must wait for UDP unicast and UDP multicast packets

always at port number 3610 to receive and process them. It is recommended to wait for UDP broadcast packets at port 3610 to receive and process messages.

1. 3 References

The specific command contents (device types, specific codes, etc.) of JEM-1439, which specifies the home network (especially home equipment) standard, issued in August 1988 by the Japan Electrical Manufacturers' Association (JEMA) were used for specific device object type and code specifications.

"JEM 1439 Housekeeping Command Code Assignment for Use in Home Bus System"

Source:
The Japan Electrical Manufacturers' Association (JEMA)
General Affairs Division
Tel: +81-3-3581-4841

Chapter 2 ECHONET Objects

2. 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 Lite communication whenever possible without concern for detailed specifications. The ECHONET Objects are processed in the ECHONET Lite Communications Processing Block. Control content exchanged in communications can be classified into those relating to functions unique to each device and those relating to data profiling something other than the functions unique to each device. In ECHONET Lite, all of these are specified as objects, and control and data exchange were achieved to enable their manipulation. The ECHONET Lite Specification stipulates two types of ECHONET Objects:

(1) Device Objects
(2) Profile 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).
- ECHONET Objects defined in the ECHONET Lite Specification comply with the ECHONET Specification. Of the properties of each object defined in "APPENDIX, Detailed Requirements for ECHONET Device objects", however, properties using array elements service are not specified in ECHONET Lite Specifications.

2. 2 Device Objects

"Device mechanical functions" of a device are specified as a Device Object. A Device Object aims to facilitate controls and status verifications through communications between devices. Device Object data resides in the ECHONET Lite Communication Middleware, but the device mechanical functions themselves reside in the Application Software Block. The ECHONET Lite 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, Detailed Requirements for

ECHONET Device objects".)

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. Fig. 2.1 illustrates this relationship by specific examples.

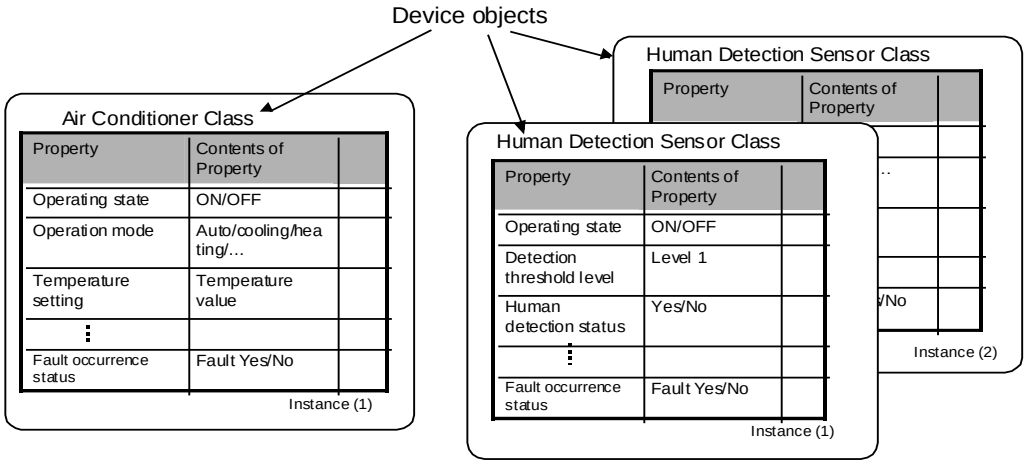


Fig. 2.1 Device object example

Class definitions for the Device Objects (Air Conditioner, etc.) (i.e., property configurations and other specific definitions and code specifications) are listed in "APPENDIX, Detailed Requirements for ECHONET Device objects". Other ECHONET Lite nodes seeking to control the functions and confirm the status of an ECHONET Lite node via ECHONET Lite 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 "APPENDIX, Detailed Requirements for ECHONET Device objects" 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 Lite communication-based writing operation.

2. 3 Profile Objects

ECHONET Lite Node Profile data, such as ECHONET Lite node operating status, 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 the ECHONET Lite Profile Class of Node Profile Objects, with detailed specifications to be provided individually. Similar to the Device Objects shown in Fig. 2.1 on the preceding section, Profile Objects define the properties to be used in each class and the services corresponding to the content and properties thereof (see "APPENDIX, Detailed Requirements for ECHONET Device objects"). Operations on the various profiles of an ECHONET Lite node are performed by manipulating (reading/writing) these profile objects.

2. 4 ECHONET Objects as Viewed from Application Software

Control from application software 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

(1) ECHONET Objects when obtaining other node status

The ECHONET Lite Specification provides two methods for obtaining the status of another node. These methods are shown in Fig. 2.2 and Fig. 2.3. In the method shown in Fig. 2.2, 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 Lite Communication Middleware for the node (Node A in the figure) making the request. In the second method, shown in Fig. 2.3, even when no request is received from an application, the ECHONET Lite 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 Lite Communication Middleware. In the former method (Fig. 2.2), because the access is performed from an application, a virtual copy of the ECHONET Objects in the other node exists in the ECHONET Lite 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. 2.4 within the ECHONET Lite Communication Middleware.

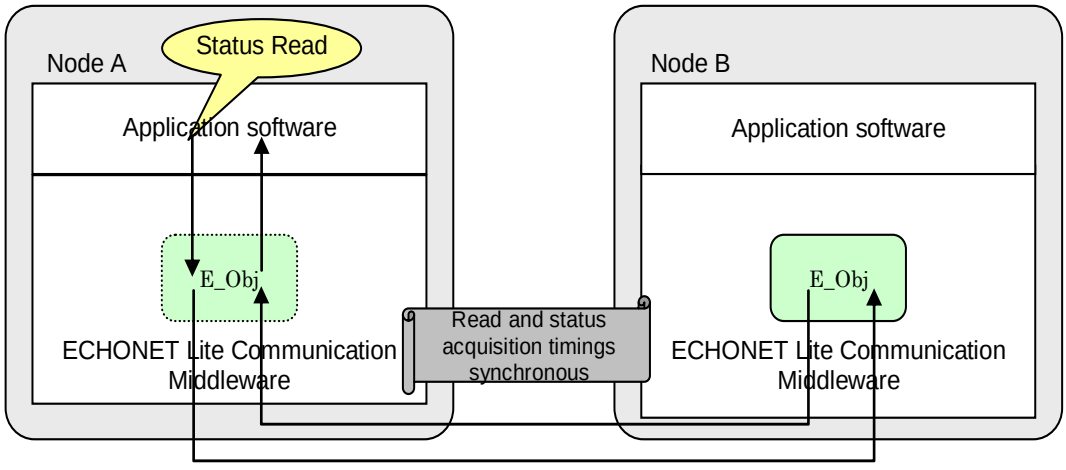


Fig. 2.2 Acquisition of other node status (1)

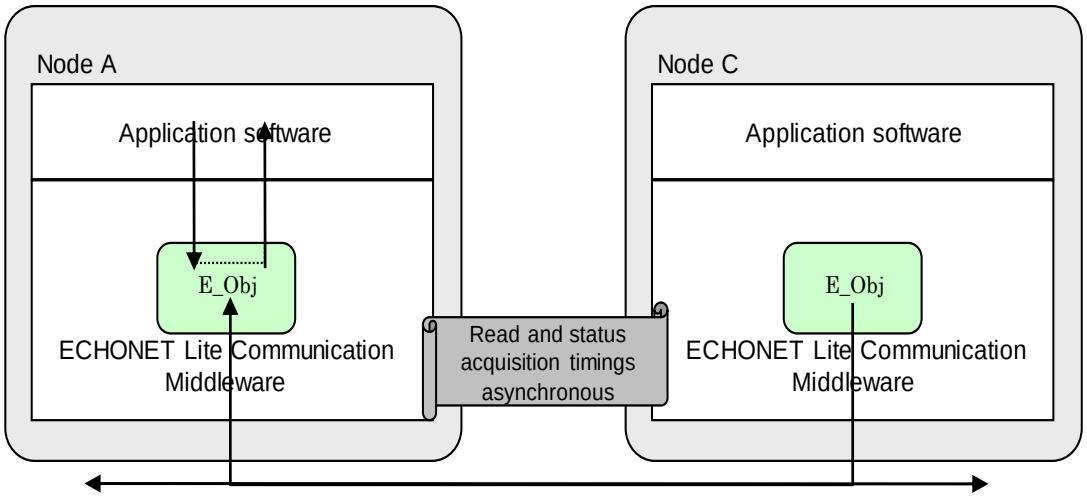


Fig. 2.3 Acquisition of other node status (2)

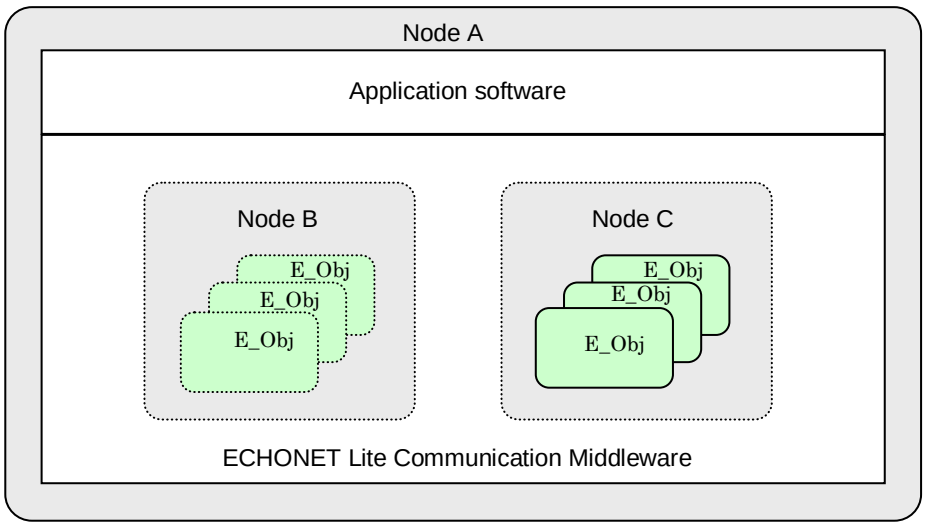


Fig. 2.4 Objects seen from application software

(2) ECHONET Objects when controlling other node functions

ECHONET Lite provides a method for controlling the functions of other nodes, as shown in Fig. 2.5. Just as in Fig. 2.2, 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 Lite Communication Middleware for the node (Node A) making the request. From the viewpoint of the application, ECHONET Objects are seen in the relationship shown by Node B in Fig. 2.6 within the ECHONET Lite Communication Middleware.

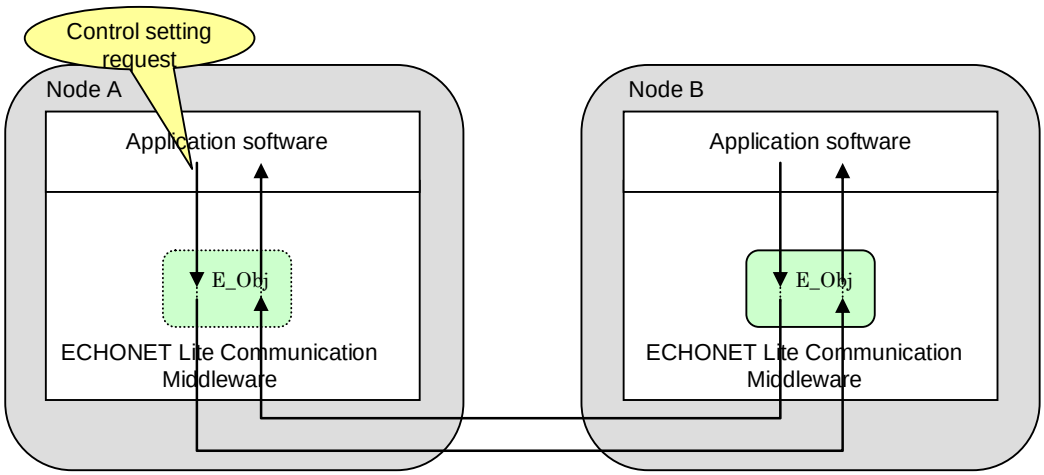


Fig. 2.5 Method of controlling other nodes

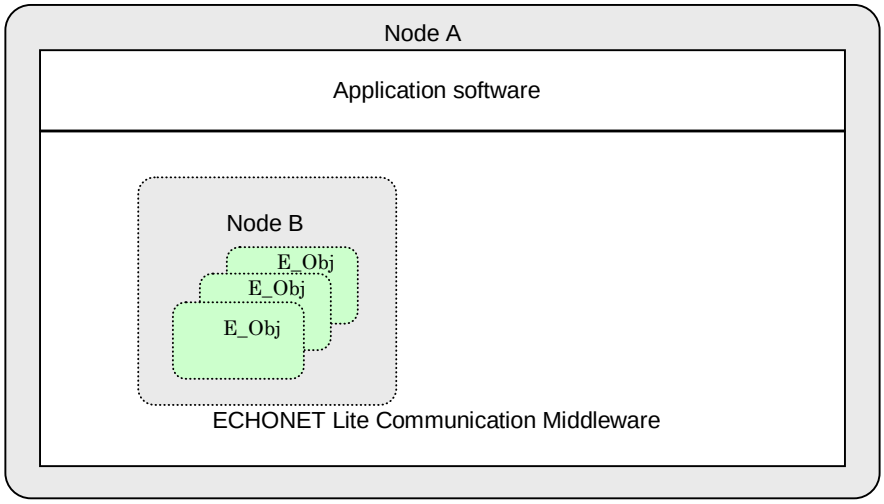


Fig. 2.6 Objects seen from application software

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

ECHONET Lite provides two methods for notifying application software on another node of the status of the self-node. These methods are shown in Fig. 2.7 and Fig. 2.8. In the method shown in Fig. 2.7, 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 Lite Communication Middleware for the node (Node A) announcing the status. In the second method, shown in Fig. 2.8, upon receiving a request from an application, the ECHONET Lite 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 Lite Communication Middleware. In the former method (Fig. 2.7), however, because communication is stipulated by the application, a virtual copy of the ECHONET Objects exists in the ECHONET Lite 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 Lite Communication Middleware (Fig. 2.9)

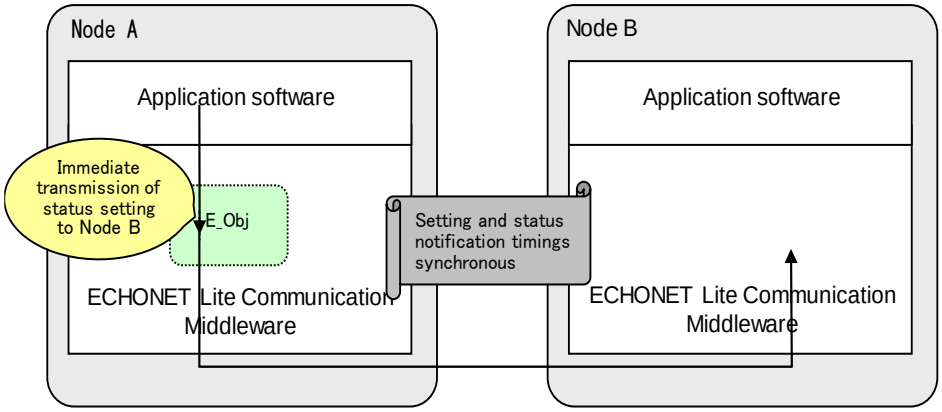


Fig. 2.7 Method of notification to other nodes (1)

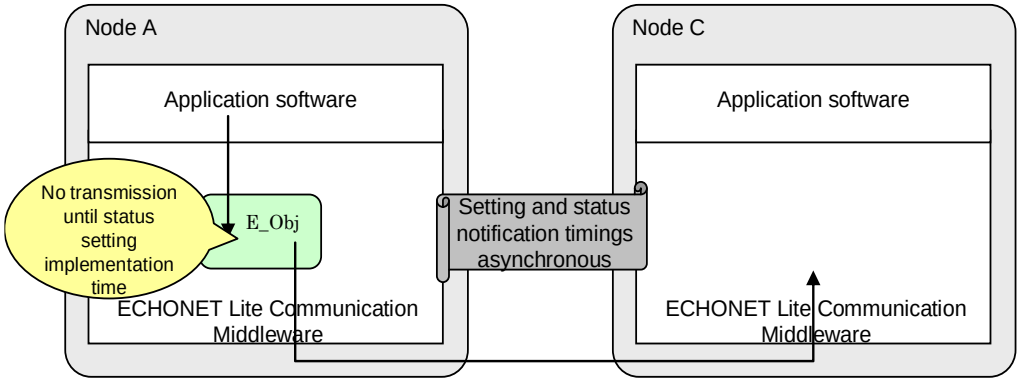


Fig. 2.8 Method of notification to other nodes (2)

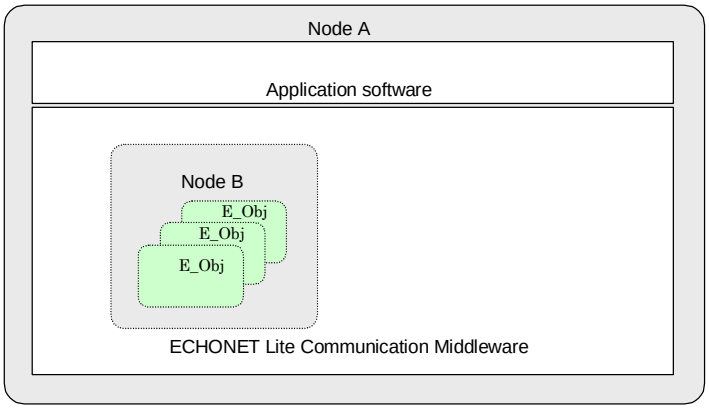


Fig. 2.9 Objects seen from application software

As is clear from the three cases shown above, the ECHONET Lite 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 Lite Communication Middleware, but there may be as many other devices as there are other related nodes.

Based on the above, Fig. 2.10 shows a typical ECHONET Lite 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.

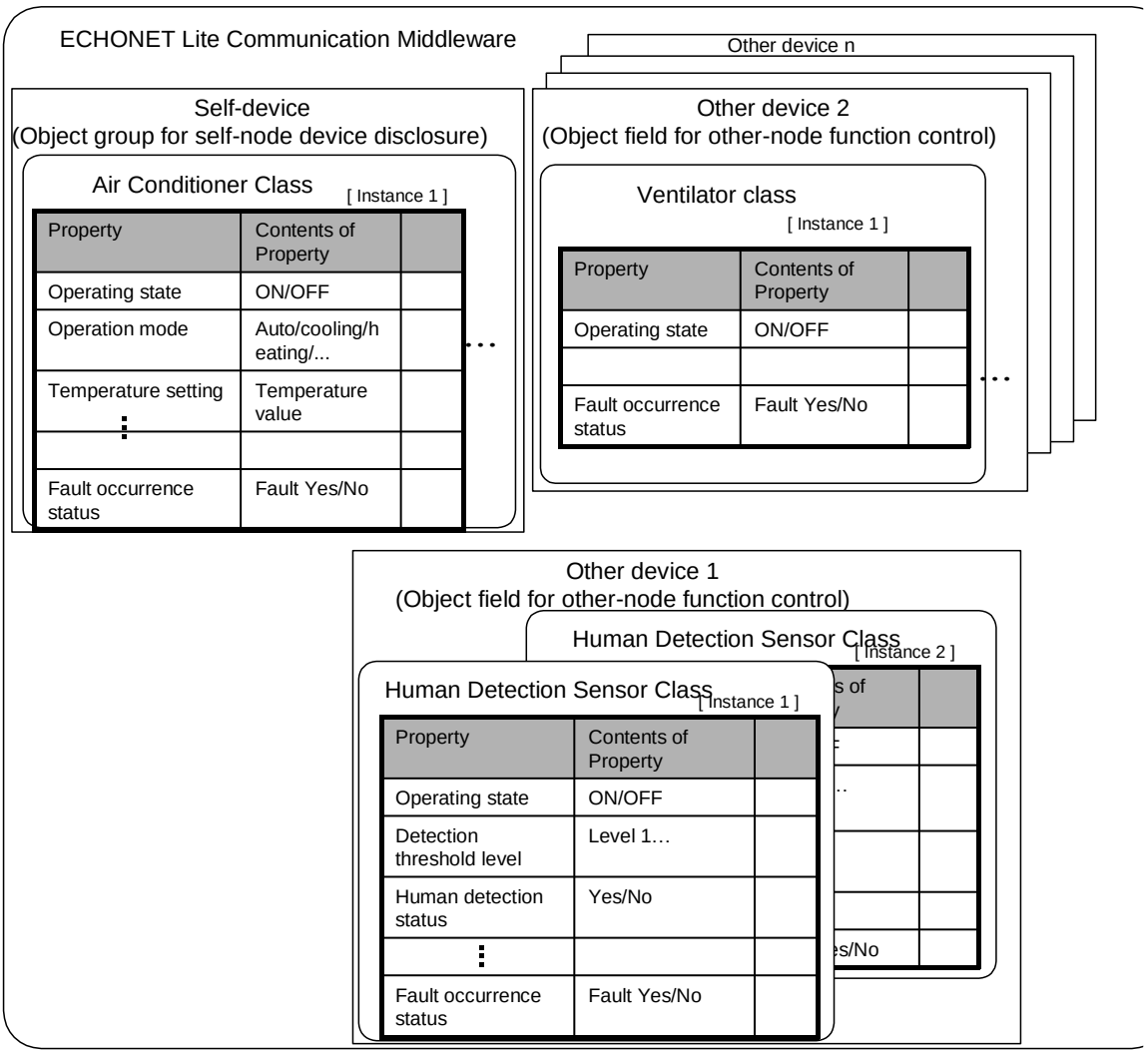


Fig. 2.10 Example of Object Configuration

Chapter 3 Message Structure (Frame Format)

3. 1 Basic Concept

To reduce the mounting load on simple devices, ECHONET Lite specifies the frame format for the ECHONET Lite Communication Middleware Block to minimize message size while fulfilling the requirements of the communications layer structure.

3. 2 Frame Format

Fig. 3.1 shows the format of ECHONET Lite frames processed by the ECHONET Lite Communication Middleware. Detailed specifications for each message component are provided on the following pages.

In this Specification, messages exchanged between ECHONET Lite Communication Processing Blocks are called ECHONET Lite frames. ECHONET Lite frames are roughly divided into two types depending on the specified EHD (see3．2．1): the message format specified by ECHONET Lite and the message format unique to the user. The ECHONET Lite frame length depends on the lower-layer communication media.

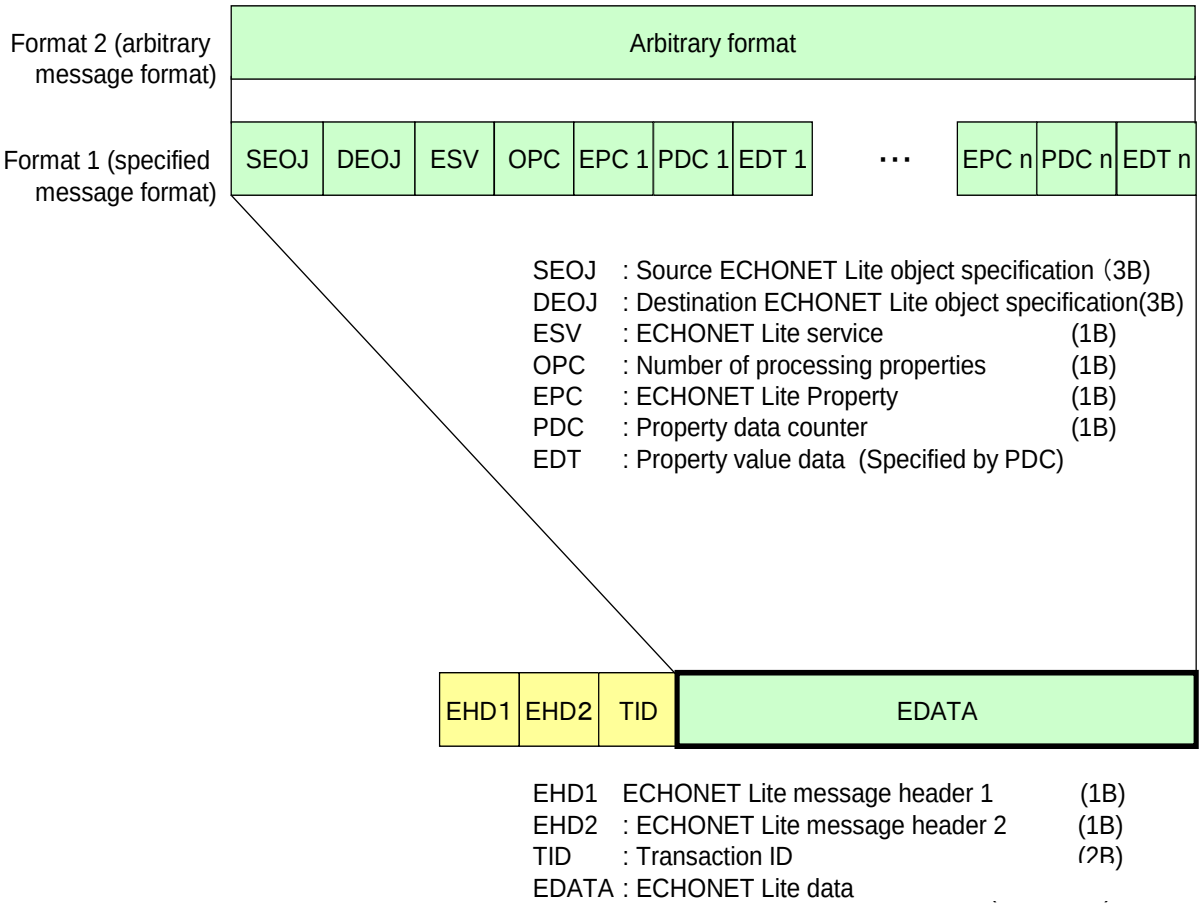


Fig. 3.1 ECHONET Lite frame format

3. 2. 1 ECHONET Lite Header (EHD)

EHD consists of ECHONET Lite Header 1 and ECHONET Lite Header 2.

3.2.1.1 ECHONET Lite Header 1 (EHD1)

The figure below shows the detailed specifications of ECHONET Lite Header 1 (EHD1) shown in Fig. 3.1.

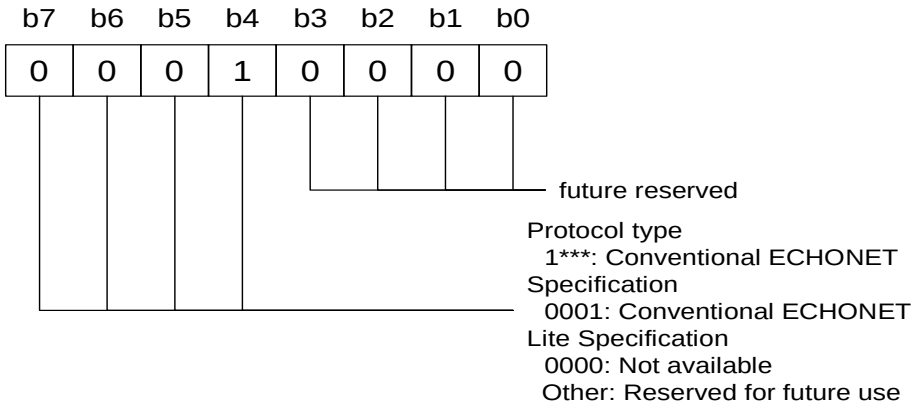


Fig. 3.2 Detailed specifications of EHD1

The combination of b7 to b4 specifies an ECHONET protocol type. b7:b6:b5:b4=0:0:0:1 indicates the ECHONET Lite Protocol defined in this Specification. b7:b6:b5:b4=0:0:0:0 shall not be used because it enables coexistence with the conventional ECHONET Protocol.

3.2.1.2 ECHONET Lite Header 2 (EHD2)

The figure below shows the detailed specifications of ECHONET Lite Header 2 (EHD2) shown in Fig. 3.1.

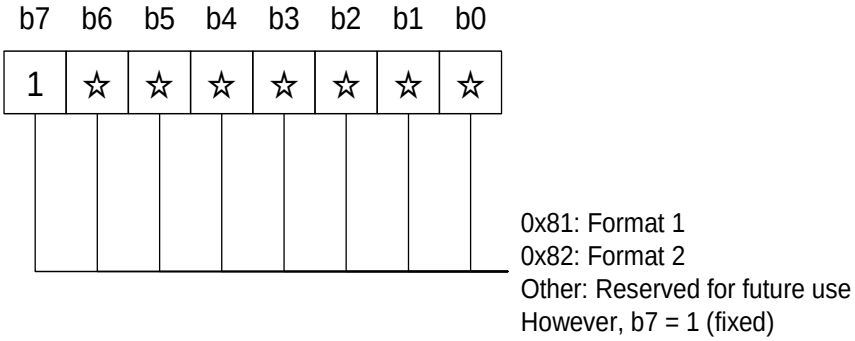


Fig. 3.3 Detailed specifications of EHD2

EHD2 defines the EDATA frame format. When EHD2 is 0x81, the EDATA frame format is Format 1 (specified message format) defined in this Specification. When EHD2 is 0x82, the EDATA frame format is Format 2 (arbitrary message format). For coexistence with the conventional ECHONET Protocol, b7 is fixed at 1.

3. 2. 2 Transaction ID (TID)

TID is a parameter used to string a sent request and a received response when a request sender receives a response in ECHONET Lite communications. A response sender shall store the same value as that contained in the request message. The TID values of property value notifications and other messages that do not need to receive a response are not expressly specified.

3. 2. 3 ECHONET Lite Data (EDATA)

EDATA refers to the data area of a message exchanged by the ECHONET Lite Communication Middleware.

3. 2. 4 ECHONET Object (EOJ)

The figure below shows the detailed specifications of ECHONET Objects in Fig. 3.1.

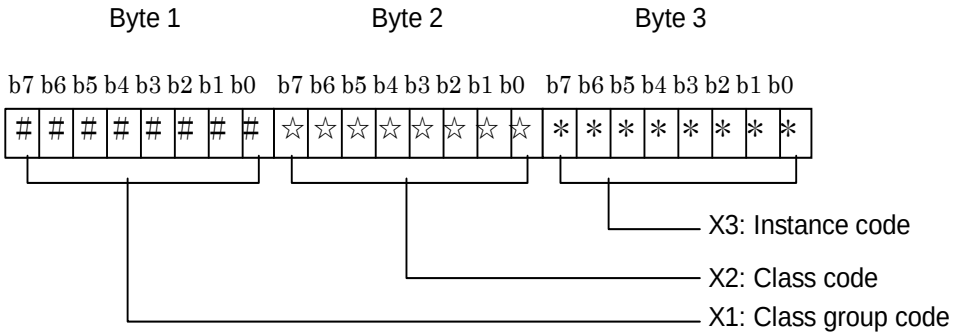


Fig. 3.4 Detailed specifications of EOJ code

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 Lite 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 Table 3.2 to Table 3.8 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 “O” in the Remarks column, with the detailed specifications to be provided in "APPENDIX, Detailed Requirements for ECHONET Device objects".

Instance code 0x00 is taken as the code for specifying all instances. This indicates that all instances in a specified class are specified.

- X1 : Class group code
  0x00–0xFF. For details, refer to Table 3.1.
- X2 : Class code
  0x00-0xFF. For details, refer to Table 3.2 to Table 3.8.
- X3 : Instance code
  0x00-0x7F. This is an identification code when the same class as that of attributes specified by [X1. X2] exists more than once in the same node.
  However, 0x00 is used as a designation of all instances of the same class.

Table 3.1 List of Class Group Codes

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	
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	
0x07-0x0D	Reserved for future use	
0x0E	Profile class group	
0x0F	User definition class group	
0x10-0xFF	Reserved for future use	

Table 3.2 Class Code List (Class Group Code X1=0x00)

For details, refer to “APPENDIX, Detailed Requirements for ECHONET Device objects”.

Table 3.3 Class Code List (Class Group Code X1=0x01)

For details, refer to “APPENDIX, Detailed Requirements for ECHONET Device objects”.

Table 3.4 Class Code List (Class Group Code X1=0x02)

For details, refer to “APPENDIX, Detailed Requirements for ECHONET Device objects”.

Table 3.5 Class Code List (Class Group Code X1=0x03)

For details, refer to “APPENDIX, Detailed Requirements for ECHONET Device objects”.

Table 3.6 Class Code List (Class Group Code X1=0x04)

For details, refer to “APPENDIX, Detailed Requirements for ECHONET Device objects”.

Table 3.7 List of Class Codes for Class Group Code (X1=0x05)

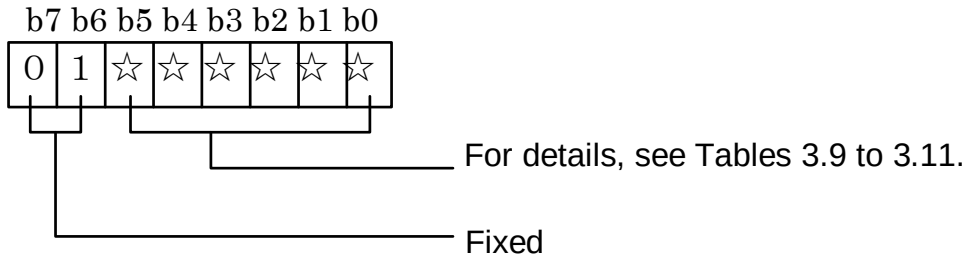
CLASS CODE	CLASS NAME	DETAILED SPECS.	REMARKS
0x00-0xFC	Reserved for future use		
0xFD	Switch		
0xFE	Portable terminal		
0xFF	Controller		

Table 3.8 List of Class Codes for Class Group Code (X1=0x0E)

CLASS CODE	CLASS NAME	DETAILED SPECS.	REMARKS
0x00-0xEF	Reserved for future use		
0xF0	Node profile	●	Detailed specifications for this class are given in Part 2, Paragraph 6.11.1.
0xF1-0xFF	Reserved for future use		

3. 2. 5 ECHONET Lite Service (ESV)

This section provides detailed specifications for the ECHONET Lite service (ESV) code shown in Fig. 3.1.



Note: Except when b7:b6=0:1, b0 to b5 have different meanings.

Fig. 3.5 ESV code detailed specifications

The service provided by this code specifies an operation for properties stipulated by the EPC. However, it does not stipulate the order of operations. The order of property operations depends on the actual implementation.

The following three types of operations are provided. The response is subdivided into two types: “response” and “response not possible”. The “response” is used when the service request in relation to all the EPC-stipulated properties is accepted. The “response not possible” 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” (response/response not possible), and “Notification”

The “response” is a response to a “request” that requires a response. It must be returned when an EOJ-stipulated object exists. When the service processing request related to all the EPC-stipulated properties is accepted, the “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, “response 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.

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, write & read, and notification (notification/notification with response required). The six operations shown below are set:

(1) Property value write (no response required)
(2) Property value write (response required)
(3) Property value read

(4) Property value write & read
(5) Property value notification
(6) Property value notification (response required)

Table 3.9 to 3.11 show specific ESV code assignments based on the content described above.

Table 3.9 List of Service Codes for Request

Service Code (ESV)	ECHONET Lite Service Content	Symbol	Remarks
0x60	Property value write request (no response required)	SetI	Broadcast possible
0x61	Property value write request (response required)	SetC	
0x62	Property value read request	Get	Broadcast possible
0x63	Property value notification request	INF_REQ	Broadcast possible
0x64-0x6D	Reserved for future use		
0x6E	Property value write & read request	SetGet	Broadcast possible
0x6F	Reserved for future use		

Table 3.10 List of ESV Codes for Response/Notification

Service Code (ESV)	ECHONET Lite Service Content	Symbol	Remarks
0x71	Property value Property value write response	Set_Res	ESV=0x61 response; Individual response
0x72	Property value read response	Get_Res	ESV=0x62 response; Individual response
0x73	Property value notification	INF	*1 : Both individual notification and broadcast notification
0x74	Property value notification (response required)	INFC	Individual notification
0x75-0x79	Reserved for future use		
0x7A	Property value notification response	INFC_Res	ESV=0x74 response; Individual response
0x7B-0x7D	Reserved for future use		
0x7E	Property value write & read response	SetGet_Res	ESV=0x6E response; Individual response
0x7F	Reserved for future use		

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

Table 3.11 List of ESV Codes for "Response Not Possible"

Service Code (ESV)	ECHONET Lite Service Content	Symbol	Remarks
0x50	Property value write request "response not possible"	SetI_SNA	ESV=0x60 response not possible; Individual response
0x51	Property value write request "response not possible"	SetC_SNA	ESV=0x61 response not possible; Individual response
0x52	Property value read "response not possible"	Get_SNA	ESV=0x62 response not possible; Individual response
0x53	Property value notification "response not possible"	INF_SNA	ESV=0x63 response not possible; Individual response
0x54-0x5D	Reserved for future use		
0x5E	Property value write & read "response not possible"	SetGet_SNA	ESV=0x6E response not possible; Individual response
0x5F	Reserved for future use		

3. 2. 6 Processing Target Property Counters (OPC, OPCSet, and OPCGet)

A target property counter consists of 1 byte. If the ESV service is for writing, reading, or notifying property values, the number of properties to be written, read, or notified is held, respectively. For the write or read service by ESV, the number of properties to be written is held in OPCSet and that of properties to be read is held in OPCGet.

The minimum value of a processing target counter is 1 and its maximum value is limited by the message length by lower communication media in transmission and reception. The value of the processing target counter can be 0 only in the condition of SetGet_SNA. A node discards a received ECHONET Lite frame if the value of the processing target property counter is different from the number of subsequent requests or responses.

If, for instance, there are three requests as shown in Fig. 3.6, the processing target property counter is 0x03.

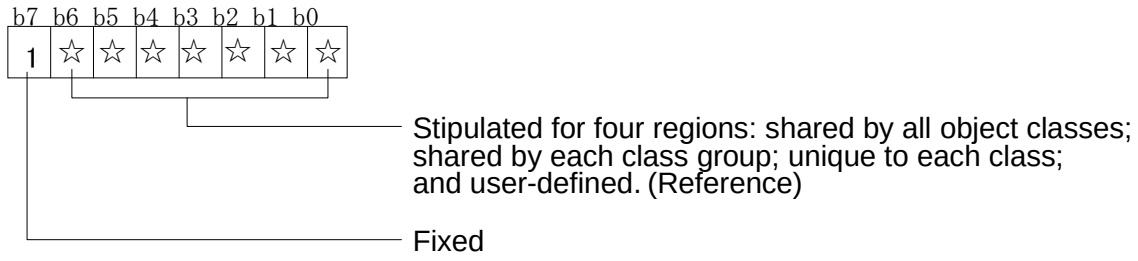
Fig. 3.6 Processing Target Property Counter for Three Requests

3. 2. 7 ECHONET Property (EPC)

This section provides detailed specifications for the ECHONET property (EPC) code shown in Fig. 3.1. 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 “APPENDIX, Detailed Requirements for ECHONET Device objects”. These codes correspond to the object property identifiers in the object definitions. However, an ECHONET Lite node will not support the array element EPC specified in “APPENDIX, Detailed Requirements for ECHONET Device objects”.

The ESV and message configuration and their relationship to EPC and ESV are described here. The EPC of an ECHONET Lite message is such that the ESV value determines whether the target object is stipulated by the SEOJ or DEOJ. When the ESV 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 ESV is a “request”, it is concluded that the EPC forms a DEOJ and that the “request” is issued from an SEOJ-stipulated object.



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

Fig. 3.7 EPC Detailed Specifications

Table 3.12 EPC Code Allocation Table

b3–b0 values (hex) \ b7–b4 values (hex) →	8	9	A	B	C	D	E	F
0								
1	Region shared by all object classes		Region shared by each class group*2		Region unique to each class*2			User-defined*1
2								
3								
4								
5								
6								
7								
8								
9								
A								
B								
C								
D								
E								
F								

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) These two regions are used in principle, but in practice the boundary line will change for each class group. Individual regions will be specified in the object class detailed specifications in Chapter 6 and "APPENDIX, Detailed Requirements for ECHONET Device objects".

Users may use the areas of 0xF0 to 0xFF in their own ways.

3. 2. 8 Property Data Counter (PDC)

The property data counter retains the number of bytes in ECHONET Property Value Data (EDT). If, for instance, the ECHONET Property Value 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 0x02, 0x01, and 0x05, respectively, as shown in Fig. 3.8. In the case of read-requests, the value of PDC is 0x00.

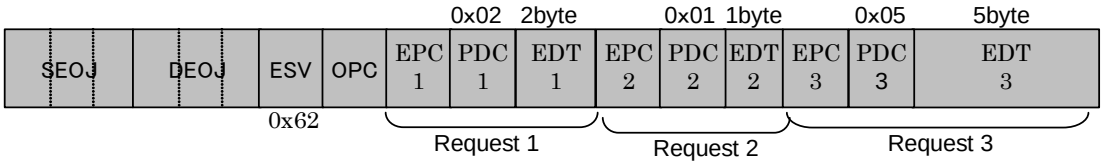


Fig. 3.8 Property Data Counter

3. 2. 9 ECHONET Property Value Data (EDT)

This section presents detailed code specifications for the ECHONET property value data (EDT) range shown in Fig. 3.1. EDT consists of data for the relevant ECHONET property (EPC), such as status notification or specific setting and control by an ECHONET Lite service (ESV). Detailed specifications are provided for the size, code value, etc. of the EDT for each EPC (see “APPENDIX, Detailed Requirements for ECHONET Device objects”).

Chapter 4 Basic Sequences

4. 1 Concept

Of the sequences exchanged between the ECHONET Lite Communication Middleware for nodes connected to the ECHONET Lite network, those that must be implemented are called “basic sequences”. This chapter divides these basic sequences into two main categories for specification:

(1) Basic sequences for object control
(2) Basic sequences for node startup

Depending on the type of device, some of the basic sequences specified in this chapter, 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 Lite Communications Processing Block's internal processing sequence that is performed at node startup is described in Section 5.4 “Startup Processing”.

4. 2 Basic Sequences for Object Control

ECHONET Lite Communication Middleware exchanges are performed by stipulating the service (ESV: ECHONET Lite 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 and detailed sequences concerning service content are described below.

(1) Basic sequences for service content
(2) Basic sequences for object control in general
(3) Detailed sequences concerning service content

4. 2. 1 Basic Sequences for Service Content

The ECHONET Lite Communication Middleware has five basic processing sequences for receiving object property-related services (specified in the table), 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 processing a notification request
(D) Basic sequence for autonomous notification
(E) Basic sequence for processing a request requiring a notification response

(A) Basic sequence for receiving a request (no response required)

There are some operations (ESV = 0x60-0x6E) that an ECHONET Lite node performs in relation to properties. The figure below shows the ECHONET Lite node's basic sequence that is performed upon receipt of ESV = 0x60:

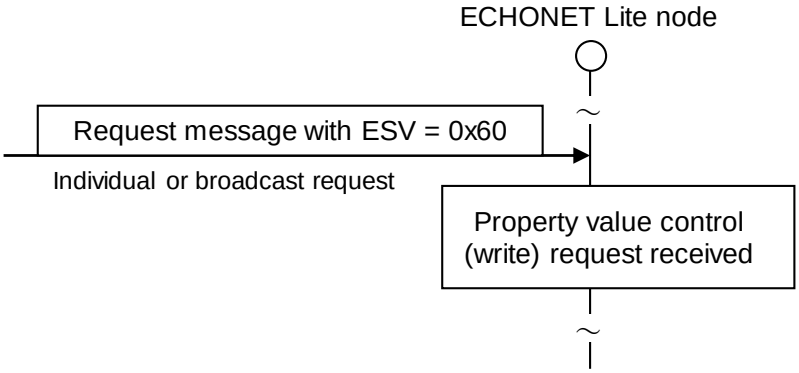


Fig. 4.1 Basic Sequence for Receiving a Request for ESV = 0x60

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

The figure below shows the basic sequence, for each ESV, for an ECHONET Lite node that has received a property value-related manipulation from another ECHONET Lite node (ESV = 0x60-0x6E), where ESV = 0x61, 0x62 or 0x6E.

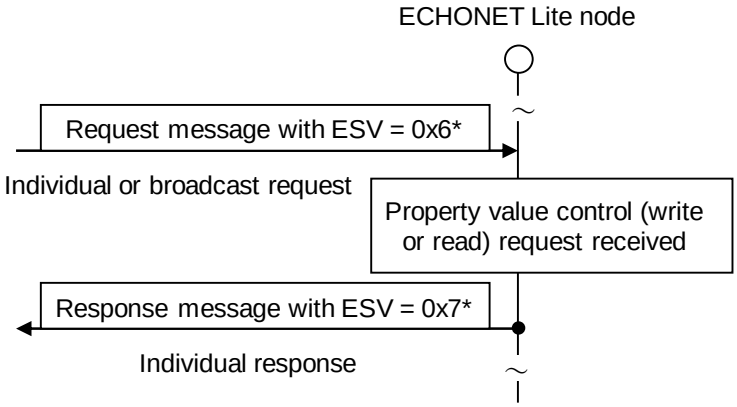


Fig. 4.2 Basic Sequence for Receiving Request for ESV = 0x6* (*: 1, 2 and E)

(C) Basic sequence for processing a notification request

The figure below shows a basic sequence that an ECHONET Lite node performs when ESV＝0x63, among operations (ESV = 0x60-0x6E) concerning property values, was received from another ECHONET Lite node.

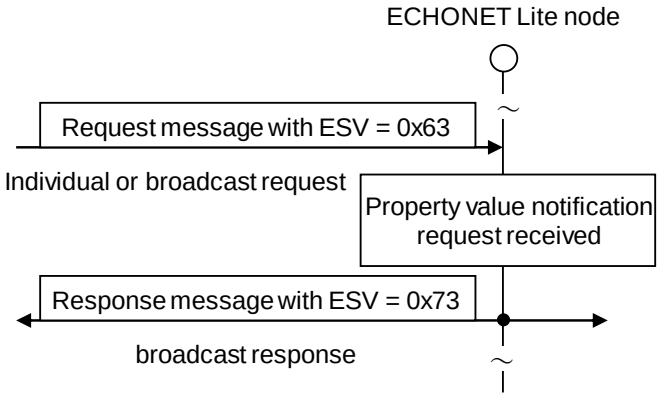


Fig. 4.3 Basic Sequence for Processing a Notification Request for ESV = 0x63

(D) Basic sequence for autonomous notification

The figure below shows a basic sequence for an autonomous notification concerning property values from the self ECHONET Lite node. For all properties, a notification message may be sent at any time based on this sequence.
Regarding a property whose status change requires notification, however, a change in the value of the property of an object must be notified by a general broadcast.

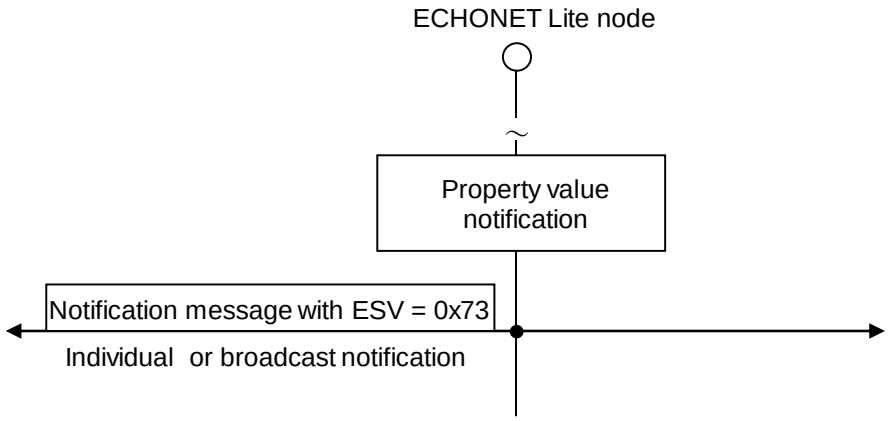


Fig. 4.4 Basic sequence for property value notification

(E) Basic sequence for processing a request requiring a notification response

The figure below shows the basic sequence for an ECHONET Lite node when a request requiring a notification response (ESV = 0x74) concerning property values was received from another ECHONET Lite node.

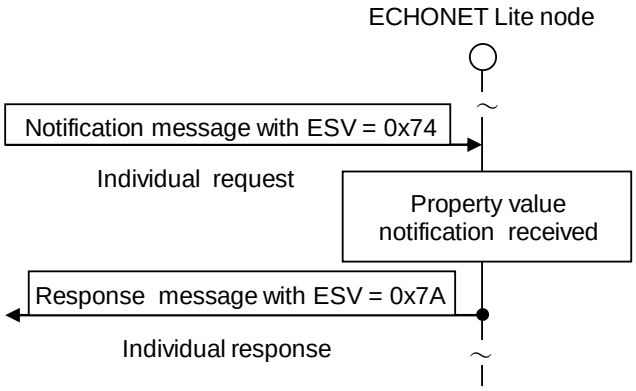


Fig. 4.5 Basic Sequence for Processing a Request Requiring a Notification Response (ESV = 0x74)

4. 2. 2 Basic Sequences for Object Control in General

The ECHONET Lite Communication Middleware performs the following six processes as basic processing when it receives a service (specified in Table 3.9 to Table 3.11) for an object property. The first five processes are described here. The sixth process (F) is described in the previous section under Basic Sequences for Service Content.

(A) Processing when the controlled object does not exist
(B) Processing when the controlled object exists, except when ESV = 0x60 to 0x63, 0x6E and 0x74
(C) Processing when the controlled object exists but the controlled property does not exist or can be processed only partially
(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 but the EDT size does not match
(F) Processing when the controlled property exists, the stipulated service processing functions are available and also the EDT size matches

(A) Processing when the controlled object does not exist

The received ECHONET Lite message is discarded and no response is returned in the following cases:

(1)The DEOJ code specified in the received ECHONET Lite message does not match the EOJ code of the ECHONET object mounted on the self ECHONET Lite node.

(2)The instance code of the DEOJ code of the received ECHONET Lite message is 0x00 and does not match the combination of the EOJ class group code and class code of the ECHONET object mounted on the ECHONET Lite node.

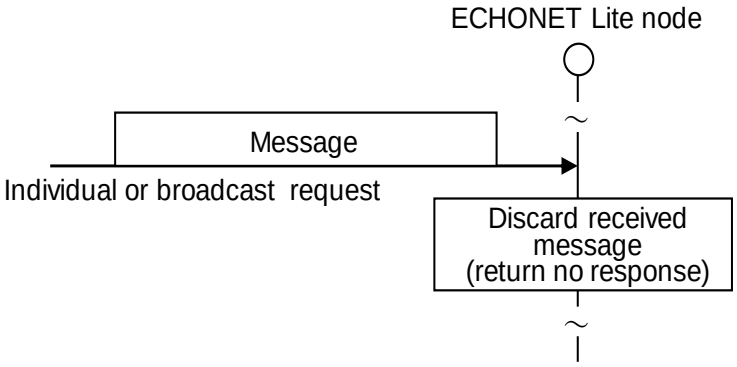


Fig. 4.6 Basic Sequence When Controlled Object Does Not Exist

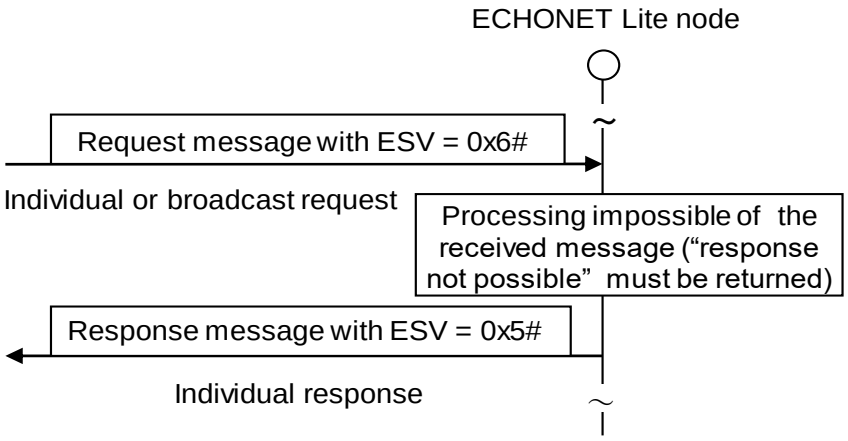
(B) Processing when the controlled object exists, except when ESV = 0x60-0x63, 0x6E and 0x74

Processing similar to (A) above.

(C) Processing when the controlled property exists but the controlled property doesn't exist or can be processed only partially

A response of processing impossible (ESV = 0x50-0x53, 0x5E) corresponding to the received ECHONET Lite message (ESV = 0x60-0x63, 0x6E) is returned in the following cases:

(1) The EPC specified in the ECHONET Lite message does not match that of the object mounted on the self ECHONET Lite node.



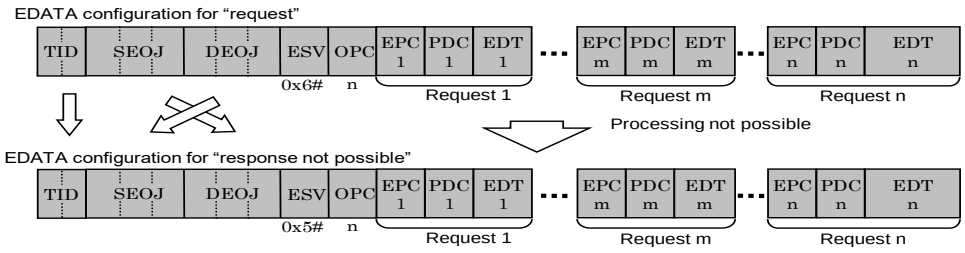


Fig. 4.7 Basic Sequence Processing performed when the object to be controlled exists but not properties to be controlled or the properties to be controlled can be processed only partially

(D) Processing when the controlled property exists but the stipulated service (ESV=0x60-0x63, 0x6E) processing functions are not available

Processing similar to (C) above

(E) Processing when the controlled property exists and the stipulated service (ESV=0x60, 0x61, 0x6E) processing functions are available but the EDT size does not match

Processing similar to (A) or (C) above.

4. 2. 3 Detailed sequences concerning service content

In diagrams 4.2.3.1 to 4.2.3.6, the EOJ 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 EOJ value indicates 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.

4.2.3.1 Property value write service (no response required) [0x60, 0x50]

In the case of a “request” (0x60), this indicates a request to write the content shown in the EDT to the property stipulated in the EPC of the DEOJ-stipulated object. When more than one property is stipulated, the writing sequence is not specified.

When the request is not accepted, or when the stipulated DEOJ exists but the stipulated EPC does not exist, “response not possible” (0x50) is returned. With a message of “response not possible”, the

value of the object stipulated by the request is set in the SEOJ, the value of the request-source object is set in the DEOJ, the same value as for the request is set in the OPC, and the same property code for the request is set in the EPC. However, for the EPC that accepted the request, 0 is set in the succeeding PDC and no EDT is attached. For the EPC that did not accept the request, the same value as for the request is set in the succeeding PDC, and the requested EDT is attached to indicate that the request could not be accepted. When the stipulated DEOJ exists but there are too many target properties of control requests to process them all, the number of properties processed from the beginning (following a judgment on whether the request is accepted or not) is set in the OPC and “response not possible” (0x50) is returned as a response. 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. Then the destination address of the lower communication layer shall be the source of "request" (the source address of the "request" message in the lower communication layer).

When the relevant object itself does not exist, neither “response” nor “response not possible” is returned.

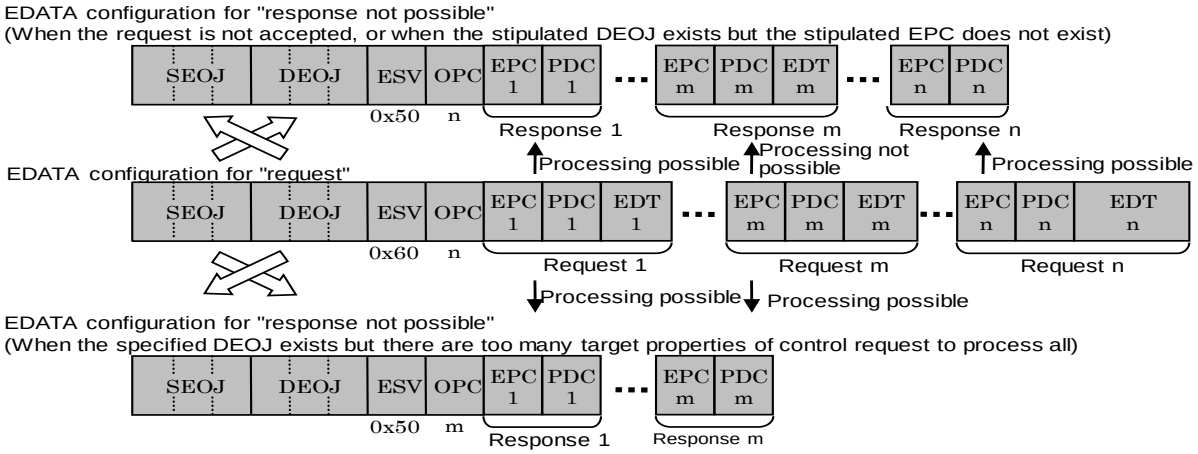


Fig. 4.8 EDATA configuration for property value write service (no response required)

4.2.3.2 Property value write service (response required) [0x61,0x71,0x51]

In the case of “request” (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. When more than one property is stipulated, the writing sequence is not specified.

In response to this “request”, when the request is to be (or has already been) accepted, a “response” (0x71) is returned. However, this “response” is not a processing implementation response but an acceptance response. In the frame format for response, the value of the object stipulated by the

request is set in the SEOJ, and the same value as for the request is set in the OPC. In the EPC, the same property code for the request is set. To indicate that the request was accepted, the PDC is set to 0 and no EDT is attached.

Moreover, because a response is an acceptance response, the property value stipulated in the EPC as mentioned above should be obtained via the property value read service, to confirm whether the equipment that actually received the request message has implemented the process.

When the request is not accepted, or when the stipulated DEOJ exists but the stipulated EPC does not exist, “response not possible” (0x51) is returned. In the same way as for a message of "response", the request-stipulated object value is set in the SEOJ, the request-source object value in the DEOJ, the same value as for the request in the OPC, and the same property code for the request in the EPC for a message of "response not possible". For the EPC that accepted the request, 0 is set in the succeeding PDC and no EDT is attached. For the EPC that did not accept the request, the same value as for the request is set in the succeeding PDC and the requested EDT is attached to indicate that the request could not be accepted.

When the stipulated DEOJ exists but there are too many target properties of control requests to process them all, the number of properties processed from the beginning (following a judgment on whether the request is accepted or not) is set in the OPC and “response not possible” (0x51) is returned as a response. The value settings for PDC and EDT shall be the same as in normal cases of response not possible. 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.

Whether a response is possible or not, the destination address of the lower communication layer shall be the source of "request" (the source address of the "request" message in the lower communication layer). When the relevant object itself does not exist, neither “response” nor “response not possible” is returned.

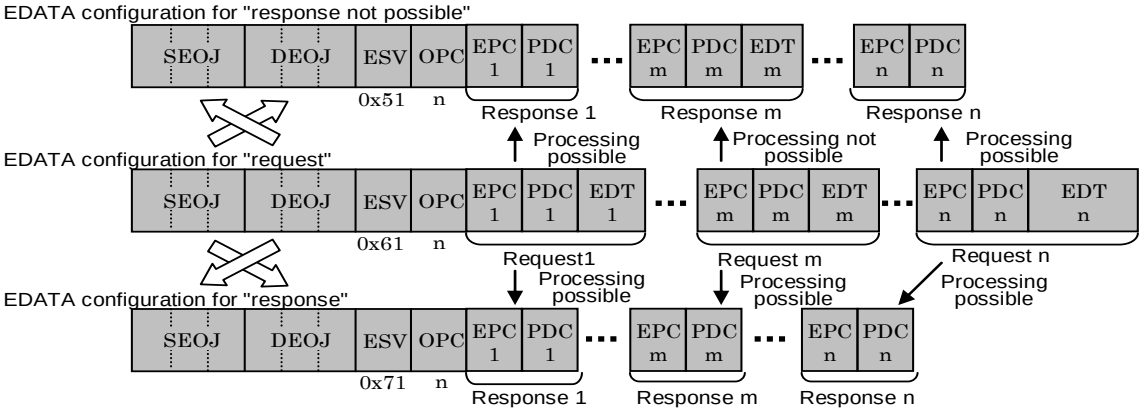


Fig. 4.9 EDATA configuration for property value write service (response required)

4.2.3.3 Property value read service [0x62,0x72,0x52]

In the case of “read” (0x62), this indicates a request to read EPC-stipulated properties from the DEOJ-stipulated object. When more than one property is stipulated, the reading sequence is not

specified. For messages in the case of a request, the PDC is set to 0.

When the request is to be (or has already been) accepted for all properties, a “response” (0x72) is returned. In the frame format for response, the value of the object stipulated by the request is set in the SEOJ, and the value of the request-source object in the DEOJ. In the OPC, the same value as for the request is set. To indicate that the request was accepted, the length of the read property is set in the PDC and the read property value in the EDT.

When the request is not accepted, or when the stipulated DEOJ exists but the stipulated EPC does not exist, “response not possible” (0x52) is returned. In the same way as for a message of "response", the request-stipulated object value is set in the SEOJ, the request-source object value in the DEOJ, the same value as for the request in the OPC, and the same property code for the request in the EPC for a message of "response not possible". For the EPC that accepted the request, the length of the read property is set in the succeeding PDC and the read property value in the EDT. For the EPC that did not accept the request, 0 is set in the succeeding PDC and no EDT is attached to indicate that the request was not accepted.

When the stipulated DEOJ exists but there are too many properties subject to control requests to process them all, or when not all of the read-requested property values can be returned because they exceed the allowable message length, the number of properties processed from the beginning (following a judgment on whether the request is accepted or not) is stored in the OPC and “response not possible” (0x52) is returned as a response. The value settings for PDC and EDT shall be the same as in normal cases of response not possible. 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.

When the relevant object itself does not exist, neither “response” nor “response not possible” is returned. Whether a response is possible or not, the destination address of the lower communication layer shall be the source of "request" (the source address of the "request" message in the lower communication layer).

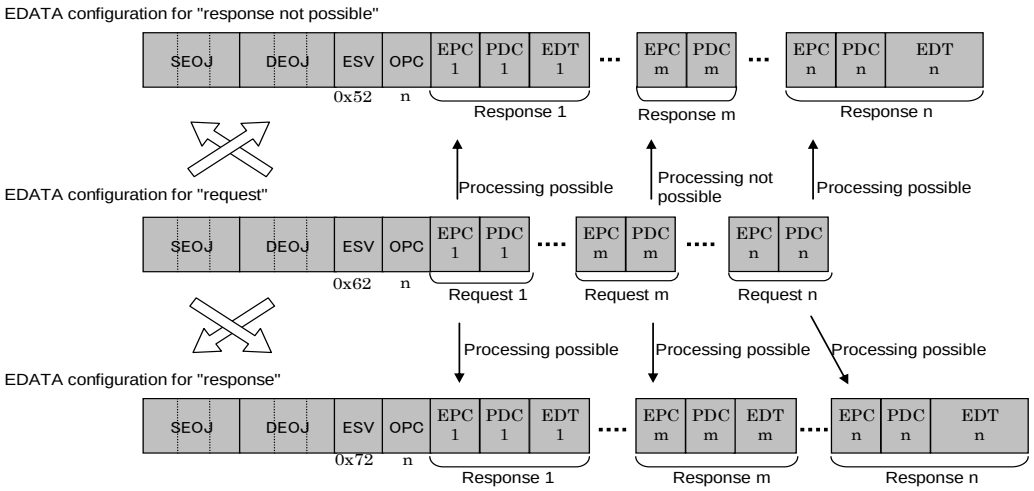


Fig. 4.10 EDATA configuration for property value read service

4.2.3.4 Property value write & read service [0x6E,0x7E,0x5E]

"Write & read" (0x6E) indicates a service to process two requests by a single message: a request for writing EDT-stipulated contents into EPC-stipulated properties of a DEOJ-stipulated object and a request for the contents of EPC-stipulated properties from a DEOJ-stipulated object. The number of write-requested properties is stored in OPCSet and that of read-requested properties is set in OPCGet. The PDC corresponding to a read-requested EPC is set to 0. The sequence of processing write-requests and read-requests is not specified. That is, whether the value before or after the completion of write-request processing is stored as a response to a property stipulated in a read-request depends on the actual implementation. Similarly, if more than one property is stipulated in both write-request and read-request, the sequence of processing for each property is not specified.

Moreover, because the response is an acceptance response, the property value stipulated in the EPC as mentioned above should be obtained via the property value read service, to confirm whether the equipment that actually received the request message has implemented the process.

When the request is to be (or has already been) accepted, a “response” (0x7E) is returned. In the frame format for response, the value of the object stipulated by the request is set in the SEOJ and the request-source object value in the DEOJ. The same value as for the request is set in OPCSet, and the same property code for the request is set in the EPCs set. The PDC is set to 0 and no EDT is attached. The OPCGet for the request is set in OPCGet, the same property code for the request in the EPC, the length of the read property in the PDC, and the read property value in the EDT.

When the request is not accepted, or when the stipulated DEOJ exists but the stipulated EPC does not exist, “response not possible” (0x5E) is returned. When the specified DEOJ exists but there are too many target properties of control requests to process them all, or all the property values requested for write or read cannot be returned because the allowable message length is too short, the number of properties processed from the beginning is stored in OPCSet and OPCGet. Then "response not possible" (0x5E) is returned as a response. 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.

When the relevant object itself does not exist, neither “response” nor “response not possible” is returned. Whether a response is possible or not, the destination address of the lower communication layer shall be the source of "request" (the source address of the "request" message in the lower communication layer).

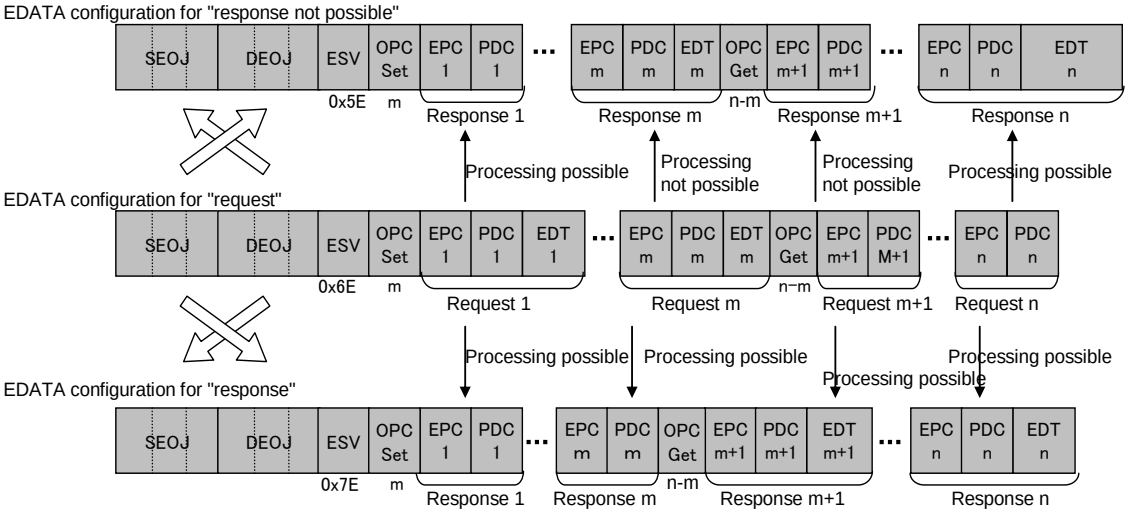


Fig. 4.11 EDATA configuration for property value write & read service

This service is an option. If a node not supporting this option receives a request for the service, the message will be discarded if the stipulated DEOJ is not incorporated. If the stipulated DEOJ is incorporated, 0 will be stored in OPCSet and 0 in OPCGet, and “response not possible” (0x5E) will be returned as a response.

4.2.3.5 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 these 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 broadcast simultaneously; hereafter “announce” will signify a broadcast) the content of the property stipulated in the EPC of the DEOJ-stipulated object. For messages in the case of a request, the PDC is set to 0. When more than one property is stipulated, the notification sequence is not specified.

In response to this “notify request”, when the request is to be accepted, a “response” (0x73) value is notified by broadcast transmission. The request-stipulated object value is set in the SEOJ, the request-source object value in the DEOJ, and the same value as for the request in the OPC. The same property code as for the request is set in the EPC and the property length of notification is set in the PDC. In the EDT, the requested property value (contents of notification) is stored. For broadcast, destination addresses in lower communication layers are set.

When the request is not accepted, or when the stipulated DEOJ exists but the stipulated EPC does not exist, “response not possible” (0x53) is returned to the request source by individual transmission. In the same way as for a message of "response", the request-stipulated object value is set in the SEOJ, the request-source object value in the DEOJ, the same value as for the request in the OPC, and the same property code for the request in the EPC for a message of "response not possible". For the EPC that accepted the request, the length of the read property is set in the succeeding PDC and the read property value in the EDT. For the EPC that did not accept the request, 0 is set in the

succeeding PDC and no EDT is attached to indicate that the request was not accepted. When the specified DEOJ exists but there are too many target properties of control request to process them all, or the property value (contents of notification) requested for read cannot be returned because the allowable message length is too short, the number of properties processed from the beginning is stored in the OPC, the same property code for the request in the EPC, the length of the read property in the PDC, and the read property value in the EDT. Then "response not possible" (0x53) is returned as a response. In this case, the responding side can determine the number of property values to be returned. Also for a response not possible, the address of the lower communication layer of the request source shall be set as the destination address of the lower communication layer. When the relevant object itself does not exist, neither “response” nor “response not possible” is returned. In the case of an autonomous “notification”, the DEA is set to a broadcast for a required status change notification. In other cases, however, the destination of the lower communication layer can be set arbitrarily for broadcast or individual transmission.

For an autonomous "notification", a node profile class is stored if there is no DEOJ, explicitly specified EOJ in particular.

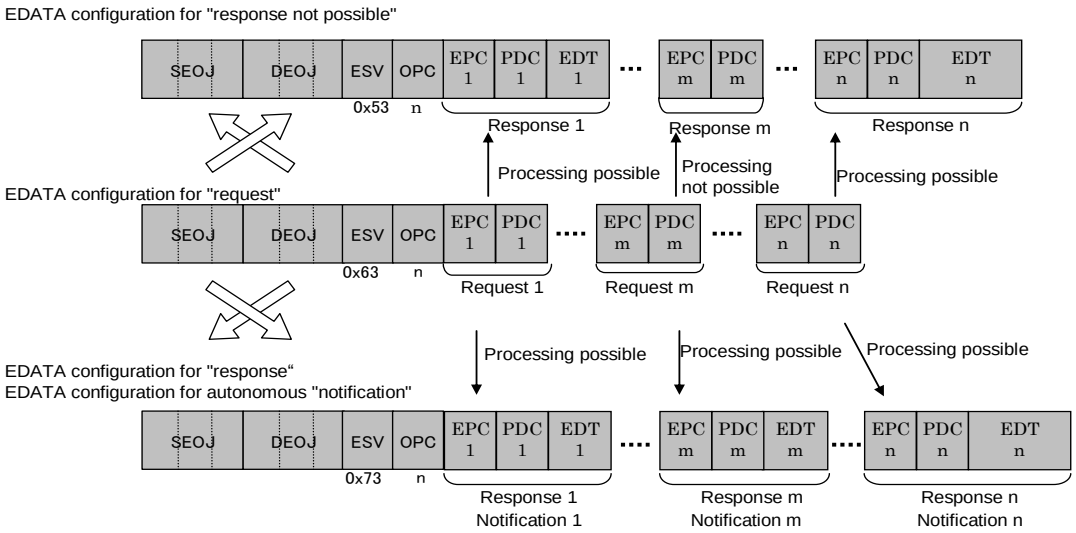


Fig. 4.12 EDATA configuration for property value notification service

4.2.3.6 Property value notification service (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. When more than one property is stipulated, the notification sequence is not specified.

When the specified DEOJ exists, a “response” (0x7A) for autonomous notification reception is returned. In a response message, the requested object value is set in the SEOJ and the request-source object value in the DEOJ. The same value as for notification is set in the OPC and the same property code as for notification is set in the EPC. To indicate that the notification was received, the PDC is set to 0 and no EDT is attached.

When the specified DEOJ does not exist, the message shall be discarded.

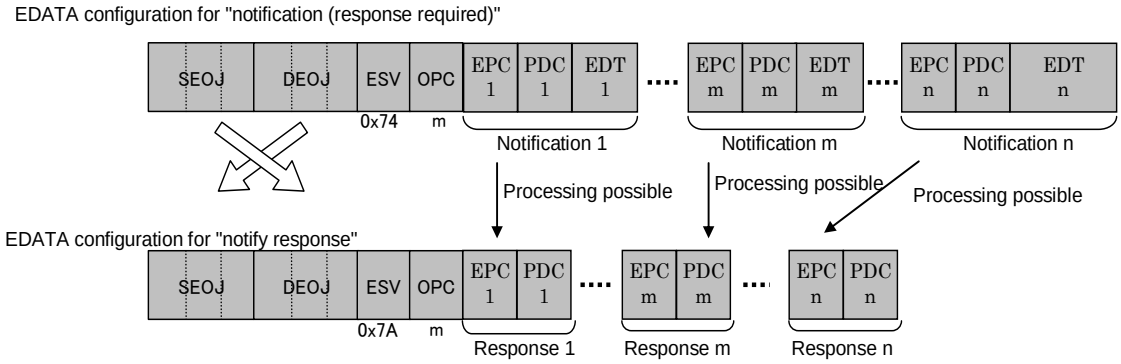


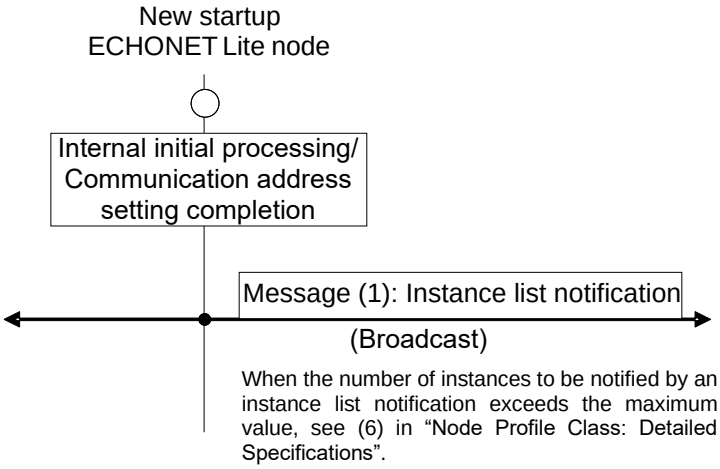
Fig. 4.13 EDATA configuration for property value notification (response required) service

4. 3 Basic Sequence for ECHONET Lite Node Startup

For the ECHONET Lite nodes described in this section, startup begins with the acquisition of a communication address for self-recognition and specification. This section specifies the startup sequences, assuming that the communication address has already been obtained when the ECHONET Lite Communication Middleware begins operation.

4. 3. 1 Basic Sequence for ECHONET Lite Node Startup

The figure below shows the basic sequence that an ECHONET Lite node performs at startup.
This processing is also executed when a communication address is changed. It is preferable to send an instance list notification not with another EPC but with OPC set to 1 because some nodes can receive the notification only when OPC is1.



Message (1)	・Stipulates node profile objects (0x0EF001) with SEOJ. In a transmission-only node case, however, 0x0EF002 is stipulated. ・Stipulates node profile objects (0x0EF001) with DEOJ ・Stipulates instance list notification properties (0xD5) by EPC. ・Stipulates notification (0x73) by ESV. ・Stipulates OPC=1 as a rule. ・Stipulates instance list information in EDT=self node.

Fig. 4.14 Basic Sequence for ECHONET Lite Node Startup

Chapter 5 ECHONET Lite Communications Processing Block Processing Specifications

5. 1 Basic Concept

This section presents the specifications for ECHONET Lite communications processing in the ECHONET Lite 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 Lite Communications Processing Block and are not intended as specifications for actual software structure.

(1) Object processing
(2) Send message assembly and management processing
(3) Startup processing

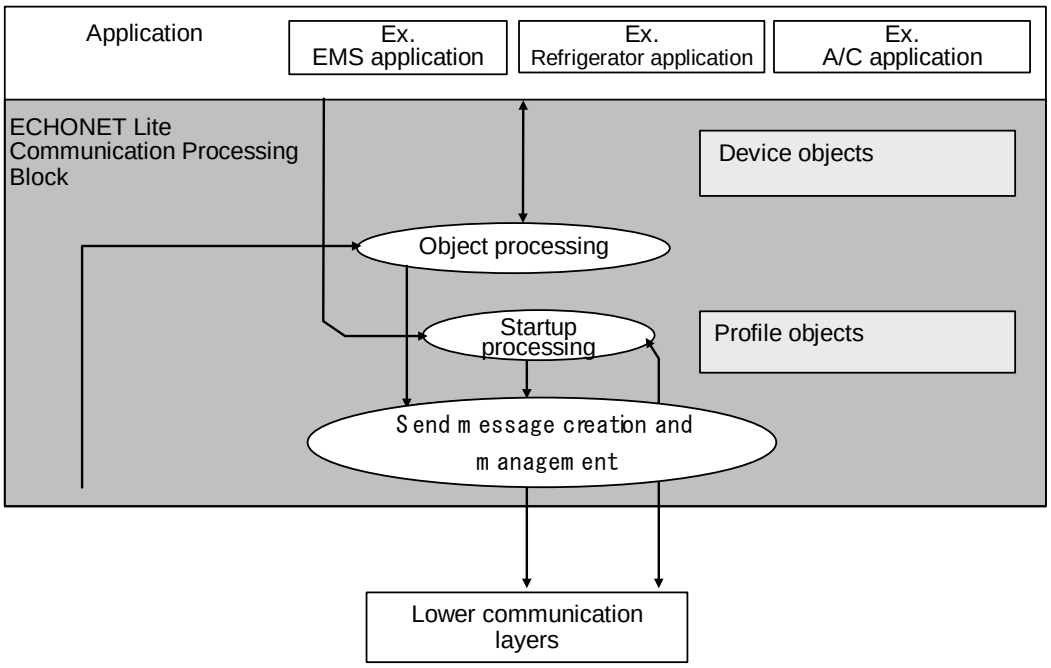


Fig. 5.1 Overview of Communication Middleware Processing (Layer Configuration)

5. 2 Object Processing Specifications

In the ECHONET Lite Communications Processing Block, device functions are expressed as objects, and through these objects operations are performed between nodes. See Chapter 2 and “APPENDIX, Detailed Requirements for ECHONET Device objects”. for detailed information on objects. This section gives the object processing specifications below.

Processing using operation data from application software 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 setting or control (read/write) data is received from application software, the block first determines which type of object the data concerns and then performs the appropriate processing. Processing specifications for the two categories are described 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 which 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 application software and the stipulated object and property exist, processing is performed in accordance with the request stipulated in application software processing.

(2) Other device object processing specifications
The data (reference/control content) is received from application software, the stipulated object and property data and the destination address data are handed off to send message creation/management processing, and processing is terminated.

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

5. 3 Send Message Creation/Management Processing

When the data necessary to create an ECHONET Lite message is received from startup processing or object processing, the data required for an ECHONET Lite message, such as ECHONET Lite header (EHD), is added to create the message and send it through the lower communication interface.

5. 4 Startup Processing

When communication address setting is completed, the startup sequence processing specified in Chapter 4 is performed, and the message data to be transmitted is handed off to send message creation/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.

Chapter 6 ECHONET Objects: Detailed Specifications

6. 1 Basic Concept

This section specifies specific values for the class codes of ECHONET objects processed in the ECHONET Lite Communication Middleware, whose types and overview were given in Chapter 2, along with property configurations and their detailed specifications. ECHONET objects described in this chapter and in “APPENDIX, Detailed Requirements for ECHONET Device objects” are divided into two main classes: device objects and profile 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 chapter 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

Detailed specifications for each device object class are provided in “APPENDIX, Detailed Requirements for ECHONET Device objects".

Each ECHONET Lite node must implement a device object and a node profile class.

6. 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 “APPENDIX, Detailed Requirements for ECHONET Device objects”.

6. 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 Lite Communication Middleware data as ECHONET property value data (EDT) beginning from the most significant byte.

6. 2. 2 ECHONET Property Value Range

The definition range for the ECHONET properties specified in this chapter and "APPENDIX, Detailed Requirements for ECHONET Device objects", 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 6.1 shows the underflow and overflow codes for each data type.

Table 6.1 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) For the handling of other ECHONET property values, see“ECHONET Lite System Design Guidelines”.

6. 2. 3 Class-specific Mandatory Properties

The properties defined as the “mandatory” properties for specific classes in the property specifications in this chapter and “APPENDIX, Detailed Requirements for ECHONET Device objects” shall be implemented as part of the respective classes.

However, transmission-only devices are exceptional. Even properties classified as "Mandatory" need not always be implemented. For the handling of transmission-only devices, see “ECHONET Lite System Design Guidelines”.

6. 2. 4 Properties that Must Have a Status Change Announcement Function

Any property may transmit a property value notification (0x73) message by an individual notification or broadcast and a property value notification (requiring a response) (0x74) by an individual notification 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, Detailed Requirements for ECHONET Device objects” requires the incorporation of a function to send a property value notification service message in the form of a broadcast upon a change in the status (property value) of that property. A node profile object (0x0EF001) is set to DEOJ. 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 a broadcast.

6. 2. 5 Access Rules

The access rules regulate a group of services that can be implemented. In this specification, the following three types are stipulated:

Set : Processing a service related to a property value write request
Processing acceptance of a property value write request (no response required) (0x60), property value write request (response required) (0x61), or property value write-read request (0x6E).

Get : Processing a service related to a property value read or notification request
Processing acceptance of a property value read request (0x62), property value write-read request (0x6E), or property value notification request (0x63).

Anno : Processing a service related to a property value notification request
Processing acceptance of a property value notification request (0x63)

6. 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 and detailed in “APPENDIX, Detailed Requirements for ECHONET Device objects”.

6. 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 device object super class property is implemented by each device object class. Specifications for the device object super class are detailed in “APPENDIX, Detailed Requirements for ECHONET Device objects".

6. 4 Sensor-related Device Class Group Objects: Detailed Specifications

Stated in “APPENDIX, Detailed Requirements for ECHONET Device objects”

6. 5 Air Conditioning-related Device Class Group Objects: Detailed Specifications

Stated in “APPENDIX, Detailed Requirements for ECHONET Device objects”

6. 6 Housing/Equipment-related Device Class Group Objects: Detailed Specifications

Stated in “APPENDIX, Detailed Requirements for ECHONET Device objects”

6. 7 Cooking/Housework-related Device Class Group Objects: Detailed Specifications

Stated in “APPENDIX, Detailed Requirements for ECHONET Device objects”

6. 8 Health-related Device Class Group Objects: Detailed Specifications

Stated in “APPENDIX, Detailed Requirements for ECHONET Device objects”

6. 9 Management/Control-related Device Class Group Objects: Detailed Specifications

Stated in “APPENDIX, Detailed Requirements for ECHONET Device objects”

6. 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.

6. 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 6.2 below.

Table 6.2 List of Profile Object Super Class Configuration Properties

Property Name	EPC	Contents of Property Value Area (Decimal notation)	Data Type	Data Size (Bye)	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	3	Get	O		(2)
Business facility code	0x8B	Stipulated in 3-byte business facility code (Specified individually by each manufacturer)	unsigned char×3	3	Get			(3)
Product code	0x8C	Specified in ASCII code (Specified individually by each manufacturer)	unsigned char×12	12	Get			(4)
Production number	0x8D	Specified in ASCII code (Specified individually by each manufacturer)	unsigned char×12	12	Get			(5)
Production date	0x8E	Stipulated in 4 bytes YYMD (1 byte/character) YY : Year (0x07CF for 1999) M : Month (0x0C for 12) D : Day (0x14 for 20)	unsigned char×4	4	Get			(6)
Status change announcement property map	0x9D	See Supplement 1 of "APPENDIX Detailed Requirements for ECHONET Device objects".	unsigned char× (MAX17)	Max. 17	Get	O		
Set property map	0x9E	See Supplement 1 of "APPENDIX Detailed Requirements for ECHONET Device objects"	unsigned char× (MAX17)	Max. 17	Get	O		
Get property map	0x9F	See Supplement 1 of "APPENDIX Detailed Requirements for ECHONET Device objects"	unsigned char× (MAX17)	Max. 17	Get	O		

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

(1) Fault status
The "Fault status" property of the device object super class indicates whether a fault has occurred in

the actual device. This property shall be set to 0x41 when there is a fault and 0x42 when there is no fault.

(2) Manufacturer code
The “Manufacturer code” property identifies the manufacturer using a 3-byte code. Each ECHONET Consortium member is assigned a unique “Manufacturer code” property value by the Consortium.

(3) Business facility code
The “Business facility code” property identifies the relevant business facility of the manufacturer using a 3-byte code. “Business facility code” property values are not defined by the ECHONET Consortium; they are defined by each manufacturer.

(4) Product code
The “Product code” property identifies the relevant product of the manufacturer using a 12-byte ASCII code. “Product code” property values are not defined by the ECHONET Consortium; they are defined by each manufacturer. When the “Product code” property value is less than 12 bytes, the product code shall be left-justified in the data area and the remainder of the data area shall be padded with NULLs or spaces.

(5) Production number
The “Production number” property indicates the production number of the relevant product of the manufacturer using a 12-byte ASCII code. “Production number” property values are not defined by the ECHONET Consortium; they are defined by each manufacturer. When the “Production number” property value is less than 12 bytes, the production number shall be left-justified in the data area and the remainder of the data area shall be padded with NULLs or spaces.

(6) Production date
The “Production date” property indicates the production date of the relevant product of the manufacturer using a 4-byte code. Two of the 4 bytes are used to indicate the year of production. The remaining 2 bytes are used to indicate the month of production and the day of production, with one byte used for each.

6. 10. 2 Property Map

Regarding each property specified in a profile object, three property maps specified for profile object super-class shall be the same as those of the device object super-class specified in “APPENDIX, Detailed Requirements for ECHONET Device objects”.

6. 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 6.3 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 6. 10 "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 “mandatory” 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).

Table 6.3 Object Class List (Profile Class Group)

Class Group Code	Class Code	Object Class Name	Mandatory
0x0E	0xF0	Node profile	O

6. 11. 1 Node Profile Class: Detailed Specifications

Class group code: 0x0E
Class code: 0xF0
Instance code: 0x01 (general node), 0x02 (transmission-only node)

As an instance code, 0x01 is used for a general node and 0x02 for a transmit-only node.

Property Name	EPC	Contents of Property Value Area (Decimal notation)	Data Type	Data Size (Bytes)	Access Rule	Mandatory	Announce Status Change	Remarks
Operating status	0x80	Indicates the node operating status. Booting = 0x30, not booting = 0x31	unsigned char	1	Set Get	 O	O	(1)
Version information	0x 82	Indicates ECHONET Lite 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 type with a bitmap.	unsigned char×4	4	Get	O		(9)
Identification number	0x83	Number to identify the node implementing the device object in the domain. 1st byte: lower communication ID field 0xFE: Set bytes 2 to 17 in the manufacturer-specified format. Other: reserved for future use	unsigned char × 17	17	Get	O		(10)
Fault content	0x89	Fault content 0x0000-0x03EC,0x03FF	unsigned short	2	Get			(2)
Unique identifier data	0xBF	Stipulated in 2 bytes See (3) below.	unsigned short	2	Set/Get			(3)
Number of self-node instances	0x D3	Total number of instances held by self-node 1st to 3rd bytes: Total number of instances	unsigned char×3	3	Get	O		(4)
Number of self-node classes	0x D4	Total number of classes held by self-node 1st and 2nd bytes: Total number of classes	unsigned char×2	2	Get	O		(5)
Instance list notification	0x D5	Instance list when self-node instance configuration is changed	unsigned char×(MA	Max. 253	Anno	O	O	(6)

		1st byte: Number of notification instances 2nd to 253rd bytes: ECHONET object codes (EOJ3 bytes) enumerated	X) 253					
Self-node instance list S	0x D6	Self-node instance list 1st byte: Total number of instances 2nd to 253rd bytes: ECHONET object codes (EOJ3 bytes) enumerated	unsigned char×(MAX)253	Max. 253	Get	O		(7)
Self-node class list S	0x D7	Self-node class list 1st byte: Total number of classes 2nd to 17th bytes: Class codes (EOJ high-order 2 bytes) enumerated	unsigned char×(MAX) 17	Max. 17	Get	O		(8)

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

(1) Operating status

Indicates whether or not the current operating status permits ECHONET Lite node communications.

(2) Fault content

The values of 0x0000 to 0x03E8 and 0x03FF will be the same as the code assignment for fault content properties for device objects.

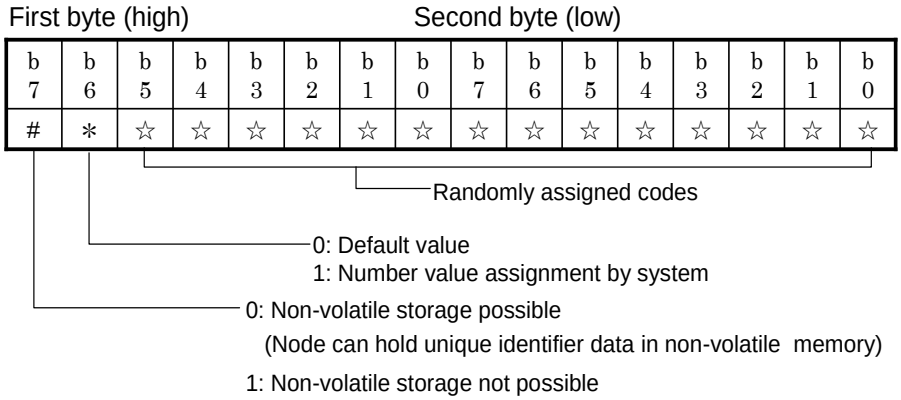
The values of 0x03E9 to 0x03EC are abnormality codes of ECHONET Lite middleware adapters described in "Part III, ECHONET Lite Communications Equipment Specifications".

(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. Unique identifier data preset on the device side shall be called a default value, and that set by another ECHONET Lite node after ECHONET Lite system participation shall be called an assigned value.

As a rule, unique identifier data must be held in 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 available, 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). It is prohibited for another ECHONET Lite node to set unique identifier information where b6 at the first byte is 0.

Code description specifications are shown below.



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) Number of self-node instances

The total number of instances across all classes of device objects disclosed by the self-node
The number of self-node instances does not include instances of node profile objects.

(5) Number of self-node classes

The total number of classes disclosed by the self-node, including node profile classes

(6) Instance list notification

A property to announce the configuration of instances to be disclosed to the network at startup. This property also announces instances held at the self-node each time the configuration of instances disclosed to the network is changed during system operation, such as instance addition or deletion. This property is for announcement only by expecting another node as a trigger for recognizing an instance change in detail. The number of instances reported by the message in question is inserted in the first byte, while instances retained by the self-node are listed in bytes 2-253 (EOJ3 bytes). However, the instance list does not include node profile objects. The maximum number of instances announced at one time is 84. If the total instance list numbers 85 or more, refer to “ECHONET Lite System Design Guidelines”. This instances of all device objects held at the self-node are targeted for announcement.

(7) Self-node instance list S

A list of the instances of device objects disclosed by the self-node. When the total number of instances is 85 or more, the total number is inserted in the first byte as the instance count, with the number of held instances in the second and subsequent byte, for transmission. The number

of instances to be inserted shall depend on the implementation. The value of the first byte is specified as follows:

0x00–0xFE : Total number of instances (when 254 or less) instruction
0xFF : Overflow (when 255 or more) instruction

To acquire all instances from a node having 85 or more instances, an instance list notification request shall be made to the node.

(8) Self-node class list S

A list of classes disclosed by the self-node, excluding the node profile. If the total number of classes is 9 or more, the total number of classes is inserted in the first byte and classes to be held are inserted in the second and subsequent bytes for transmission. The number of classes to be inserted shall depend on implementation. The first byte shall be set as follows:

0x00–0xFE : Total number of classes (when 254 or less) instruction
0xFF : Overflow (when 255 or more) instruction

To acquire more instances from a node having 9 or more classes, an instance list notification request shall be made to the node and classes to be held shall be judged.

Here is an example of property values in the node profile object of a node that has a node profile object (EOJ = 0x0EF001), two temperature sensor objects (EOJ = 0x001101, 0x001102), and one humidity sensor object (EOJ = 0x001201).

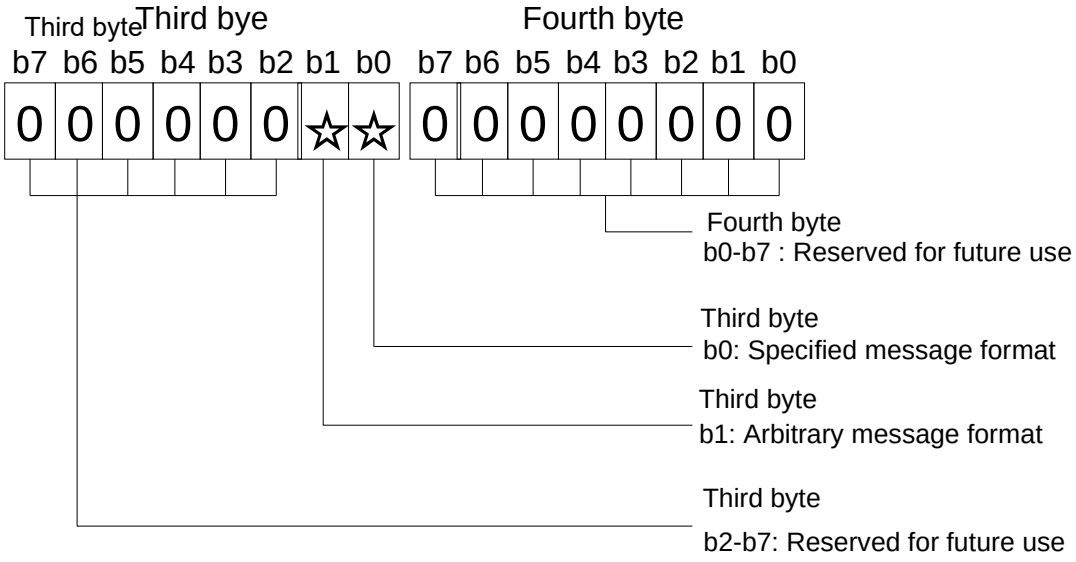
・Number of self-node instances (0xD3): 0x000003 (temperature sensor *2, humidity sensor *1)
・Number of self-node classes (0xD4): 0x0003 (node profile, temperature sensor, and humidity sensor)
・Instance list notification (0xD5): 0x03001101001102001201 (10 bytes)
・Self-node instance list S(0xD6): 0x03001101001102001201 (10 bytes)
・Self-node class list S (0xD7): 0x0200110012 (5 bytes)

(9) Version information

A 2-byte binary value shows the version number of the specification corresponding to the communication middleware, 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.



(10) Identification number

Identification number refers to a number used to identify an object uniquely within a domain. Since ECHONET Lite does not define protocol types for lower communication layers, only 0xFE is supported as protocol types for lower communication layers.

The manufacturer-specified format (0xFE) consists of a manufacturer code field to store the code of each manufacturer and a field defined by each manufacturer.

The first to third bytes indicate a 3-byte manufacturer code specified by the ECHONET Consortium.

Byte 4 and later store the unique ID of each vendor. Each vendor shall ensure that the codes will not overlap.

Manufacturer code (3 bytes)	Unique ID section (defined by the manufacturer) (13 bytes)

Appendix 1 Error Processing at Message Reception

If an error is found in an ECHONET Lite message received, process it as follows:

Error Type	Definition	Error Processing
EOJ error	A DEOJ code specified in a received ECHONET Lite message does not match the EOJ code of an ECHONET object installed in the self ECHONET Lite node. Or the instance code of the DEOJ code specified in the received ECHONET Lite message is 0x00, and, it does not match the combination of the EOJ class group code and class code of an ECHONET object installed in the ECHONET Lite node.	In all cases: Discard
EPC error	An EPC specified in a received ECHONET Lite message free of an EOJ error does not match the EPC of an object installed in the self ECHONET Lite node.	ESV=0x60 to 0x63,0x6E: Response not possible ESV=0x74: Response to process ESV is not as above: Discard
ESV error	An EPC specified in a received ECHONET Lite message free of an EOJ or EPC error matches the EPC of an object installed in the self ECHONET Lite node. However, an ESV not complying with the access rules is specified.	
EDT size error	The EDT size of a received ECHONET Lite message free of an EOJ or EPC error does not match the EDT size specified in the ECHONET Lite Specification. The EDT size assumed in the ECHONET Lite Specification is the size of each property size specified in Chapter 3 "Frame Format" and "APPENDIX, Detailed Requirements for ECHONET Device objects".	ESV=0x60,0x61,0x6E: Discard or response not possible ESV=0x74: Discard or response to process (EDT size: 0) Response to process (in other cases)