Part IX ECHONET Gateway Specifications

Revision record

• Version 1.0	March 18th	2000 released	Open to consortium members
	July	2000	Open to the public
• Version 1.01	May 23rd	2001	Open to the public Version 1.0 addendum & corrigendum Parts 1, 8, 9, and 10 were unchanged from Version 1.0.
• Version 2.00 Draft	August 7th	2001	Open to consortium members Specifications for Java-API were added to Part 4. Other parts were unchanged from Version 1.01.
• Version 2.01	December 19th	2001	Open to consortium members
• Version 2.10 Preview	December 28th	2001	Open to consortium members
• Version 2.10 Draft	February 15th	2002	Open to consortium members
• Version 2.10	March 7th	2002	Open to consortium members
• Version2.11	April 26th	2002	Open to consortium members

Contents

Chapter 1 Overview

1.1 Basic Concept

The ECHONET principal specifications provide 1) communication protocol specifications for devices connected to equipment systems and 2) intra-device interface and processing specifications for homes, small buildings, and stores. In such environments the equipment system is rarely operated independently (i.e., without connections to other systems). Instead, it is usually connected both to an external system(s) and to an AVCC system within the building. In the ECHONET standard, devices which are located between the ECHONET system and an external system and which supervise the linked action relay are called ECHONET Gateways.

ECHONET Gateway software must provide numerous functions to enable standardization with external systems, security access assurance, and prevention of excessive loads. The minimum necessary and common functions are selected, defined, and specified as the ECHONET Gateway Service Middleware to facilitate connection between ECHONET and other systems. In other words, the Service Middleware provides ECHONET Gateway functions mounted not as Service Middleware but rather as functions to facilitate the creation of ECHONET Gateway software. In the Service Middleware, functions are specified on the assumption that the method of connecting to the ECHONET system is given not only to application software on the ECHONET Gateway but also to applications of other devices on an external system(s).

Here, “external systems” include external systems, in-home AVCC systems, and so on. This specification is provided in consideration of the different Service Middleware functions required for each external system.

1.2 Concept of ECHONET Connections with External Systems

The basic concept of connections with external systems in ECHONET is presented below.

(1) Basic gateway functions are designed with commonality and defined as Service Middleware (Gateway Service Middleware). Application software for connecting with other systems must always access ECHONET Devices in the ECHONET Domain through the Gateway Service Middleware.

(2) Access to an ECHONET Device from outside the building shall be allowed only after a security check by the ECHONET Gateway.

(3) The ECHONET communication protocol does not specify a code for identification of the ECHONET Domain. To identify multiple ECHONET Domains from an external system, each application software program shall define its own identifier for each domain.

1.3 Concept of Function Definitions as Service Middleware

The devices handled and the services created by the ECHONET system are closely related to the lifestyles of the people living in the building. The potential exists for ECHONET malfunctions to disrupt a normal lifestyle. The ECHONET system is designed to handle many devices, many of which are not easily accessible to users. Thus it may not always be

possible to recover such malfunctions with simple reset processing.

Accordingly, the ECHONET Gateway must be provided with a means of preventing adverse effects to the ECHONET system when connected to an external system. Multiple interface points may exist between the ECHONET system and the external system. Gateway functions must be implemented under the same policy at these points. The functions required to implement the gateway functions under the same policy are designed for commonality and selected to enable standardization of the ECHONET system by external systems, access security assurance, and prevention of excessive loads. This leads to the functions of the ECHONET Gateway Service Middleware.

1.4 ECHONET Gateway Types

ECHONET Gateways are classified into three types based on usage. This classification is based on the information security level of the ECHONET Domain to be secured. This standard does not specify gateway specifications for each type; instead, only examples are presented for reference.

(1) External service vendor gateway type
(2) Outdoor user gateway type
(3) In-house system gateway type

(1) External service vendor gateway type

The external service vendor gateway is a gateway for a system to be used by external service vendors connected to in-home ECHONET Domains. The home dweller shall have control over information within the in-home ECHONET Domain. Public disclosure of information shall be limited to the scope specified by the dweller.

(2) Outdoor user gateway type

The outdoor user gateway is a gateway for external systems connected to in-home ECHONET Domains. However, unlike the “External service vendor gateway”, the user has control over information contained in the in-home ECHONET Domain. That is, this gateway is tele-controlled based on the assumption that outside users are accessing the system.

(3) In-house system gateway type

The in-home gateway is a gateway for an ECHONET Domain connected to another system in the same house. Here, the user of the other system is presumed to be the same as the person with control over information in the ECHONET Domain. One example might be gateways for connection with AVCC system networks.

Fig. 1.1 shows the positioning of the ECHONET Gateway Service Middleware.

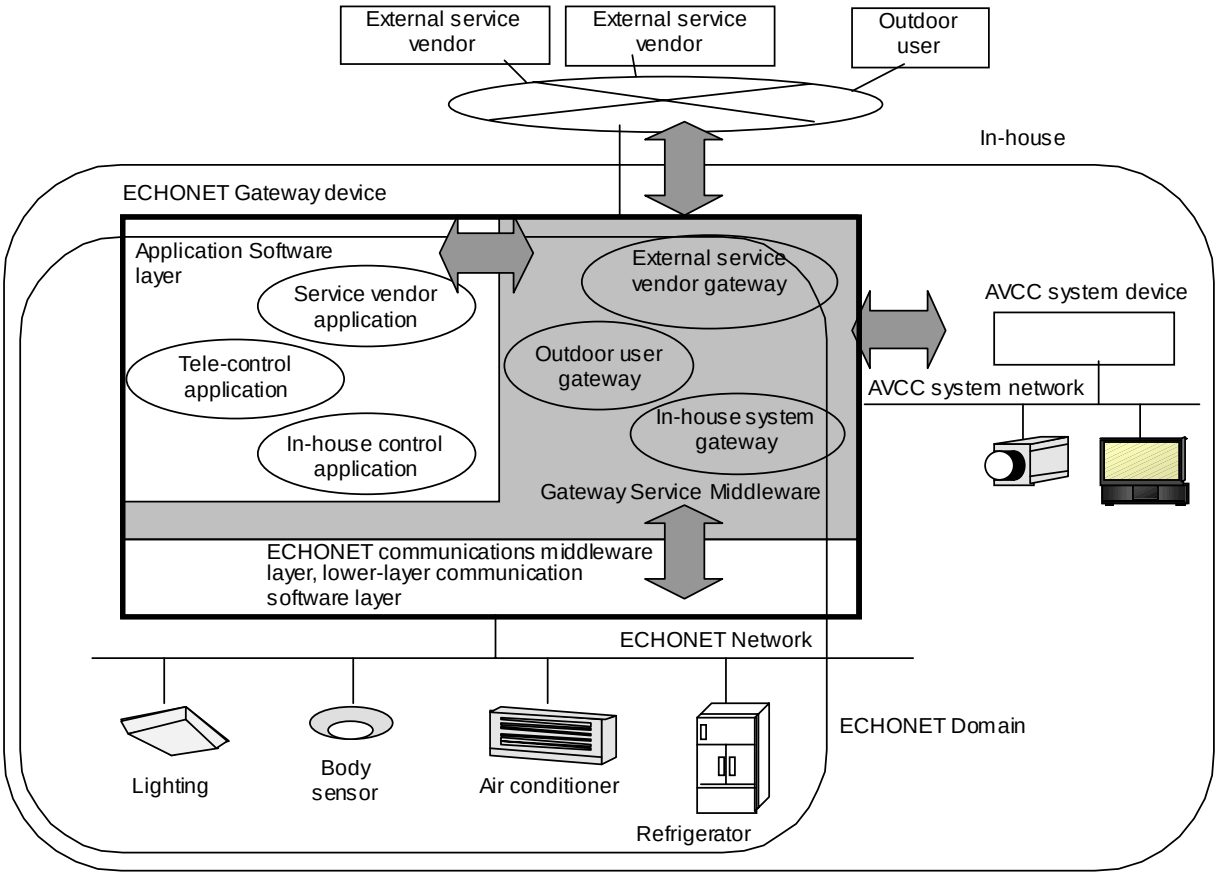

Fig. 1.1 Positioning of ECHONET Gateway Service Middleware

1.5 Software Configuration

The software of the ECHONET Gateway Service Middleware consists of two blocks divided according to function: one block to perform processing dependent on the ECHONET Domain and one block to perform processing dependent on an external system. Since the former performs basic functions independent of external system type, it is defined as a “gateway basic block”. The latter, which performs functions dependent on each external system, is termed a “gateway discrete block”.

ECHONET Gateways must always include the gateway basic block, but the gateway discrete block is optional.

Table 1.1 Mounting Specification of ECHONET Gateway Service Middleware Component Software

Gateway basic block	Required
Gateway discrete block	Optional

Figure 1.2 shows the relationship between the gateway basic block and gateway discrete block of the ECHONET Gateway, external system, ECHONET Gateway application, and ECHONET Communication Middleware. Here, “external system” signifies a system outside the ECHONET Domain, an in-home or external system other than ECHONET, or an ECHONET system in another domain.

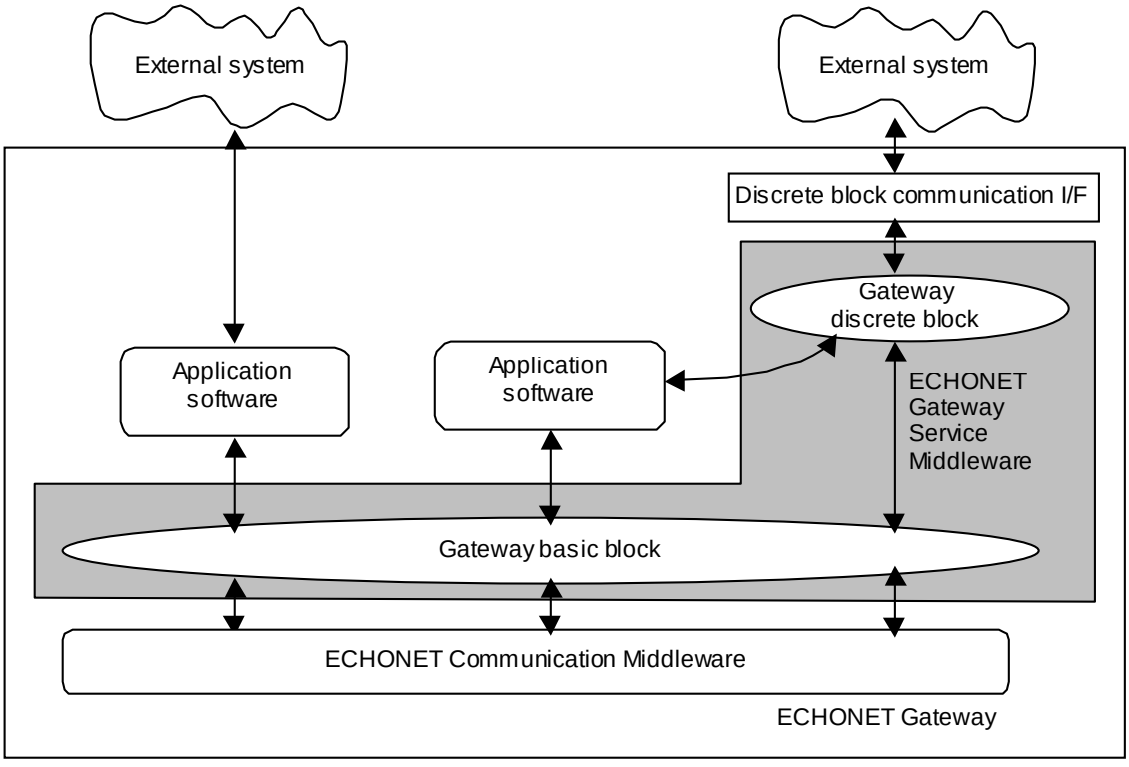

Fig. 1.2 Software Configuration in ECHONET Gateway

Chapter 2 Functions of Gateway Basic Block

This chapter describes the functions of the gateway basic block of the ECHONET Gateway Service Middleware. The major roles of the gateway basic block are: to mask access to ECHONET Devices in the ECHONET Domain to avoid disclosing all information in the domain to external systems; to perform control and monitoring of ECHONET Devices; and to notify connection/disconnection for plug and play.

(1) Access control function

When the ECHONET Gateway Service Middleware receives a request to use the functions noted in items (2) to (6) below, including requests for access to ECHONET Devices, this function controls access by judging whether processing is acceptable in terms of ECHONET Domain user security.

(2) Direct access function

This function accepts requests for control or status reference to an ECHONET Device from external systems through the gateway discrete block or application, delivers them to the ECHONET Communication Middleware, and returns a response, thereby relaying access to the ECHONET Device.

(3) Asynchronous access function

This function requests asynchronous access to ECHONET Devices from external systems through the gateway discrete block or application. This function accepts reservations for processing and executes processing at or after the specified time.

(4) Message filtering function

This function filters messages from the gateway discrete block, application, or message (alarm information) in the ECHONET system to provide exchange of specified messages only.

(5) ECHONET device status monitoring function

This function monitors the status of the specified ECHONET Device, holds related information, and sends this information in response to requests for reference from the gateway discrete block or applications.

(6) ECHONET device connection status change notice function

This function monitors the connection status of each ECHONET Device, holds related information, and sends this information in response to requests for reference from the gateway discrete block or applications. This function also notifies the gateway discrete block or an application of a change in the connection status of the ECHONET Device.

Table 2.1 shows the mounting specifications for these functions in the gateway basic block.

Table 2.1 Mounting Specification of Gateway Basic Block Functions

Function	Mounting specification
Access control function	Required
Direct access function	Required
Asynchronous access function	Optional
Message filtering function	Optional
ECHONET device status monitoring function	Optional
ECHONET device status change notice function	Optional

Fig. 2.1 shows the relationship between these functions.

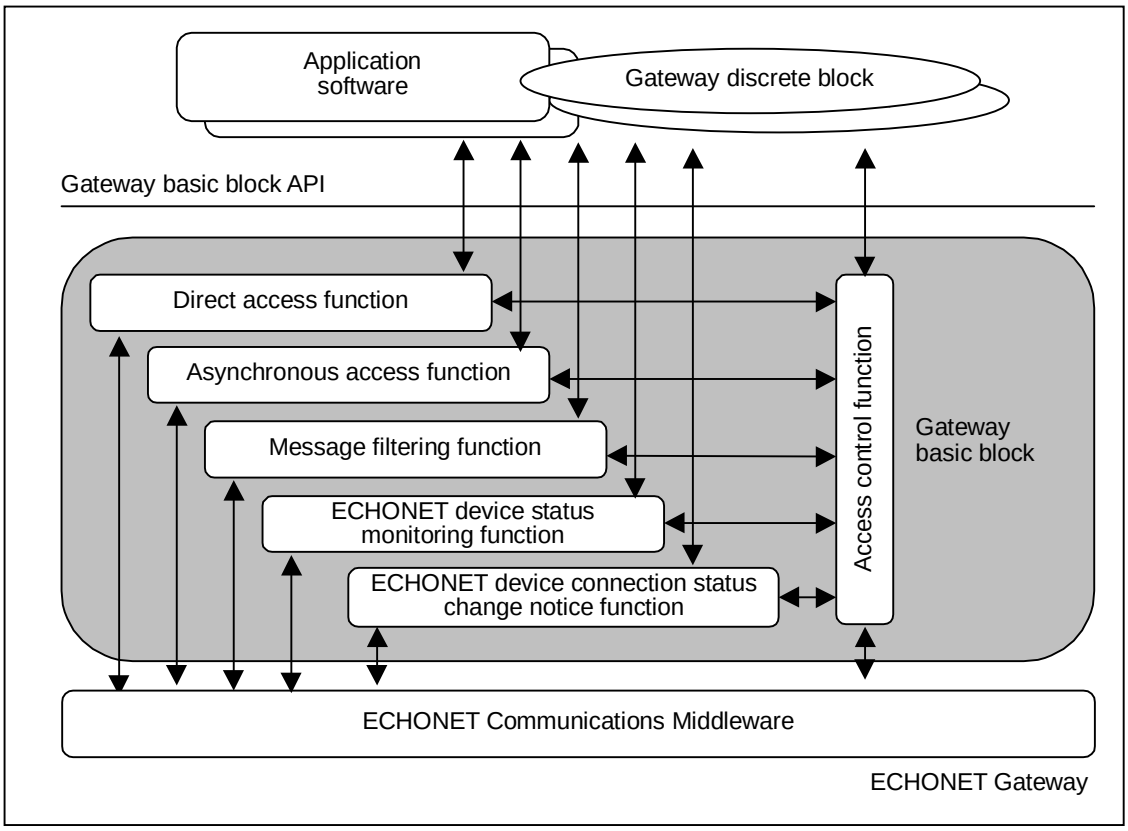

Fig. 2.1 Gateway Basic Block Functions and Inter-Relationships

Chapter 3 Functions of Gateway Discrete Block

To provide interchangeable connections with external systems, it would be better from the standpoint of development efficiency to specify a connection specification for each external system and to mount the gateway discrete block in accordance with this specification. The functions provided by the gateway discrete block are shown below. The gateway discrete block is an optional function and need not always be mounted. Gateway discrete block specifications are provided for each external system, but basically the following functions are mounted:

(1) ECHONET system view providing function

This function lets an external system view the ECHONET system and provides a referencing method for device operations and information and corresponding items. This function also performs protocol translation.

(2) Authorizing function

Upon receiving a request for using a gateway basic block function of the ECHONET Gateway Service Middleware, such as a request for access to an ECHONET Device, this function authorizes the requesting source.

Chapter 4 Functions of Gateway Basic Block

The gateway basic block has the basic function of disclosing the ECHONET Domain to external systems and converting it to an interface in which application software on the ECHONET Gateway or the gateway discrete block is dependent on each external system. Below, these functions will be referred to collectively as “applications” unless either application software or gateway discrete block is specified.

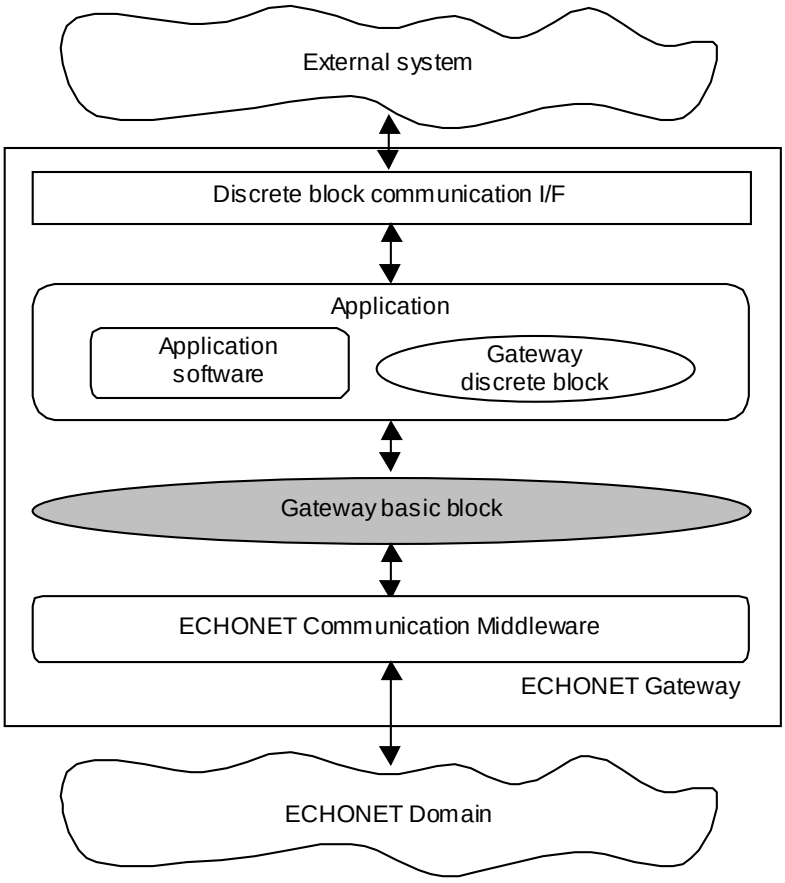

Fig. 4.1 Relationships in Gateway Basic Block

4.1 Access Control Function

This function checks function access permission as a security check for requests to access gateway basic block functions, such as access to an ECHONET Device. This control is exerted according to the registration contents in the access control list stored by the gateway basic block. Registration should be performed by the in-home user.

The following information is stored as a rule for disclosing the access control list to outside systems. Fig. 4.2 shows a processing sequence example in which the access control function is performed upon receipt of a request for executing a direct access function to be described later. In the figure, “Gateway” refers to the gateway basic block. (The same is true in the following “Explanation of Functions”.)

(1) List of device objects to be disclosed outside the system
A list of ECHONET address (EA), class identifier, and instance identifier (EOJ) sets.

(2) Access rule
Information on whether Set/Get for each object property is executed.

(3) List of accessible users
A list of combinations of accessible user (or application) identifiers and items (1) and (2) above.

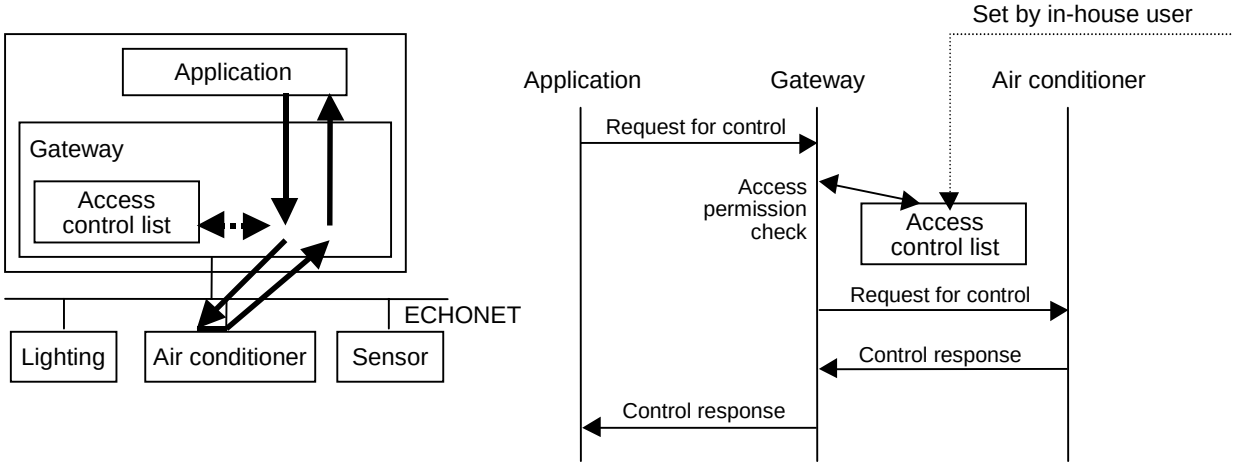

Fig. 4.2 Processing Sequence Example for Access Control Function

4.2 Direct Access Function

This function allows an application to access an ECHONET Device. Fig. 4.3 shows a processing sequence example for this function. For example, the function may be used to set the temperature of an air conditioner or to refer to the operation mode from an external system.

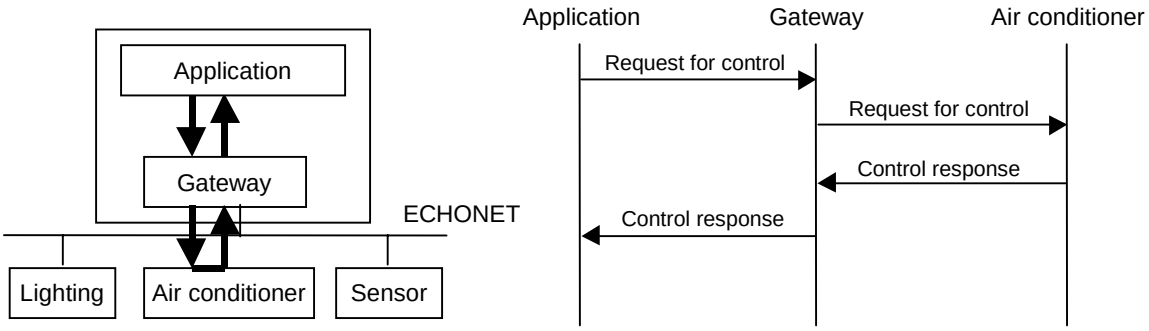

Fig. 4.3 Processing Sequence Example for Direct Access Control Function

4.3 Asynchronous Access Function

This function executes a request for asynchronous access from an application to an ECHONET Device. It accepts reservations for processing and executes processing at or after the specified time.

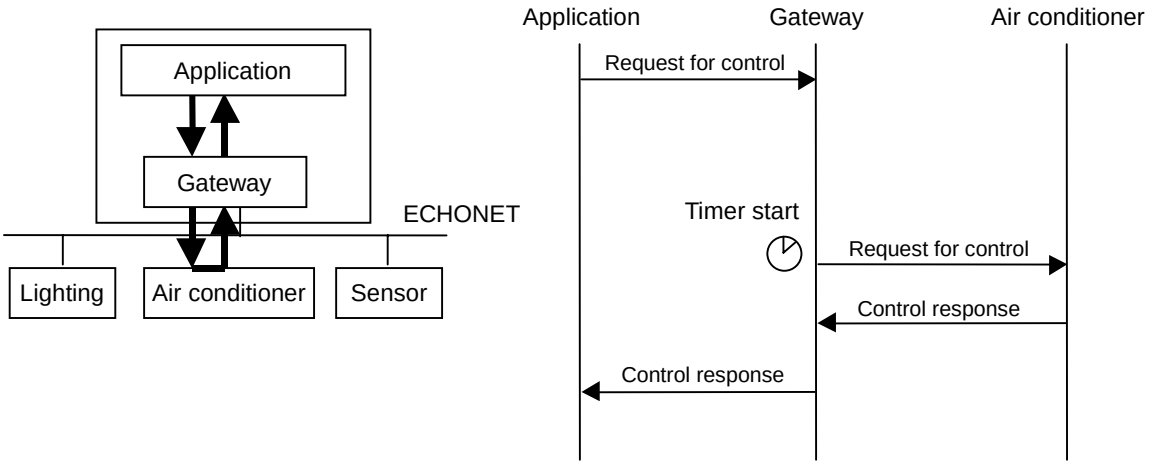

Fig. 4.4 Processing Sequence Example for Asynchronous Access Function

4.4 Message Filtering Function

This function filters messages (for example, fire alarm information in the ECHONET Domain) to be delivered to an external system and delivers only the previously specified type of message to the application. It receives status change announcement data for the ECHONET Device object, collects status information for the device object using the ECHONET Device status monitoring function described later, judges whether the status of this device object matches previously registered conditions, and notifies the application of the status value if it matches.

Fig. 4.5 shows a processing sequence example for filtering the data in the ECHONET Domain and transferring only information on the previously specified data to an application. The application performs filter setting for a message received for the gateway basic block beforehand (e.g., transmitting source object class, instance and property set are registered). The gateway basic block abandons all received data other than the contents of the registered filter. Upon receiving data matching this registered filter, the gateway basic block transfers the information to an application.

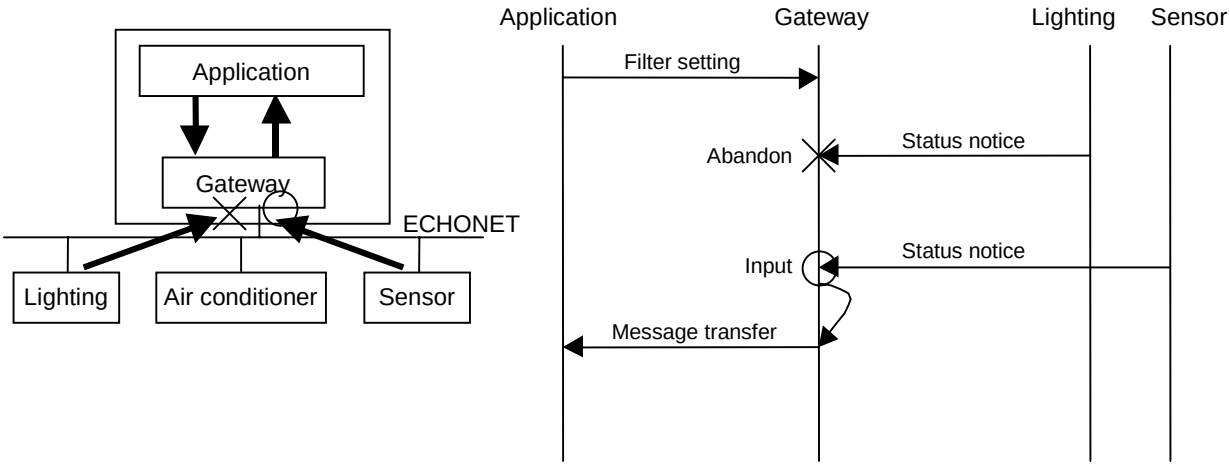

Fig. 4.5 Processing Sequence Example for Message Filtering Function

4.5 ECHONET Device Status Monitoring Function

This function monitors the status of the specified ECHONET Device and stores the information in the Gateway. The optional properties of an ECHONET Device's optional object that are specified by an application are monitored periodically, and the information is stored. For requests from the application, the information is returned as a response.

Fig. 4.6 shows a processing sequence example in which the ECHONET Gateway basic block monitors the operation mode status of an air conditioner by polling and returns a response to a request for acquisition from the application. The collected device status information is transferred to the aforementioned message filtering function and checked to determine whether it matches the conditions for notification to an external system.

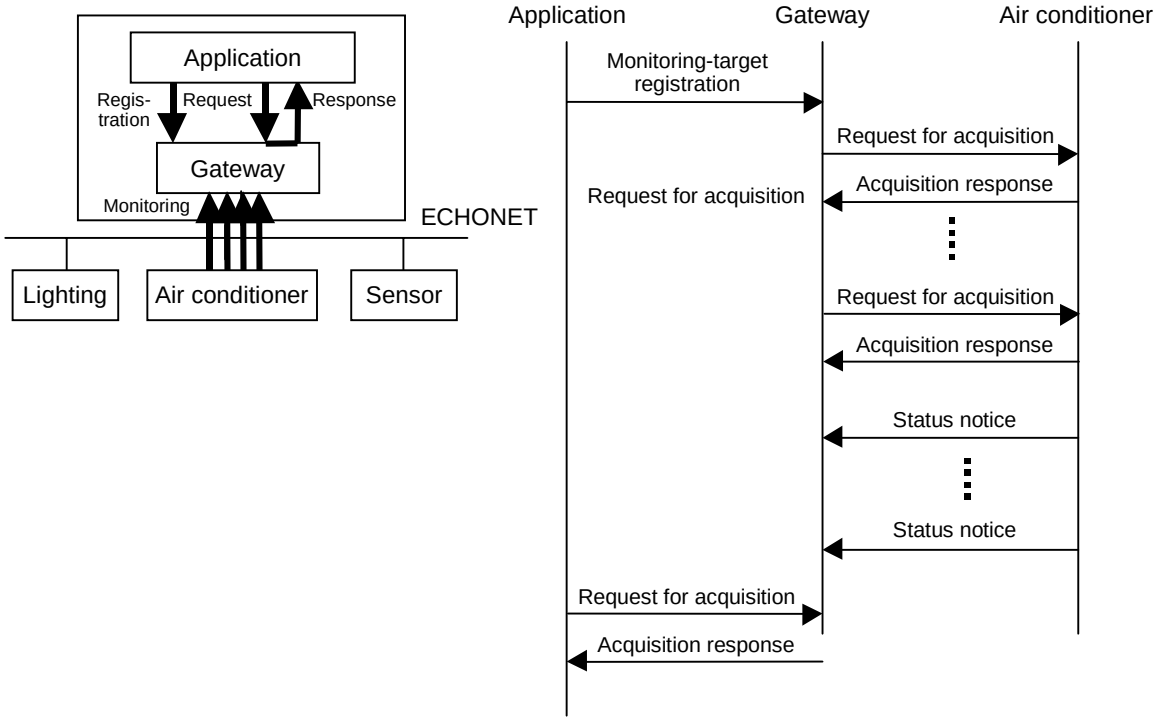

Fig. 4.6 Processing Sequence Example for ECHONET Device Status Monitoring Function

4.6 ECHONET Device Connection Status Change Notice Function

This function monitors the connection status of each ECHONET Device, stores the information, and notifies an application of a connection status change. It performs connection detect processing and disconnection detect processing for registered device types from an application beforehand, and notifies the application when it detects a change in connection status. Specific methods for connection detect processing and disconnection detect processing are not specified.

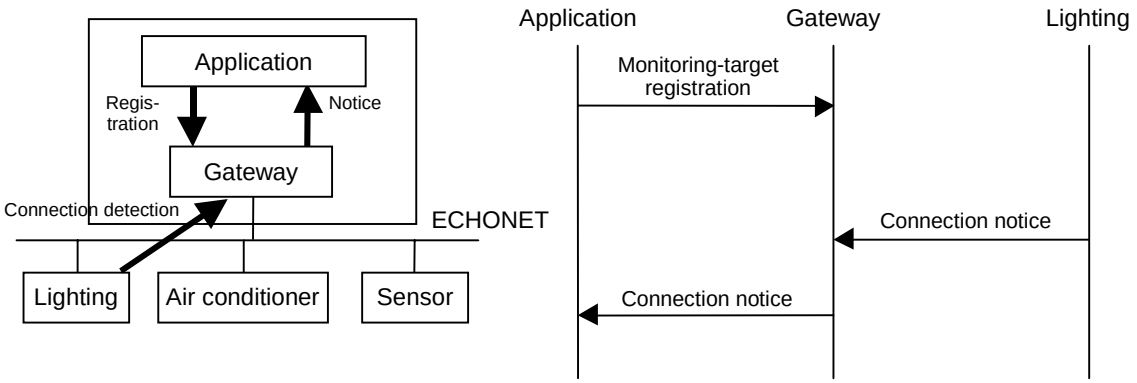

Fig. 4.7 Processing Sequence Example for ECHONET Device Connection Status Change Notice Function

Chapter 5 Functions of Gateway Discrete Block

The gateway discrete block differs with each system model of the external system to be connected and each communication protocol. Fig. 5.1 shows relationships in the gateway discrete block. This chapter provides specifications for each external system to be connected.

Here, external systems include HAVi, OpenPLANET, BACnet, and so on.

Specific types and detailed specifications of external systems will be reviewed after Ver. 1.0 and discussed in further detail in the future.

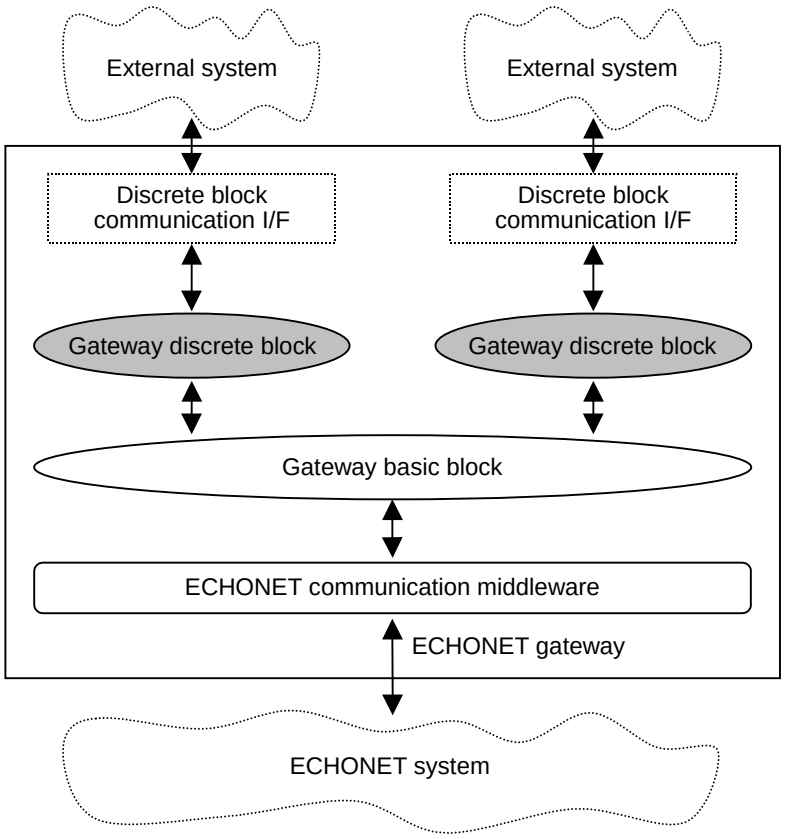

Fig. 5.1 Relationship in the Gateway Discrete Block

Chapter 6 Definition of Gateway Basic Block Object

This chapter specifies the object definition of the gateway basic block of the ECHONET Gateway Service Middleware. Other ECHONET Devices in the ECHONET Domain can control this block and obtain its status using the ECHONET protocol in accordance with this specification.

6.1 Class Number

The gateway basic block shall be provided with a class code so that it can be accessed by the ECHONET protocol from another ECHONET Node, and its assignment shall be as shown below. For the instance code, an optional value can be obtained in the following range:

Class group code	: 0x0D
Class code	: 0xDF
Instance code	: 0x01 to 0x7F (0x00: All-instance specification code)

6.2 Object Service

This object shall properly support necessary services from the object services described in Part 2.

6.3 Object Property

The gateway basic block supports the object properties shown below. Mentioned properties are used to access gateway basic block objects using the ECHONET protocol.

Table 6.1 Object Properties of Gateway Basic Block

Property code EPC	Mounting specification	Access rule	Property name	Data type	Explanation	Value description, range, etc.
0x80	Required	Get/Set	State	unsigned char	Status of gateway basic block	0x30,0x31, 0xFF
0xB1	Required	Get/Set*	Disclosed_ Object	array of struct	Arrangement of object EOJ as element for disclosure to external systems	
0xB2	Required	Get/Set*	ACL	array of struct	Access control list	
0xB3	Required	Get/Set*	User_ID	long	User ID authorized to use gateway object functions	
0xB4	Optional	Get	Async_Job_List	array of struct	Queue of asynchronous execution jobs	
0xB5	Optional	Get	Max_Filter	unsigned short	Maximum allowable registration number of following filters	Range 0-65535
0xB6	Optional	Get/Set	Activate_Filter	char	Status of filtering function	0x30,0x31, 0x32
0xB7	Optional	Get/Set	filter	array of struct	Registered filter	
0xB8	Optional	Get	Max_ Monitor	unsigned short	Maximum allowable registration number of following monitors	Range 0-65535
0xB9	Optional	Get/Set	Activate_ Monitor	char	Status of ECHONET Device status monitoring function	0x30,0x31
0xBA	Optional	Get/Set	Monitor	array of struct	Object property of polling target	
0xBB	Optional	Get	Max_Alive	unsigned short	Maximum allowable registration number of following Monitor_alive	Range 0-65535
0xBC	Optional	Get/Set	Activate_ Monitor_Alive	char	Operation status of device connection status monitoring function	0x30,0x31, 0x32
0xBD	Optional	Get/Set	Monitor_Alive	array of struct	Device type (object class) to be monitored by device connection status monitoring function	

* The access rule differs with each opposite-side application. Usually, only “Get” is to be enabled, with applications permitting “Set” limited to the ECHONET Domain administrator or proprietor. The ECHONET Gateway should be mounted so that this can be set using any user interface.

Each property is explained below.

(1) State

Status of gateway basic block, which takes one of the following values:

0x30: Operating status

0x31: Stop status

0xFF: Error stop status

(2) Disclosed_Object

A list of device objects to be disclosed to external systems. Specifically, this is a list of device object ECHONET Address (EA), class identifier and instance identifier (EOJ), and disclosed properties (EPC) sets. This is defined as an array of data having the following structure:

Table 6.2 Structure Members Comprising the Disclosed_Object Property

Member name	Member in the structure member	Data type	Explanation	Value description, range, etc.
EA		array of unsigned char	ECHONET address	
	NetID	unsigned char	NetID	See Part 2
	NodeID	unsigned char	NetID	See Part 2
EOJ		array of unsigned char	ECHONET object code	
	ClassGroup	unsigned char	Class group code	See Part 2
	Class	unsigned char	Class code	See Part 2
	Instance	unsigned char	Instance code	See Part 2
	Reserved	unsigned char	Filter	
EPC_list		array of struct	ECHONET property code	See Part 2

The EPC_list is a structure type consisting of the following members:

Type : Description type. 1: Description type 1, 2: Description type 2

Map : Enable/disable information map

For details of each member, see Part 2, Appendix 2 Property Map Description Types.

(3) ACL

Set/Get execution enable/disable information for each object property corresponding to the above item (2).

ACL[0]: “Set” execution enable/disable information

ACL[1]: “Get” execution enable/disable information

Each execution enable/disable information is a structure type consisting of the following members:

Type : Description type. 1: Description type 1, 2: Description type 2

Map : Enable/disable information map

For the details of each member, see Part 2, Appendix 2 Property Map Description Types.

(4) User_ID

Accessible user (or application) identifier. No identifier is specified.

(5) Async_Job_List

A list of information indicating access contents to ECHONET Devices waiting for execution by the alternative access function. It is an array of sets of time information and contents of control execution data.

Table 6.3 Structure Members Comprising the Async_Job_List Property

Member name	Member in the structure member	Data type	Explanation	Value description, range, etc.
Time	Year	unsigned short	Year (Gregorian year)	
	Month	unsigned char	Month	
	Day	unsigned char	Day	
	Hour	unsigned char	Hour	
	Minute	unsigned char	Minute	
	Second	unsigned char	Second	
ECHONET_ Message		array of unsigned char	ECHONET data	

(6) MAX_Filter

Maximum number of filters that can be registered in the gateway basic block. Usually it takes a fixed value.

(7) Activate_Filter

Filter status.

0x30: Stop status. In this status, messages are not transferred to the higher-layer application software or the gateway discrete block.

0x31: Operating status. The registered filter of (8) below is valid.

0x32: Through status. Every message received by the gateway is transferred to the higher-layer application software or the gateway discrete block. However, a security check is executed based on the information for access control held in (1) to (3) above. Only messages judged to be OK for safe disclosure to external systems are transferred.

(8) Filter

An array of structures to store the values of registered filters. The maximum number of arrays is equal to the value in (6) Max_Filter above. The filter is a structure having the following members:

Table 6.4 Members Comprising the Filter Property

Member name	Member in the structure member	Data type	Explanation	Value description, range, etc.
EA		array of unsigned char	ECHONET address	
	NetID	unsigned char	NetID	See Part 2
	NodeID	unsigned char	NodeID	See Part 2
EOJ		array of unsigned char	ECHONET Object Identifier Code	
	ClassGroup	unsigned char	Class group code	See Part 2
	Class	unsigned char	Class code	See Part 2
	Instance	unsigned char	Instance code	See Part 2
	Reserved	unsigned char	Filler	
EPC		unsigned char	ECHONET property code	See Part 2
Logic		unsigned char	Comparison condition logic with value of following “Data” division on notification to application	0x00: no check 0x01: “=“ 0x02:: “>“ 0x03: “<“ 0x04: “>=“ 0z05: “<=“ 0x06: “≠” 0x07-0x7F:Reserved 0x80-0xFF:User Defined
Value	Type	unsigned char	Data type of following “Data”	0x00: char 0x01: unsigned char 0x02: short 0x03: unsigned short 0x04: long 0x05: unsigned long 0x06: float(4bytes) 0x07-0x7F:Reserved 0x80-0xFF:User Defined
	Data		Property value	Property-dependent

Regarding value comparison, the values described in “Value description, range, etc.” of “Logic” and “Type” must be supported. (Note: When the property is defined by structure, this function shall not be available.)

(9) MAX_Monitor

Maximum pieces of device status monitoring target information (monitoring-target device properties) that can be registered in the gateway basic block. Usually, this value is fixed.

(10) Activate_Monitor

Operation status of the ECHONET Device status monitoring function.

0x30: Stop status

0x31: Operating status. In this status, the device status monitoring target information registered in (11) below is valid.

(11) Monitor

An array of structures to store the registered device status monitoring target information. The maximum number of arrays is equal to the value of Max_Monitor in (9) above. Monitor is a structure having the following members:

Table 6.5 Members Comprising the Monitor Property

Member name	Member in the structure member	Data type	Explanation	Value description, range, etc.
EA		array of unsigned char	ECHONET address	
	NetID	unsigned char	NetID	See Part 2
	NodeID	unsigned char	NodeID	See Part 2
EOJ		array of unsigned char	ECHONET Object Identifier Code	
	ClassGroup	unsigned char	Class group code	See Part 2
	Class	unsigned char	Class code	See Part 2
	Instance	unsigned char	Instance code	See Part 2
	Reserved	unsigned char		
EPC		unsigned char	ECHONET property code	See Part 2

(12) MAX_Alive

Maximum number of device connection status monitor target device types that can be registered in the gateway basic block (number of property array elements of (14) below). Usually a fixed value.

(13) Active_Monitor_Alive

Operation status of the ECHONET Device connection status monitoring function.

0x30: Stop status. In this status, device connection status changes are not transferred to the higher-layer application software or the gateway discrete block.

0x31: Operating status. The connection status monitoring target information of (14) is valid.

0x32: Through status. In this status, connection status change for every device is transferred to the higher-layer application or the gateway discrete block.

(14) Monitor_Alive

An array of structures to store registered device status monitoring target information. The maximum number of arrays is defined as the value of Max_Alive in (12) above. In this case, X1 (class group code) and X2 (class code) of EOJ are stored as the device type. Thus, the devices of the class group and the class registered here are put into connection status monitoring state.

Table 6.6 Members Comprising the Monitor_Alive Property

Member name	Member in the structure member	Data type	Explanation	Value description, range, etc.
ClassGroup		unsigned char	Class group code	See Part 2
Class		unsigned char	Class code	See Part 2

6.4 State Transition

The following three states are noted as gateway basic block states and are reflected in the state property value. Fig. 6.1 shows a state transition diagram.

Table 6.7 States of Gateway Basic Block

Status	State property value
Stop status	0x30
Operating status	0x31
Error stop status	0xFF

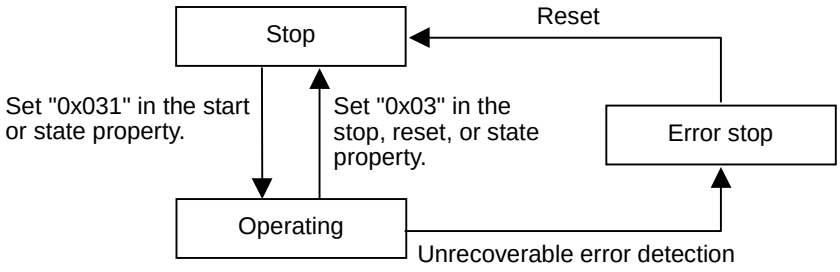

Fig. 6.1 Object State Transition

6.5 Operations

The gateway basic block is manipulated by an event from the ECHONET Communication Middleware, an event from the gateway basic block API, or an internal event. The operation of the gateway basic block upon receipt of one of these events for each status is shown in the table below.

Table 6.8 Operations of Gateway Basic Block upon Occurrence of Events

Event	Operating status	Stop status	Error stop status
A request for processing related to a service request is received through the basic API upon receipt of a service from another ECHONET Device.	Self-processing is performed according to the contents of the requested service. For example, a property value is set. A response is returned by referring to the property value.	←	←
A request for setting to a property is received through the gateway basic API.	Access permission is checked, and the specified value is then set as a property value.	An error is returned.	An error is returned if possible.
A request for referring a value to a property is received through the gateway basic API.	Access permission is checked, and the value of the specified property is then read and its value returned.	An error is returned.	An error is returned if possible.
A request for access to an asynchronous device is received.	The job in the gateway is pooled and inserted in an empty element of the Async_Job_List property.	An error is returned.	An error is returned if possible.
The execution timing of access to an asynchronous device is taken.	The job in the property is checked to determine whether a job to be executed exists. If such a job is found, it is executed and deleted from the Async_Job_List property.	N/A	N/A
A new device connection is detected.	The device profile is checked. If any device whose participation must be notified to an application is found, it is notified through the gateway basic API.	N/A	N/A
Disconnection of a device is detected.	If any device whose disconnection must be notified to an application is found, it is notified through the gateway basic API.	N/A	N/A
A timer trigger to monitor the device status occurs.	A request for referring to the property value is transmitted to the device.	N/A	N/A
The comparison logic for monitoring matches.	The occurrence of an event is notified to an application.	N/A	N/A
Notice data is received from an ECHONET Device.	A filter check is made. If any data to be notified to an application is found, it is notified.	The event is ignored.	The event is ignored.
An unrecoverable error is detected.	The current status proceeds to the error stop status.	The current status proceeds to the error stop status.	N/A
Start event	An error is returned.	The current status proceeds to the operating status.	The event is ignored.
Stop event	The current status proceeds to the stop status.	An error is returned.	The event is ignored.
Reset event	The current status proceeds to the stop status.	The current status proceeds to the stop status.	Initialize processing is executed, if possible, to proceed to the stop status.

"←" means contents are the same as column to the left.

Chapter 7 Gateway Basic Block APIs

7.1 Function Overview

The gateway basic block API functions are outlined here.

The gateway basic block is provided with APIs to implement the functions described below. However, multiple functions may be grouped and implemented as a single API, or one function may be implemented by multiple APIs.

(1) Gateway basic block start/stop

Accepts and processes request for starting/stopping gateway basic block.

(2) Access limitation information registration/deletion

Registers or deletes limitation information at access limitation for an ECHONET Device.

(3) Access limitation information reference

References limitation information at access limitation for an ECHONET Device.

(4) Request for device control

Accepts request for control and for status reference to an ECHONET Device and transfers it to ECHONET Communication Middleware.

(5) Control response

Receives and returns response to request for control and request for status reference transmitted to ECHONET Device from ECHONET Communication Middleware.

(6) Alternative access

Accepts requests for executing control or status reference to an ECHONET Device at or after the specified time. A response is returned by (5) above.

(7) Filter message reception

Filters and receives only specified messages from among optional messages in the ECHONET system.

(8) Message filter registration or deletion

Registers or deletes type of message to be received by filter message reception API above.

(9) Device status monitoring reference

Accepts request for reference to ECHONET Device status monitored and stored by gateway basic block, references it, and returns its result.

(10) Device status monitoring registration or deletion

Registers or deletes an ECHONET Device, object, or property type whose status is to be monitored by the gateway basic block.

(11) Device connection notice

Monitors connection status of specified ECHONET Device. If connection is detected, gateway discrete block or application is notified.

(12) Device disconnection notice

Monitors connection status of specified ECHONET Device. If disconnection is detected, gateway discrete block or application is notified.

(13) Device connection status reference

Returned for request for reference to ECHONET Device connection status information held by gateway basic block.

(14) Device connection status change notice target registration/deletion

Registers or deletes target device whose connection status is to be monitored.

Table 7.1 List of API Functions

No.	API function	Directly related function
(1)	Gateway basic block start/stop	All functions
(2)	Access limitation information registration/deletion	Access control function
(3)	Access limitation information reference	Access control function
(4)	Request for device control	Direct access function
(5)	Device control response	Direct access function
(6)	Alternative access	Alternative access function
(7)	Filter message reception	Message filtering function
(8)	Message filter registration/deletion	Message filtering function
(9)	Device status monitoring reference	ECHONET Device status monitoring function
(10)	Device status monitoring registration/deletion	ECHONET Device status monitoring function
(11)	Device connection notice	ECHONET Device connection status change notice function
(12)	Device disconnection notice	ECHONET Device connection status change notice function
(13)	Device connection status reference	ECHONET Device connection status change notice function
(14)	Device connection status change notice target registration/deletion	ECHONET Device connection status change notice function

7.2 Level 1

Input/output items are specified for the function level APIs described in “Function Overview”.

Table 7.2 List of Level 1 APIs (1/2)

No.	Level 1 API	Directly related function	Mounting specification
(1)	Gateway basic block start API	All functions	Required
(2)	Gateway basic block stop API	All functions	Required
(3)	Disclosed object registration API	Access control function	Required
(4)	Disclosed object deletion API	Access control function	Required
(5)	ACL setting API	Access control function	Required
(6)	Access enable user registration API	Access control function	Required
(7)	Access enable user registration API	Access control function	Required
(8)	Access enable user deletion API	Access control function	Required
(9)	Disclosed object list reference API	Access control function	Required
(10)	ACL reference API	Access control function	Required
(11)	Access enable user list reference API	Access control function	Required
(12)	Device control request API	Direct access function, alternative access function	Required
(13)	Control response API	Direct access function, alternative access function	Required
(14)	Filter message reception API	Message filtering function	Optional
(15)	Message filter registration API	Message filtering function	Optional
(16)	Message filter deletion API	Message filtering function	Optional
(17)	Device status monitoring reference API	ECHONET Device status monitoring function	Optional
(18)	Device status monitoring registration/deletion API	ECHONET Device status monitoring function	Optional
(19)	Device status monitoring registration/deletion API	ECHONET Device status monitoring function	Optional
(20)	Device connection notice API	ECHONET Device connection status change notice function	Optional
(21)	Device disconnection notice API	ECHONET Device connection status change notice function	Optional
(22)	Device connection status reference API	ECHONET Device connection status change notice function	Optional
(23)	Connected device list reference API	ECHONET Device connection status change notice function	Optional
(24)	Device connection status change notice target registration API	ECHONET Device connection status change notice function	Optional
(25)	Device connection status change notice target deletion API	ECHONET Device connection status change notice function	Optional

Table 7.2 List of Level 1 APIs (2/2)

No.	Level 1 API	Directly related function	Mounting specification
(26)	Function start API	Message filtering function, ECHONET Device status monitoring function, ECHONET Device connection status change notice function	Optional
(27)	Function stop API	Message filtering function, ECHONET Device status monitoring function, ECHONET Device connection status change notice function	Optional

(1) Gateway basic block start API
Accepts request for starting gateway basic block and performs processing.
Input/output data is not specified.

(2) Gateway basic block stop API
Accepts request for stopping gateway basic block and performs processing.
Input/output data is not specified.

(3) Disclosed object registration API
Registers ECHONET object to be disclosed to external systems.

Table 7.3 Disclosed Object Registration API Inputs/Outputs

	Data	Explanation	Mounting specification
Input	EA	ECHONET address	Required
Input	EOJ	ECHONET object identifier	Required

(4) Disclosed object deletion API
Cancels registered ECHONET object that is disclosed to external systems.

Table 7.4 Disclosed Object Deletion API Inputs/Outputs

	Data	Explanation	Mounting specification
Input	EA	ECHONET address	Required
Input	EOJ	ECHONET object identifier	Required

(5) ACL setting API

Registers information on whether ECHONET object access is enabled for each property. However, registration shall be performed only for disclosed objects already set in (3) above.

Table 7.5 ACL Setting API Inputs/Outputs

	Data	Explanation	Mounting specification
Input	EA	ECHONET address	Required
Input	EOJ	ECHONET object identifier	Required
Input	Readable (Gettable) property list	Read enable property list	Required
Input	Settable property list	Write enable property list	Required

(6) Access enable user registration API

Registers user identifier allowed to access ECHONET Device through the gateway.

Table 7.6 Access Enable User Registration API Inputs/Outputs

	Data	Explanation	Mounting specification
Input	User identifier	User identifier	Required

(7) Access enable user deletion API

Deletes user identifier allowed to access ECHONET Device through the gateway.

Table 7.7 Access Enable User Deletion API Inputs/Outputs

	Data	Explanation	Mounting specification
Input	User identifier	User identifier	Required

(8) Disclosed object list reference API

References list of objects to be disclosed to external systems.

Table 7.8 Disclosed Object List Reference API Inputs/Outputs

	Data	Explanation	Mounting specification
Output	Object_List	List of identifiers of ECHONET objects to be disclosed.	Required

(9) ACL reference API

References the ACL.

Table 7.9 ACL Reference API Inputs/Outputs

	Data	Explanation	Mounting specification
Input	EA	ECHONET address	Required
Input	EOJ	ECHONET object identifier	Required
Output	Readable property list	Read enable property list	Required
Input	Settable property list	Write enable property list	Required

(10) Access enable user list reference API

References list of objects to be made accessible to external systems.

Table 7.10 Access Enable User List Reference API Inputs/Outputs

	Data	Explanation	Mounting specification
Output	User_List	List of access enable user identifiers	Required

(11) Device control request API

Accepts requests for control and for status reference to an ECHONET Device and transfers them to ECHONET Communication Middleware.

Table 7.11 Device Control Request API Inputs/Outputs

	Data	Explanation	Mounting specification
Input	EA	ECHONET address	Required
Input	EOJ	ECHONET object identifier	Required
Input	ESV	ECHONET service code	Required
Input	EPC	ECHONET property code	Required
Input	Data_Size	Transmission data size	Required
Input	Data	Transmission data	Required

(12) Control response API

Receives and returns response to requests for control and for status reference transmitted to an ECHONET Device from the ECHONET Communication Middleware. It is also possible to specify in ReceiveType that any control response received by a node is to be returned in sequence.

Table 7.12 Control Response API Inputs/Outputs

	Data	Explanation	Mounting specification
Input	ReceiveType	Reception type.*	Optional
Input*	EA	ECHONET address	Required
Input*	EOJ	ECHONET object identifier	Required
Input*	ESV	ECHONET service code	Required
Input*	EPC	ECHONET property code	Required
Output*	EA	ECHONET address	Required
Output*	EOJ	ECHONET object identifier	Required
Output*	ESV	ECHONET service code	Required
Output*	EPC	ECHONET property code	Required
Output	Data_Size	Received data size	Required
Output	Data	Received data	Required

*: Select either Input or Output exclusively. This may be mounted on a different API. If a single API is used for mounting, it is to be specified in ReceiveType.

(13) Alternative access API

Accepts request for executing control and status reference to an ECHONET Device at or after the specified time. A response is returned by (12) above.

Table 7.13 Alternative Access API Inputs/Outputs

	Data	Explanation	Mounting specification
Input	EA	ECHONET address	Required
Input	EOJ	ECHONET object identifier	Required
Input	ESV	ECHONET service code	Required
Input	EPC	ECHONET property code	Required
Input	Data_Size	Data size	Required
Input	Data	Data	Required
Input	Time	Time of execution	Required

(14) Filter message reception API

Filters and receives only specified messages from among optional messages in ECHONET system.

Table 7.14 Filter Message Reception API Inputs/Outputs

	Data	Explanation	Mounting specification
Output	EA	ECHONET address	Required
Output	EOJ	ECHONET object identifier	Required
Output	ESV	ECHONET service code	Required
Output	EPC	ECHONET property code	Required
Output	Data_Size	Data size	Required
Output	Data	Data	Required

(15) Message filter registration API

Registers type of message to be received by filter message reception API.

Table 7.15 Message Filter Registration API Inputs/Outputs

	Data	Explanation	Mounting specification
Input	Filter_member	Array element of filter property	Required

(16) Message filter deletion API

Deletes type of message to be received by filter message reception API.

Table 7.16 Message Filter Deletion API Inputs/Outputs

	Data	Explanation	Mounting specification
Input	Filter_member	Array element of filter property	Required

(17) Device status monitoring reference API

Accepts request for status reference of an ECHONET Device whose status is monitored and stored by gateway basic block, references it, and returns its result.

Table 7.17 Device Status Monitoring Reference API Inputs/Outputs

	Data	Explanation	Mounting specification
Input	EA	ECHONET address	Required
Input	EOJ	ECHONET object identifier	Required
Input	ESV	ECHONET service code	Required
Input	EPC	ECHONET property code	Required
Output	Data_Size	Data size	Required
Output	Data	Data (property value)	Required

(18) Device status monitoring registration API

Registers ECHONET Device, object, and property type whose status is to be monitored by gateway basic block.

Table 7.18 Device Status Monitoring Registration API Inputs/Outputs

	Data	Explanation	Mounting specification
Input	EA	ECHONET address	Required
Input	EOJ	ECHONET object identifier	Required
Input	ESV	ECHONET service code	Required
Input	EPC	ECHONET property code	Required
Input	Interval	Polling interval time	Optional

(19) Device status monitoring deletion API

Deletes ECHONET Device, object, and property type whose status is to be monitored by gateway basic block.

Table 7.19 Device Status Monitoring Deletion API Inputs/Outputs

	Data	Explanation	Mounting specification
Input	EA	ECHONET address	Required
Input	EOJ	ECHONET object identifier	Required
Input	ESV	ECHONET service code	Required
Input	EPC	ECHONET property code	Required

(20) Device connection notice API

Monitors connection status of specified ECHONET Device. If connection is detected, gateway discrete block or application is notified.

Table 7.20 Device Connection Notice API Inputs/Outputs

	Data	Explanation	Mounting specification
Output	EA	ECHONET address	Required
Output	EOJ_List	List of ECHONET object identifiers	Optional

(21) Device disconnection notice API

Monitors connection status of specified ECHONET Device. If disconnection is detected, gateway discrete block or application is notified.

Table 7.21 Device Disconnection Notice API Inputs/Outputs

	Data	Explanation	Mounting specification
Output	EA	ECHONET address	Required

(22) Device connection status reference API

This API returns requests to reference connection status information of ECHONET Device held by gateway basic block.

Table 7.22 Device Connection Status Reference API Inputs/Outputs

	Data	Explanation	Mounting specification
Input	EA	ECHONET address	Required
Output	Connective state	Connection status of ECHONET Device	Required

(23) Connected device list reference API

Returns list of connected ECHONET Devices.

Table 7.23 Connected Device List Reference API Inputs/Outputs

	Data	Explanation	Mounting specification
Output	EA_List	List of ECHONET address	Required

(24) Device connection status change notice target registration API

Registers target device for connection status monitoring.

Table 7.24 Device Connection Status Change Notice Target Registration API Inputs/Outputs

	Data	Explanation	Mounting specification
Input	ClassNo	Class group code and class code	Required

(25) Device connection status change notice deletion API

Deletes target device for connection status monitoring.

Table 7.25 Device Connection Status Change Notice Target Deletion API Inputs/Outputs

	Data	Explanation	Mounting specification
Input	ClassNo	Class group code and class code	Required

(26) Function start API

Switches operating status of message filtering function, ECHONET Device status monitoring function, and ECHONET Device connection status change notice function, which are functions of the gateway basic block, to start status. This API is also used to change the status of the message filtering function and the ECHONET Device connection status change notice function to operating status or through status.

Table 7.25 Function Start API Inputs/Outputs

	Data	Explanation	Mounting specification
Input	Function	Target function	Required
Input	Option	Status specification	Optional

(27) Function stop

Switches operating status of message filtering function, ECHONET Device status monitoring function, and ECHONET Device connection status change notice function, which are functions of the gateway basic block, to stop status.

Table 7.25 Function Stop API Inputs/Outputs

	Data	Explanation	Mounting specification
Input	Function	Target function	Required

7.3 Level 2

This section provides detailed specifications for APIs for specific operating systems (OS) and languages. These specifications are scheduled to be offered after Ver. 1.0.