ECHONET Lite providers
User's Guide

Version 1.0.0

March 4, 2021

NOTE:

[Revision History]

Version	Date	Content
1.0.0	2021-03-04	First edition

[Compatible models]

Model	Version	Notes
fuel cell class	Detailed Requirements for ECHONET Device objects Release N	
instantaneous water heater class	Detailed Requirements for ECHONET Device objects Release N	
storage battery class	Detailed Requirements for ECHONET Device objects Release N	

[Operation confirmed models]

Model	Version	Notes

Contents

1. Introduction

This document is a user's guide for providers who communicate with ECHONET Lite enabled devices. Figure 1-1 shows the overall configuration of this provider and the device. The providers are referred to as ECHONET Lite providers.

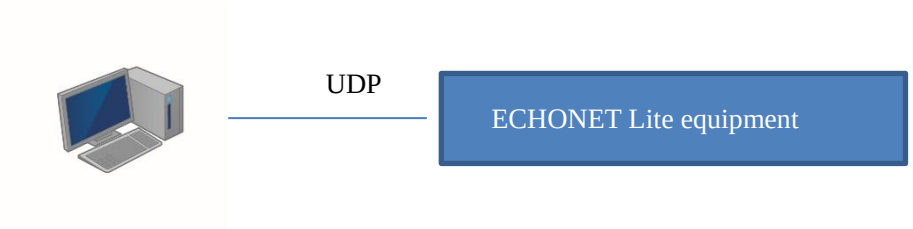

Fig. 1-1 Configuration Diagram

Figure 1-2 shows the correspondence between this provider and each device.
(* This is an example. It does not represent everything.)

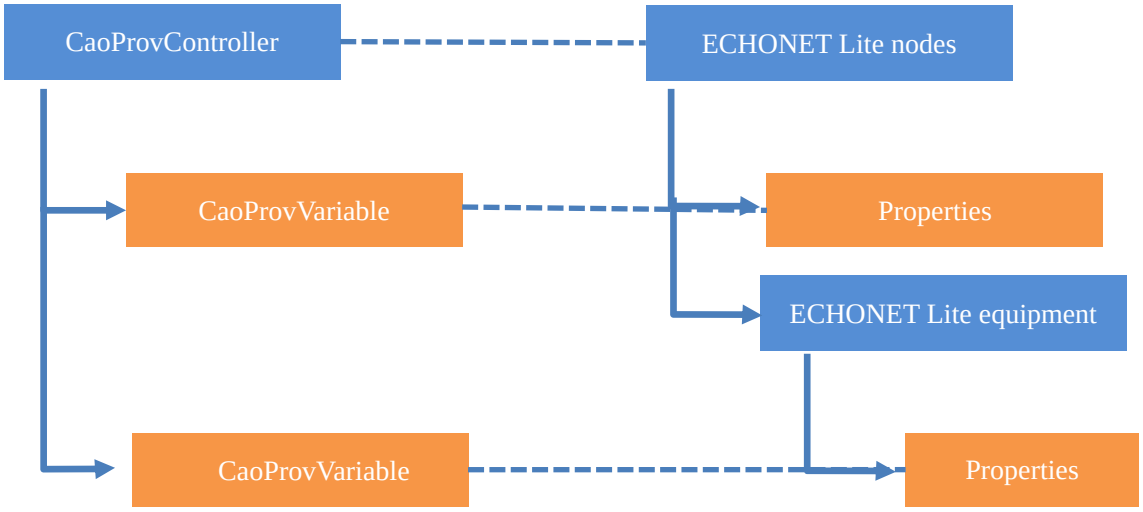

Fig. 1-2 Provider configuration and device information

2. Setting Up Your Environment for Application Development

2.1. Connecting ECHONET Lite to the client-PC

ECHONET Lite providers connect to ECHONET Lite via UDP communication.

Since ECHONET Lite providers use UDP port 3610 for transmission and reception, UDP port 3610 must be allowed to arrive in the firewall setting beforehand.

2.2. Setting up a PC development environment

2.2.1. Manually installing ECHONET Lite providers

If you install ECHONET Lite provider manually, you must register the registry below.

To register the registry, start the command prompt with administrator privileges, and execute DotNet ¥Bat¥ RegistAsm.bat in the folder where ORiN2 is installed using the provider's DLL as the parameter.

If you want to unregister the registry, start a command prompt with administrator privileges and run DotNet ¥Bat¥ UnregistAsm.bat under the folder where ORiN2 is installed using the provider's DLL.

When executing the command, either move to the path where the file is located or specify the file path.

Table 2-1 ECHONET Lite providers

File name	CaoProvECHONETLite.dll
ProgID	CaoProv.ECHONETLite
Registry registration	RegistAsm.bat CaoProvECHONETLite.dll
Deletion of Registry Registration	UnregistAsm.bat CaoProvECHONETLite.dll

3. Command Reference

3.1. Method/Property List

Table 3-1 List of methods and properties

Category	Methods/Properties[1]		Function	Reference
CaoWorkspace				
	AddController	M	Connected to controller	P.8
CaoController				
	Index	P	Obtaining the Controller Number	P.10
	Name	P	Get Controller Name	P.10
	Help	P	Getting Help	P.11
	GetVariableNames	M	Get a list of variable names that can be connected	P.11
	Variables	P	Retrieving Variable Collections Holded by the Controller	P.11
	AddVariable	M	Adding Variable Objects	P.11
	Execute	M	Execute Extended Commands	P.13
	OnMessage	E	Message reception event	P.16
CaoVariable				
	Index	P	Obtaining Variable Numbers	P.17
	Name	P	Retrieving Variable Names	P.17
	Help	P	Getting Help	P.17
	Value	P	Get/set value	P.17

3.2. Method properties

3.2.1. CaoWorkspace classes

3.2.1.1. AddController method

Add controller objects to CaoWorkspace.ECHONET Lite provider refers to the parameters passed when AddController method is running and connects to ECHONET Lite equipment connected to the relevant ECHONET Lite nodes. The following are the specifics of AddController method:

Format

AddController

[1] M:Indicates methods, P: properties, and E: events, respectively.

```
(
"<controller name>",              // Controller name (optional)
"CaoProv.ECHONETLite",            // Provider name (fixed)
"<machine name>",                         // Provider execution machine name (unused)
"<Option>"                  // Option character string (optional)
)
```

Option

The following is an optional specification for Option character string: Option character string is a comma (,) string consisting of the options listed below.

Option	Required	Description	Value Range	Default value
ObjectCode		See ObjectCode optional (Link) below.	0~16777215	-
Address	※1	See Address optional (Link) below.		-
IdentificationNumber	※1	See IdentificationNumber optional (Link) below.		-
Timeout		See Timeout optional (Link) below.	1~65535	1000
MyIP		See MyIP optional (Link) below.		0.0.0.0

※1 When ObjectCode is specified, either Address or IdentificationNumber is required.

Sample usage (C#)

```
// Engine
ORiN2.ManagedCAO.CCaoEngine engine = new ORiN2.ManagedCAO.CCaoEngine();
// Workspace
ORiN2.ManagedCAO.CCaoWorkspace workspace = engine.AddWorkspace("NewWrks", "");
// Controller
ORiN2.ManagedCAO.CCaoController controller=
Workspace.AddController("StorageBattery", "CaoProv.ECHONETLite", "",
"ObjectCode=0x027D01,IdentificationNumber=FE00007738C986312C2D0EF00100000000");
```

3.2.1.1.1. ObjectCode Optional

Specify the object code of the connected ECHONET Lite node or ECHONET Lite device.

If "0x" is present at the beginning of the character string, it is handled as a hexadecimal number; if it does not exist, it is handled as a decimal number.

By default, controller objects are added that can only be used for GetDeviceList and GetVariableList of extended methods.

3.2.1.1.2. Address Optional

Specify the IPaddress of ECHONET Lite node to which you want to connect.

If you specify ObjectCode, you must specify either Address or IdentificationNumber.

Address takes precedence if both Address and IdentificationNumber are specified at the same time.

3.2.1.1.3. IdentificationNumber Optional

Specify the identification number (EPC:0x83) of the connected ECHONET Lite node. The number is 17 hexadecimal digits (34 characters).

IdentificationNumber is handled as a hexadecimal number regardless of the presence or absence of the leading "0x" of the character string.

When IdentificationNumber is specified, the address of ECHONET Lite node is obtained based on the identification number of ECHONET Lite node, and communication is performed based on the acquired address.

If ECHONET Lite is operated in DHCP, the addressing may change. Use a IdentificationNumber.

Address takes precedence if both Address and IdentificationNumber are specified at the same time.

3.2.1.1.4. Timeout Optional

Specifies the number of milliseconds to wait for a reply after sending a request to a ECHONET Lite.

When GetDeviceList is executed, it waits only for Timeout after sending the request. During this time, it obtains ECHONET Lite that returned the response.

3.2.1.1.5. MyIP Optional

Specify the address to be used for sending and receiving on a PC with multiple network adapters.

If you have more than one network adapter, the unexpected network adapter may be used for communication, so be sure to specify it if you have more than one network adapter.

3.2.2. CaoController classes

3.2.2.1. Index Properties

Gets the controller number as a Long type (4-byte integer type). This number is used to retrieve/remove the corresponding controller from the controller collection held by CaoWorkspace.

Index property does not change when controller collection elements are added or removed.

Sample usage (C#)

```
// Get Index
Long index = controller.Index;
```

3.2.2.2. Name Properties

Retrieves the controller name specified by AddController method of CaoWorkspace class.

Sample usage (C#)

```
System.Diagnostics.Debug.WriteLine(controller.Name);
```

3.2.2.3. Help Properties

Acquires the name of ECHONET device specified by ObjectCode option.

Sample usage (C#)

```
// Get Help
String help = controller.Help;
```

3.2.2.4. GetVariableNames method

Retrieves a list of variable names for system variables that can be connected.

Format

```
GetVariableNames
(
"<Option>"                    // Option character string
)
```

Option

Option character string is not used by ECHONET Lite providers.

The specified Option character string is ignored.

Sample usage (C#)

```
// Get variable name list
String[] variableNames = controller.GetVariableNames("");
```

3.2.2.5. Variables Properties

Gets a collection of variables that the controller holds.

Sample usage (C#)

```
// Variable Collection Retrieval
ORiN2.ManagedCAO.CCaoVariables variables = controller.Variables;
// Variable acquisition
ORiN2.ManagedCAO.CCaoVariable variable = variables[0];
```

3.2.2.6. AddVariable method

Adds a variable object to CaoController. You can use the system variables shown in 3.3.2 for variable names and user variables that do not start with a "@".

AddVariable is specified as follows.

User variables use options to specify the properties to retrieve.

Option

The following is an optional specification for Option character string: Option character string is a comma (,) string consisting of the options listed below.

Option	Required	Description	Value Range	Default value
PropertyCode	✓	Code of the property to be acquired/set	0~255	-
ValueType		Value Type		UI1
TrueValue	※2	Value to be treated as a True when the value type is BOOL	0~255	-
FalseValue	※2	Value to be treated as a False when the value type is BOOL	0~255	
Digit		Number of digits after the decimal point	0~255	

※2 Required if BOOL is specified in ValueType.

Format

```
AddVariable
(
"<variable name>",                    // Variable Name
"<Option>"              // Option character string (optional)
)
```

3.2.2.6.1. PropertyCode Optional

Specify the code for the property of ECHONET Lite node or ECHONET Lite device to be acquired/set.

If "0x" is present at the beginning of the character string, it is handled as a hexadecimal number; if it does not exist, it is handled as a decimal number.

3.2.2.6.2. ValueType Optional

Specify the type of property of ECHONET Lite node or ECHONET Lite device to be acquired/set.

Refer to the type (Link) available on ECHONET Lite for more information.

3.2.2.6.3. TrueValue options, FalseValue options

Specifies what to treat as True and False when BOOL is specified in ValueType.

Required if BOOL is specified in ValueType.

If "0x" is present at the beginning of the character string, it is handled as a hexadecimal number; if it does not exist, it is handled as a decimal number.

3.2.2.6.4. Digit Optional

ECHONET Lite accepts a Digit of 0 in the (Link) of types for which decimal places can be specified by Digit. (e.g. if the unit is 0.001V, specify 3 as Digit.)

When Digit is specified, it is handled as VT_R8.

3.2.2.7. Execute method

Execute CaoController extension. Execute is specified as follows.

Format

```
Execute
(
"<extension command name>",          // Extended command name
"<Option string>"                    // Option character string (optional)
)
```

The following is a list of extended commands that can be specified in Execute. The usage examples are described in detail in the extended commands.

Command	Description	Reference
ProviderCancel	Keep the provider on cancellation.	P.13
ProviderClear	Cancels the cancel state of the provider and returns to normal operation.	P.14
GetDeviceList	Obtain a list of connectable ECHONET Lite nodes and ECHONET Lite devices.	P.14
GetVariableList	Gets a list of system variable information that can be added.	P.15
RefreshDeviceList	Update the list of connectable ECHONET Lite nodes and ECHONET Lite devices.	P.16

3.2.2.7.1. ProviderCancel Commands

Put all ECHONET Lite providers on CaoEngine in cancellation status.

When the provider is in the canceled state, any access to Value property of the variable waiting to be returned is returned with an error.

Also, accessing Value property of a variable for a provider that is in canceled state returns immediately with an error.

3.2.2.7.2. ProviderCancelClear Commands

Cancels the cancel state of the provider and returns to normal operation.

3.2.2.7.3. GetDeviceList Commands

Obtains a list of connectable ECHONET Lite nodes and ECHONET Lite devices.

Immediately after AddController is executed, all ECHONET Lite nodes and ECHONET Lite devices may not be acquired because ECHONET Lite nodes and ECHONET Lite devices are collected asynchronously when AddController is executed.

If all ECHONET Lite nodes and ECHONET Lite devices need to be acquired, wait for a while after executing AddController.

The following are the arguments and return values:

Item	Type	Description
Argument	Specify an integer equal to or greater than 0 for the wait time (ms) between command execution and retrieval of the value. If VT_EMPTY or an empty string is passed, 0 is specified.	
Return value	VT_ARRAY | VT_VARIANT (variable length) Each element is VT_ARRAY | VT_BSTR has the following meanings:	
	i VT_BSTR	The object code of the device (specified in ObjectCode option of AddController) is returned.
	VT_BSTR	Identification number of ECHONET Lite node. Returns (the value you specify for AddController's IdentificationNumber option.).
	VT_BSTR	Address of ECHONET Lite node. Returns (the value you specify for AddController's Address option.).
	VT_BSTR	A description of the device is returned.

Sample usage (C#)

```
// Execute GetDeviceList
Object[] deviceList = controller.Execute("GetDeviceList", null) as object[];
Foreach(string[] valueList in deviceList)
{
    String objectCode = valueList[0];
    String identificationNumber = valueList[1];
    String address = valueList[2];
    String description = valueList[3];
}
```

3.2.2.7.4. GetVariableList Commands

Obtains the system-variable list for ECHONET Lite node and ECHONET Lite device.

The following are the arguments and return values:

<table>
<tr><th>Item</th><th colspan="3">Type Description</th></tr>
<tr><td>Argument</td><td colspan="3">When an empty string is specified in VT_EMPTY or VT_BSTR, the system variable list of the executed controller is acquired.
When an object code is specified in VT_BSTR or VT_UI4, the list of system variables defined for ECHONET Lite node and ECHONET Lite device corresponding to the object code is obtained.
When an argument is specified with VT_BSTR, "0x" at the beginning is used as a hexadecimal number, and when it does not exist, it is used as a decimal number.</td></tr>
<tr><td rowspan="10">Return value</td><td colspan="3">VT_ARRAY | VT_VARIANT (variable length)
Each element is VT_ARRAY | VT_BSTR has the following meanings:</td></tr>
<tr><td>i</td><td>VT_BSTR</td><td>The name of the system variable is returned.</td></tr>
<tr><td></td><td>VT_BSTR</td><td>A description of the system variable is returned.</td></tr>
<tr><td></td><td>VT_BSTR</td><td>The property code of ECHONET Lite corresponding to the system variable is returned.</td></tr>
<tr><td></td><td>VT_BSTR</td><td>The attributes of the system variable are returned.
It can be "ReadWrite" (read-write), "ReadOnly" (read-only), or "WriteOnly" (write-only).</td></tr>
<tr><td></td><td>VT_BSTR</td><td>Returns the type of the value of the system variable.</td></tr>
<tr><td></td><td>VT_BSTR</td><td>If the system variable is of type BOOL, True is returned.
If the type is not BOOL, an empty string is returned.</td></tr>
<tr><td></td><td>VT_BSTR</td><td>If the system variable is of type BOOL, False is returned.
If the type is not BOOL, an empty string is returned.</td></tr>
<tr><td></td><td>VT_BSTR</td><td>The number of decimal places of the system variable is returned.
For system variables that do not have a decimal point, an empty string is returned.</td></tr>
</table>

Sample usage (C#)

```
// Execute GetVariableList
Object[] variableList = controller.Execute("GetVariableList", "0x027D01") as object[];
Foreach(string[] valueList in variableList)
{
    String variableName = valueList[0];
    String description = valueList[1];
    String propertyCode = valueList[2];
```

```
        String attribute = valueList[3];
        String valueType = valueList[4];
        String trueValue = valueList[5];
        String falseValue = valueList[6];
        String digit = valueList[7];
    }
```

3.2.2.7.5. RefreshDeviceList Commands

Update the list of connectable ECHONET Lite nodes and ECHONET Lite devices.

It may take some time to complete the updating of the list because the collection of ECHONET Lite nodes and ECHONET Lite devices is performed asynchronously after the execution.

The following are the arguments and return values:

Item	Type Description
Argument	Not used.
Return value	VT_EMPTY

Sample usage (C#)

```
// Execute GetDeviceList
Controller.Execute("RefreshDeviceList", null);
```

3.2.2.7.6. GetDefinedClassList Commands

Retrieves the list of ECHONET Lite equipment in which the class-definition file exists.

The following are the arguments and return values:

Item	Type Description		
Argument	Not used.		
Return value	VT_ARRAY | VT_VARIANT (variable length) Each element is VT_ARRAY | VT_VARIANT has the following meanings:		
	i	VT_UI2	The class code is returned. To obtain a list of the system variables that the class has, specify GetVariableList argument as an 8-bit left-shift (256 times) class code.
		VT_BSTR	The class name is returned.

3.2.2.8. OnMessage event

You can receive controller error notifications and status changes as OnMessage events. See 3.4 for the events that can be received.

3.2.3. CaoVariable classes

3.2.3.1. Index Properties

Gets the variable number as a Long type (4-byte integer type). This number is used to retrieve/delete the corresponding variable from the collection of variables in the parent object.

Index property is not changed when an element of a variable collection is added or deleted.

Sample usage (C#)

```
// Get Index
Int index = caoVar.Index;
```

3.2.3.2. Name Properties

Retrieves the variable name specified by AddVariable method of CaoController class.

Sample usage (C#)

```
System.Diagnostics.Debug.WriteLine(caoVar.Name);
```

3.2.3.3. Help Properties

Gets the help string for a variable.

ECHONET Lite provider can obtain an explanation of the properties of ECHONET Lite nodes or ECHONET Lite equipment assigned to the system variables.

Sample usage (C#)

```
// Get Help
String help = caoVar.Help;
```

3.2.3.4. Value Properties

Obtain/set the property value from the connected ECHONET Lite node or ECHONET Lite device. The behavior depends on the variable name. For details, refer to section 3.3, Variable List.

3.3. Variable list

Defines a list of variables that can be used in each class. Variables refer to objects of CaoVariable classes.

3.3.1. System and User Variables

There are two types of variables in the provider: system variables and user variables.

System Variables

A variable that accesses only the information in the object that holds the variable. System variables are often static data. System variables are preceded by "@".

e.g. provider version, device manufacturer, serial number

User variable

A variable that you can use to specify what information you want to access when you create a variable by using Option character string. User variables can be given any name. (However, names that start with "@" are recognized as system variables and cannot be used.)

3.3.2. CaoController class system variable

Variable Name	Description	Value		Reference
		Get	Put	
@MAKER_NAME	Obtain the manufacturer's name.	✓	-	P.18
@VERSION	Get the DLL version.	✓	-	P.18
Other	Obtains/sets the value from the system variable defined for ECHONET Lite node or ECHONET Lite device.	✓	✓	P.19

3.3.2.1. @MAKER_NAME

Obtain the manufacturer's name.

Data type

Type Description	
VT_BSTR	Obtain the manufacturer's name.

Sample usage (C#)

```
// Add Variable
ORiN2.ManagedCAO.CCaoVariable var = controller.AddVariable("@MAKER_NAME","");
// Acquisition of Values
String value = var.Value as string;
```

3.3.2.2. @VERSION

Gets the DLL version.

Data type

Type Description	
VT_BSTR	Get the DLL version. *.*.*

Sample usage (C#)

```
// Add Variable
ORiN2.ManagedCAO.CCaoVariable var = controller.AddVariable("@VERSION","");
```

```
// Acquisition of Values
String value = var.Value as string;
```

3.3.2.3. Other System Variables

System variable defined for ECHONET Lite node or ECHONET Lite device.

Use the controller's GetVariableList extension method to obtain the system variables that can be used.

3.4. Event list

When ECHONET Lite node or ECHONET Lite device notifies you to update the value of the property, OnMessage event notifies you to update the value.

The message notified by the event looks like this:

Properties	Description
Number	The property code of the property for which the change was notified.
Value	The changed value. For a property for which a system variable is defined, the value is converted to the type of the system variable and notified. Otherwise, the raw data of the property is VT_ARRAY | It is notified by VT_UI1.

e.g.) If the battery operation mode (property code 0xDA(218)) is switched to quick charge (65), a message with a Number of 218 and a Value of 65 is notified.

3.5. Type that can be used with ECHONET Lite

The types available for ECHONET Lite providers are:

Table 3-2 Possible values for ValueType

Type to specify	Length of the data	Types in ORiN	Digit enabled	Remarks
BOOL	1	VT_UI1	×	If you do not specify a TrueValue and FalseValue, you will not be able to acquire or set it.
UI1	1 * n	VT_UI1	✓	When n = 1, it is handled as a single value, and when n > 1, it is handled as an array.
I1	1 * n	VT_I1	✓	When n = 1, it is handled as a single value, and when n >

				1, it is handled as an array.
UI2	2 * n	VT_UI2	✓	When n = 1, it is handled as a single value, and when n > 1, it is handled as an array.
I2	2 * n	VT_I2	✓	When n = 1, it is handled as a single value, and when n > 1, it is handled as an array.
UI4	4 * n	VT_UI4	✓	When n = 1, it is handled as a single value, and when n > 1, it is handled as an array.
I4	4 * n	VT_I4	✓	When n = 1, it is handled as a single value, and when n > 1, it is handled as an array.
UI8	8 * n	VT_UI8	✓	When n = 1, it is handled as a single value, and when n > 1, it is handled as an array.
I8	8 * n	VT_I8	✓	When n = 1, it is handled as a single value, and when n > 1, it is handled as an array.
BSTR	Optional	VT_BSTR	×	Encodes in ASCII.
Date	4	VT_DATE	×	First 2 bytes: Year, 3rd byte: Month, 4th byte: Day
Time	2	VT_BSTR	×	1st byte: hour, 2nd byte: minute. Processed as a character string in HH:mm format.
Elappsed	5	VT_BSTR	×	Elapsed time. Leading 1 byte: Unit (seconds: 0x41, minutes: 0x42, hours: 0x43, days: 0x44) followed by 4 bytes in uint. Get as a string in the form {value}{units}. ("1234s" etc.). Read-only.
ObjectCode	3	VT_UI4	×	Represents the code of an object.
PropertyMap	17	VT_UI1 | VT_ARRAY	×	Special data structure for property maps. Read-only.

4. ECHONET Lite Programming by provider

ECHONET Lite providers can connect client PCs to ECHONET Lite nodes or ECHONET Lite devices by doing the following:

- Creating a CaoEngine
- Creating a CaoWorkspace
- Creating a CaoController

After connecting to a ECHONET Lite node or ECHONET Lite device, the information of CaoVariable node or device can be accessed by generating the object.

4.1. Sample programming that connects to the battery to obtain operational status

This section shows an example of a sample program that acquires the operating status of the battery and stops it if it is in operation. Table 4-1 describes the requirements of the sample program, and Figure 4-1 describes the flow of the sample program.

Table 4-1 Sample program requirements

Requirements	Description
Host	Connect via UDP
	The destination IP address is 192.168.1.2.
Process Description	Reads the operating status of the battery.
	Stop the battery if it is running

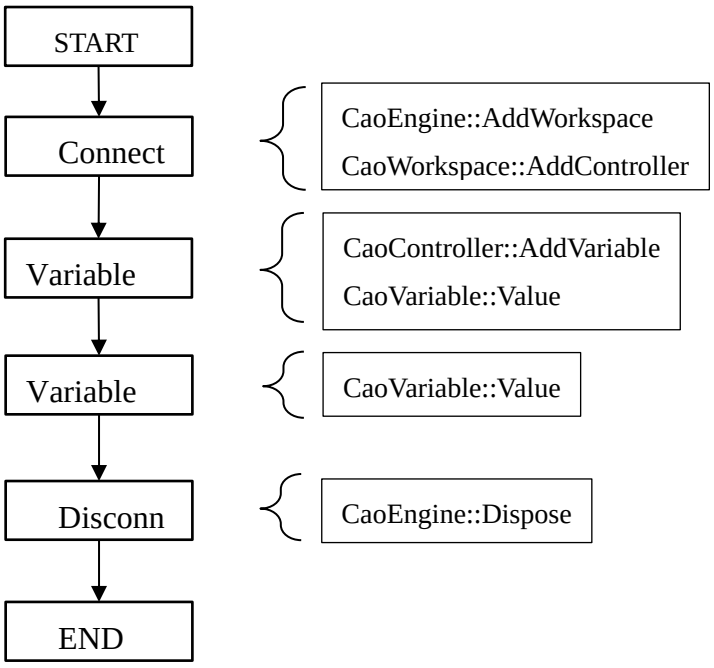

Fig. 4-1 Flowchart of Programs Connecting to a Storage Battery to Acquire the Operating Fig. 4-2 Status

Specific codes are given in the following sections.

4.1.1. Sample program

The following is an overview of the sample program.

Sample **Sample.cs**

```
// Object
Private ORiN2.ManagedCAO.CCaoEngine m_caoEngine = null;
Private ORiN2.ManagedCAO.CCaoWorkspace m_caoWorkspace = null;
Private ORiN2.ManagedCAO.CCaoController m_caoController = null;
Private ORiN2.ManagedCAO.CCaoVariable m_varOperationStatus = null;

Public void Main()
{
    Try
    {
      // Connection
      This.Connect();
      // Retrieving Values
      Bool operationStatus = Convert.ToBoolean(m_varOperationStatus.Value);
      // Set the operation status to OFF.
      If (operationStatus)
      {
        m_varOperationStatus.Value = false;
      }
    }
    Finally
    {
      // Disconnect
      This.Disconnect();
    }
}

// Connection method
Private void Connect()
{
```

```
        // Generate CaoEngine object
        This.m_caoEngine = new ORiN2.ManagedCAO.CCaoEngine();
        // Generate CaoWorkspace object
        This.m_caoWorkspace = this.m_caoEngine.AddWorkspace("NewWrks", "");
        // Generate CaoController object
        This.m_caoController = this.m_caoWorkspace.AddController(
            "ECHONET", "CaoProv.ECHONETLite", "",
            "ObjectCode=0x027D01,Address=192.168.1.2");
        // Generate CaoVariable object
        This.m_varIoeratuibStatus = m_caoController.AddVariable("@OperationStatus", "");
    }

    // Disconnect method
    Private void Disconnect()
    {
        If (this.m_caoEngine != null)
        {
          This.m_caoEngine.Dispose();
        }
        This.m_caoEngine = null;
    }
```

4.1.1.1. Connection

To connect to ECHONET Lite nodes or ECHONET Lite devices, do the following:

(1) Prepare a variable to hold the object. The objects required to connect to the controller are CaoEngine object, CaoWorkspace object, and CaoController object. CaoWorkpace object does not need to have a variable to obtain CaoController object from CaoWorkspaces. You will also need a CaoVariable for accessing the variable. The following is a code example for C#.

Sample usage (C#)

```
    // Variables for CaoEngine
    Private ORiN2.ManagedCAO.CCaoEngine m_caoEngine = null;
    // Variables for CaoWorkspace
    Private ORiN2.ManagedCAO.CCaoWorkspace m_caoWorkspace = null;
    // Variables for CaoController
    Private ORiN2.ManagedCAO.CCaoController m_caoController = null;
```

```
// Variables for CaoVariable
Private ORiN2.ManagedCAO.CCaoVariable m_ varOperationStatus = null;
```

(2) Creates a CaoEngine object. CaoEngine object is generated using the new keyword.

Sample usage (C#)

```
// Generate CaoEngine object
This.m_caoEngine = new ORiN2.ManagedCAO.CCaoEngine();
```

(3) Gets or generates a CaoWorkspace object. When you create a CaoEngine object, it defaults to one CaoWorkspaces object and one object. The following is a sample code/default CaoWorkspace for creating a new CaoWorkspace.

Sample usage (C#)

```
// Generate CaoWorkspace object
This.m_caoWorkspace = this.m_caoEngine.AddWorkspace("NewWrks", "");
```

(4) Create a CaoController object. To generate a CaoController object, set the provider name to use and the parameters to use. For ECHONET Lite providers, optionally specify the object code and the address or identification number of the node. The following is a code example:

Sample usage (C#)

```
// Generate CaoController object
This.m_caoController = this.m_caoWorkspace.AddController(
        "ECHONET", "CaoProv.ECHONETLite", "",
        "ObjectCode=0x027D01,Address=192.168.1.2");
```

(5) Create a CaoVariable object. To create a CaoVariable object, call AddVariable method on the controller. The following is a code example:

Sample usage (C#)

```
    // Generate CaoVariable object
    This.m_varIoeratuibStatus = m_caoController.AddVariable("@OperationStatus", "");
```

4.1.1.2. Obtaining the Operating Status

Acquires the value of the operation status property by acquiring the value of CaoVariable.

Sample usage (C#)

```
    // Retrieving Values
    Bool operationStatus = Convert.ToBoolean(m_varOperationStatus.Value);
```

4.1.1.3. Setting the Operating State

Write to the property by setting the value to CaoVariable.

Sample usage (C#)

```
// Set the operation status to OFF.
If (operationStatus)
{
  m_varOperationStatus.Value = false;
}
```

4.1.1.4. Disconnect

To disconnect from the controller, you can erase the generated objects and delete the objects that you want to erase from the collection class that manages the objects. However, you do not need to explicitly remove it if you use ORiN.ManagedCAO. The following is a code example:

Sample usage (C#)

```
// Remove all objects from CaoEngine
This.m_caoEngine.Dispose();
// Clear CaoEngine
This.m_caoEngine = null;
```

Appendix A. **Class definition file**

ECHONET Lite providers can add definitions for instrument objects by placing classes of instrument objects as JSON format definition files.

Place the device object definition file in the following location with the following names:

- Deployment folder: %ECHONET Lite provider Bin folder%¥ setting
- File name: {Class group code (2 hexadecimal digits without 0x)}{Class code (2 hexadecimal digits without 0x)}. json

The contents of the class definition file are as follows.

Appendix A.1. **Class Object Structure**

Example)

```
{
  "classCode": "0x027D",
  "description": {
    "value": "Storage battery",
    "localization": [
      {"language": "ja", "value": "蓄電池"}
    ]
  },
  "properties" : [...]
}
```

Property Name	Value Type	Description
ClassCode	String	Specify the type code of the model as "0x" + 4 hexadecimal digits (the same character string as the file name).
Description	Description Object	See the structural (Link) of the description object.
Properties	Array of property objects	See the Structural (Link) for the Property Object.

Appendix A.2. **Property Object Structure**

Example 1)

```
{
  "propertyCode": "0x85",
  "description": {
    "value": "Measured cumulative power consumption",
    "localization": [
      {"language": "ja", "value": "積算消費電力計測値"}
    ]
  },
  "variableName": "@MeasuredCumulativePowerConsumption",
  "valueType": "UI4",
  "attribute": "ReadOnly",
  "digit": 3
}
```

Example 2)

```
{
  "propertyCode": "0x80",
  "description": {
    "value": "Operation status",
    "localization": [
      {"language": "ja", "value": "動作状態"}
    ]
  },
  "variableName": "@OperationStatus",
  "valueType": "BOOL",
  "attribute": "ReadWrite",
  "trueValue": "0x30",
  "falseValue": "0x31"
}
```

Property Name	Value Type	Description
PropertyCode	String	Specify the code of the property as "0x" + two hexadecimal digits.
Description	Description Object	See the structural (Link) of the description

		object.
VariableName	String	Specify the system variable name of the controller.
ValueType	String	Refer to the type (Link) that can be used with ECHONET Lite.
Attribute	String	Specify the attribute of the variable as "ReadWrite" (read-write), "ReadOnly" (read-only), or "WriteOnly" (write-only). If the property attribute can be obtained when connecting to a ECHONET Lite device, it will be overwritten with the attribute obtained from ECHONET Lite device.
TrueValue,falseValue	String	See TrueValue Options, FalseValue Options (Link) in Controller.AddVariable.
Digit	Number	See Digit optional (Link) in Controller.AddVariable.

Appendix A.3. **Description Object Structure**

Example)

```
{
  "value": "Operation status",
  "localization": [
    {"language": "ja", "value": "動作状態"},
    {"language": "zh-cn", "value": "运行状态"}
  ]
}
```

Property Name	Value Type	Description
Value	String	Set the explanation when the language corresponding to localization property does not exist.
Localization	Array of Language Resource Objects	See Structural (Link) for Language Resource Objects.

Appendix A.4. **Structure of Language Resource Objects**

Property Name	Value Type	Description
Language	String	Specify the locale of the language of the resource.
Value	String	Set the description for the locale specified by languate property.