ECHONET Lite プロバイダ
ユーザーズ ガイド

Version 1.0.0

March 4, 2021

備考:

【改版履歴】

バージョン	日付	内容
1.0.0	2021-03-04	初版.

【対応機種】

機種	バージョン	注意事項
燃料電池クラス	ECHONET 機器オブジェクト詳細規定 Release N	
蓄電池クラス	ECHONET 機器オブジェクト詳細規定 Release N	
瞬間式給湯器クラス	ECHONET 機器オブジェクト詳細規定 Release N	

【動作確認機種】

機種	バージョン	注意事項

目次

1. はじめに 6

2. アプリケーション開発のための環境セットアップ 7

2.1. ECHONET Lite とクライアント PC との接続 7
2.2. PC 開発環境のセットアップ 7
2.2.1. ECHONET Lite プロバイダの手動インストール 7

3. コマンドリファレンス 8

3.1. メソッド/プロパティー覧 8
3.2. メソッド・プロパティ 8
3.2.1. CaoWorkspace クラス 8
3.2.1.1. AddController メソッド 8
3.2.2. CaoController クラス 10
3.2.2.1. Index プロパティ 10
3.2.2.2. Name プロパティ 11
3.2.2.3. Help プロパティ 11
3.2.2.4. GetVariableNames メソッド 11
3.2.2.5. Variables プロパティ 11
3.2.2.6. AddVariable メソッド 11
3.2.2.7. Execute メソッド 13
3.2.2.8. OnMessage イベント 16
3.2.3. CaoVariable クラス 17
3.2.3.1. Index プロパティ 17
3.2.3.2. Name プロパティ 17
3.2.3.3. Help プロパティ 17
3.2.3.4. Value プロパティ 17
3.3. 変数一覧 17
3.3.1. システム変数とユーザー変数 17
3.3.2. CaoController クラスシステム変数 18
3.3.2.1. @MAKER_NAME 18
3.3.2.2. @VERSION 18
3.3.2.3. その他のシステム変数 19

3.4. イベント一覧 19
3.5. ECHONET Lite で使用できる型 19
4. ECHONET Lite プロバイダによるプログラミング 21
4.1. 蓄電池に接続して動作状態を取得するサンプルプログラミング 21
4.1.1. サンプルプログラム 22
4.1.1.1. 接続 23
4.1.1.2. 動作状態の取得 24
4.1.1.3. 動作状態の設定 25
4.1.1.4. 切断 25
付録 A. クラス定義ファイル 26

1. はじめに

本書は，ECHONET Lite 対応デバイスに対して通信をするプロバイダのユーザーズガイドです．図 1-1 が本プロバイダとデバイスの全体構成図になります．以降本プロバイダを ECHONET Lite プロバイダと呼称します．

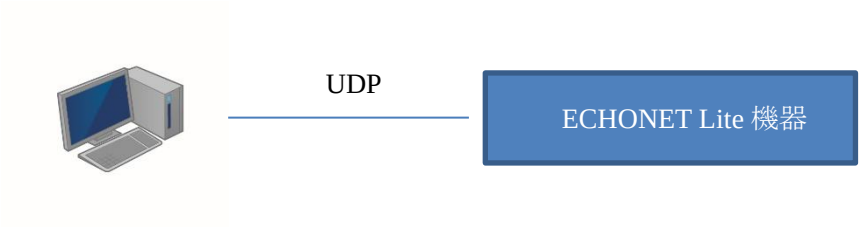

図 1-1 構成図

また，本プロバイダ及びデバイスそれぞれの対応を図 1-2 に表します．
(※一例です．全てを表しているわけではありません．)

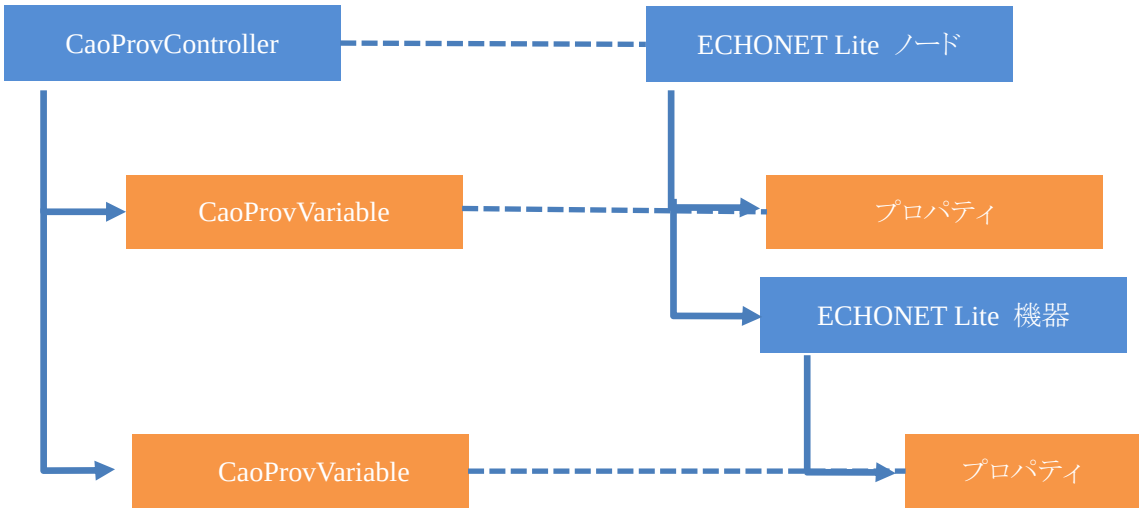

図 1-2 プロバイダの構成とデバイス情報との対応図

2. アプリケーション開発のための環境セットアップ

2.1. ECHONET Lite とクライアント PC との接続

ECHONET Lite プロバイダは UDP 通信によって ECHONET Lite と接続します.

ECHONET Lite プロバイダは送受信に UDP の 3610 番のポートを使用するため, あらかじめファイアウォールの設定で UDP の 3610 番のポートへの着信を許可しておく必要があります.

2.2. PC 開発環境のセットアップ

2.2.1. ECHONET Lite プロバイダの手動インストール

ECHONET Lite プロバイダを手動でインストールする場合は下記レジストリ登録を行う必要があります.

レジストリ登録を行う場合は, 管理者権限でコマンドプロンプトを起動し, ORiN2 をインストールしたフォルダの下の DotNet¥Bat¥RegistAsm.bat をプロバイダの DLL を引数に実行してください.

レジストリの登録を解除する場合は, 管理者権限でコマンドプロンプトを起動し, ORiN2 をインストールしたフォルダの下の DotNet¥Bat¥UnregistAsm.bat をプロバイダの DLL を引数に実行してください.

実行する際には, ファイルのあるパスまで移動するか, ファイルパスを指定して実行してください.

表 2-1 ECHONET Lite プロバイダ

ファイル名	CaoProvECHONETLite.dll
ProgID	CaoProv.ECHONETLite
レジストリ登録	RegistAsm.bat CaoProvECHONETLite.dll
レジストリ登録の抹消	UnregistAsm.bat CaoProvECHONETLite.dll

3. コマンドリファレンス

3.1. メソッド/プロパティー覧

表 3-1 メソッド/プロパティー覧

カテゴリ	メソッド/プロパティ[1]		機能	参照
CaoWorkspace				
	AddController	M	コントローラに接続	P.8
CaoController				
	Index	P	コントローラ番号の取得	P.10
	Name	P	コントローラ名の取得	P.11
	Help	P	ヘルプの取得	P.11
	GetVariableNames	M	接続可能な変数名リストの取得	P.11
	Variables	P	コントローラが保持する変数コレクションの取得	P.11
	AddVariable	M	変数オブジェクトの追加	P.11
	Execute	M	拡張コマンドの実行	P.13
	OnMessage	E	メッセージ受信イベント	P.16
CaoVariable				
	Index	P	変数番号の取得	P.17
	Name	P	変数名の取得	P.17
	Help	P	ヘルプの取得	P.17
	Value	P	値の取得/設定	P.17

3.2. メソッド・プロパティ

3.2.1. CaoWorkspace クラス

3.2.1.1. AddController メソッド

CaoWorkspace に，コントローラオブジェクトを追加します．ECHONET Lite プロバイダでは，AddController メソッド実行時に渡されたパラメータを参照し，該当する ECHONET Lite ノードに接続されている ECHONET Lite 機器と接続を行います．以下に，AddController メソッドの仕様を示します．

書式

```
AddController
(
"<コントローラ名>",                    // コントローラ名(任意)
```

[1] M:メソッド, P:プロパティ, E:イベントをそれぞれ示します．

```
"CaoProv.ECHONETLite",      // プロバイダ名（固定）
"<マシン名>",               // プロバイダ実行マシン名(未使用)
"<オプション>"              // オプション文字列(省略可能)
)
```

オプション

以下にオプション文字列に指定するオプションを示します．オプション文字列は下記に示す各オプションをカンマ(,)でつなげた文字列となります．

オプション	必須	説明	値範囲	デフォルト値
ObjectCode		下記の ObjectCode オプションを参照してください．	0～16777215	-
Address	※1	下記の Address オプションを参照してください．		-
IdentificationNumber	※1	下記の IdentificationNumber オプションを参照してください．		-
Timeout		下記の Timeout オプションを参照してください．	1～65535	1000
MyIP		下記の MyIP オプションを参照してください．		0.0.0.0

※1 ObjectCode を指定した場合は Address か IdentificationNumber のいずれかが必須です．

使用例(C#)

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

3.2.1.1.1. ObjectCode オプション

接続先の ECHONET Lite ノードまたは ECHONET Lite 機器のオブジェクトコードを指定してください．

文字列の先頭に”0x”が存在した場合は 16 進数で，存在しない場合は 10 進数で指定されたものとして扱います．

省略時は拡張メソッドの GetDeviceList および GetVariableList のみ使用できるコントローラオブジェクトが追

加されます.

3.2.1.1.2. Address オプション

接続先の ECHONET Lite ノードの IP アドレスを指定します.

ObjectCode を指定した場合は Address または IdentificationNumber のいずれかを指定しなければなりません.

Address と IdentificationNumber が同時に指定された場合は Address が優先されます.

3.2.1.1.3. IdentificationNumber オプション

接続先の ECHONET Lite ノードの識別番号(EPC:0x83)を 16 進数 17 桁(34 文字)で指定します.

IdentificationNumber は文字列の先頭の"0x"の存在の有無にかかわらず 16 進数で指定されたものとして扱います.

IdentificationNumber を指定した場合は ECHONET Lite ノードの識別番号をもとに ECHONET Lite ノードのアドレスを取得し, 取得したアドレスをもとに通信を行います.

DHCP の環境で ECHONET Lite を運用している場合はアドレスが変化する可能性がありますので, IdentificationNumber を使用するようにしてください.

Address と IdentificationNumber が同時に指定された場合は Address が優先されます.

3.2.1.1.4. Timeout オプション

ECHONET Lite ノードに対して要求を送信した後に返信を待機する時間を ms 単位で指定します.

GetDeviceList を実行した際は要求送信後に Timeout だけ待機し, その間に応答を返した ECHONET Lite ノードの情報を取得します.

3.2.1.1.5. MyIP オプション

複数のネットワークアダプタが搭載されている PC で送受信に使用するアドレスを指定します.

複数のネットワークアダプタが存在する場合は想定外のネットワークアダプタが通信に使用されることがあるため, 複数のネットワークアダプタが搭載されている場合は必ず指定するようにしてください.

3.2.2. CaoController クラス

3.2.2.1. Index プロパティ

コントローラ番号を Long 型(4 バイト整数型)で取得します. この番号は, CaoWorkspace クラスの保持するコントローラコレクションから該当のコントローラを取得/削除するために使用されます.

Index プロパティはコントローラコレクションの要素が追加/削除されても変更されることはありません.

使用例(C#)

```
// Index 取得
long index = controller.Index;
```

3.2.2.2. Name プロパティ

CaoWorkspace クラスの AddController メソッドで指定されたコントローラ名を取得します.

使用例(C#)

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

3.2.2.3. Help プロパティ

ObjectCode オプションで指定した ECHONET 機器の名称を取得します.

使用例(C#)

```
// Help 取得
string help = controller.Help;
```

3.2.2.4. GetVariableNames メソッド

接続可能なシステム変数の変数名のリストを取得します.

書式

```
GetVariableNames
(
"<オプション>"                    // オプション文字列
)
```

オプション

ECHONET Lite プロバイダではオプション文字列は使用しません.
指定されたオプション文字列は無視されます.

使用例(C#)

```
// 変数名リスト取得
string[] variableNames = controller.GetVariableNames("");
```

3.2.2.5. Variables プロパティ

コントローラが保持する, 変数コレクションを取得します.

使用例(C#)

```
// 変数コレクション取得
ORiN2.ManagedCAO.CCaoVariables variables = controller.Variables;
// 変数取得
ORiN2.ManagedCAO.CCaoVariable variable = variables[0];
```

3.2.2.6. AddVariable メソッド

CaoController に変数オブジェクトを追加します. 変数名には 3.3.2 に示すシステム変数と, 先頭に"@"を付

けないユーザー変数が使用できます.

以下に, AddVariable の仕様を示します.

ユーザー変数はオプションを使用して取得するプロパティを指定します.

オプション

以下にオプション文字列に指定するオプションを示します. オプション文字列は下記に示す各オプションをカンマ(,)でつなげた文字列となります.

オプション	必須	説明	値範囲	デフォルト値
PropertyCode	○	取得/設定するプロパティのコード	0～255	-
ValueType		値の型		UI1
TrueValue	※2	値の型が BOOL のときに True として扱う値	0～255	-
FalseValue	※2	値の型が BOOL のときに False として扱う値	0～255	
Digit		小数点以下の桁数	0～255	

※2 ValueType で BOOL を指定した場合は必須です.

書式

```
AddVariable
(
"<変数名>",                    // 変数名
"<オプション>"                 // オプション文字列(省略可能)
)
```

3.2.2.6.1. PropertyCode オプション

取得/設定する ECHONET Lite ノードまたは ECHONET Lite 機器のプロパティのコードを指定してください.

文字列の先頭に”0x”が存在した場合は 16 進数で, 存在しない場合は 10 進数で指定されたものとして扱います.

3.2.2.6.2. ValueType オプション

取得/設定する ECHONET Lite ノードまたは ECHONET Lite 機器のプロパティの値の型を指定してください.

詳細は ECHONET Lite で使用できる型を参照してください.

3.2.2.6.3. TrueValue オプション, FalseValue オプション

ValueType で BOOL を指定したときに True および False として扱う値を指定します.

ValueType で BOOL を指定したときは必須です.

文字列の先頭に”0x”が存在した場合は 16 進数で, 存在しない場合は 10 進数で指定されたものとして扱います.

3.2.2.6.4. Digit オプション

ECHONET Lite で使用できる型で Digit 有効に〇がついている型は Digit で小数点以下の桁数を指定できます.(例:単位が 0.001V の場合は Digit に 3 を指定します.)

Digit を指定した場合は VT_R8 として取り扱います.

3.2.2.7. Execute メソッド

CaoController の拡張コマンドを実行します. 以下に, Execute の仕様を示します.

書式

```
Execute
(
"<拡張コマンド名>",          // 拡張コマンド名
"<オプション文字列>"         // オプション文字列(省略可能)
)
```

以下に, Execute で指定できる拡張コマンド一覧を示します. 使用例は拡張コマンドの詳細で記述しています.

コマンド	説明	参照
ProviderCancel	プロバイダをキャンセル状態にします.	P.13
ProviderClear	プロバイダのキャンセル状態を解除し, 通常の動作に戻します.	P.14
GetDeviceList	接続可能な ECHONET Lite ノードおよび ECHONET Lite 機器の一覧を取得します.	P.14
GetVariableList	追加可能なシステム変数の情報の一覧を取得します.	P.15
RefreshDeviceList	接続可能な ECHONET Lite ノードおよび ECHONET Lite 機器の一覧の更新を実行します.	P.16

3.2.2.7.1. ProviderCancel コマンド

CaoEngine 上にあるすべての ECHONET Lite プロバイダをキャンセル状態にします.

プロバイダがキャンセル状態になるとき, 返信を待機している変数の Value プロパティへのアクセスはすべてエラーで戻ります.

また, キャンセル状態のプロバイダに対して変数の Value プロパティにアクセスすると即座にエラーで戻りま

す.

3.2.2.7.2. ProviderCancelClear コマンド

プロバイダのキャンセル状態を解除し，通常の動作に戻します.

3.2.2.7.3. GetDeviceList コマンド

接続可能な ECHONET Lite ノード及び ECHONET Lite 機器の一覧を取得します.

AddController を実行したときに非同期で ECHONET Lite ノード及び ECHONET Lite 機器の収集を非同期で行うため，AddController の実行直後はすべての ECHONET Lite ノード及び ECHONET Lite 機器が取得できないことがあります.

すべての ECHONET Lite ノード及び ECHONET Lite 機器の取得が必要な場合は AddController を実行した後にしばらく待機を行うようにしてください.

以下に引数と戻り値を示します.

項目	型説明		
引数	コマンドを実行してから値を取得するまでの待機時間（ms）を 0 以上の整数で指定してください. VT_EMPTY または空文字列を渡した場合は 0 を指定したことになります.		
戻り値	VT_ARRAY | VT_VARIANT（可変長） 各要素は VT_ARRAY | VT_BSTR で以下の意味を持ちます.		
	i	VT_BSTR	デバイスのオブジェクトコード（AddController の ObjectCode オプションに指定する値）が返却されます.
		VT_BSTR	ECHONET Lite ノードの識別番号.（AddController の IdentificationNumber オプションに指定する値）が返却されます.
		VT_BSTR	ECHONET Lite ノードのアドレス.（AddController の Address オプションに指定する値）が返却されます.
		VT_BSTR	デバイスの説明が返却されます.

使用例(C#)

```
// GetDeviceList 実行
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 コマンド

ECHONET Lite ノード及び ECHONET Lite 機器のシステム変数一覧を取得します.

以下に引数と戻り値を示します.

項目	型説明		
引数	VT_EMPTY または VT_BSTR で空文字列を指定した場合は実行したコントローラのシステム変数一覧を取得します. VT_BSTR または VT_UI4 でオブジェクトコードを指定した場合はオブジェクトコードに該当する ECHONET Lite ノード及び ECHONET Lite 機器に定義されているシステム変数の一覧を取得します. VT_BSTR で引数を指定する場合は先頭に"0x"が存在した場合は 16 進数で, 存在しない場合は 10 進数で指定されたものとして扱います.		
戻り値	VT_ARRAY | VT_VARIANT（可変長） 各要素は VT_ARRAY | VT_BSTR で以下の意味を持ちます.		
	i	VT_BSTR	システム変数の名前が返却されます.
		VT_BSTR	システム変数の説明が返却されます.
		VT_BSTR	システム変数に対応する ECHONET Lite のプロパティコードが返却されます.
		VT_BSTR	システム変数の属性が返却されます. 値は"ReadWrite"（読み取り書き込み可能）, "ReadOnly"（読み取り専用）, "WriteOnly"（書き込み専用）のいずれかです.
		VT_BSTR	システム変数の値の型が返却されます.
		VT_BSTR	システム変数が BOOL 型の時は True として扱う値が返却されます. BOOL 型以外の場合は空文字列が返却されます.
		VT_BSTR	システム変数が BOOL 型の時は False として扱う値が返却されます. BOOL 型以外の場合は空文字列が返却されます.
		VT_BSTR	システム変数の小数点以下の桁数が返却されます. 小数点以下を持たないシステム変数の場合は空文字列が返却されます.
		VT_BSTR	システム変数の括弧などが使用されていない英語の説明が返却されます.

使用例(C#)

```
// GetVariableList 実行
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 コマンド

接続可能な ECHONET Lite ノード及び ECHONET Lite 機器の一覧を更新します.

実行後に ECHONET Lite ノード及び ECHONET Lite 機器の収集を非同期で行うため，一覧の更新が完了するのに多少時間がかかることがあります.

以下に引数と戻り値を示します.

項目	型説明
引数	使用しません.
戻り値	VT_EMPTY

使用例(C#)

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

3.2.2.7.6. GetDefinedClassList コマンド

クラス定義ファイルが存在する ECHONET Lite 機器の一覧を取得します.

以下に引数と戻り値を示します.

<table>
<tr><td>項目</td><td colspan="3">型説明</td></tr>
<tr><td>引数</td><td colspan="3">使用しません.</td></tr>
<tr><td rowspan="3">戻り値</td><td colspan="3">VT_ARRAY | VT_VARIANT（可変長）
各要素は VT_ARRAY | VT_VARIANT で以下の意味を持ちます.</td></tr>
<tr><td>i</td><td>VT_UI2</td><td>クラスコードが返却されます.
クラスの持つシステム変数の一覧を取得するには GetVariableList の引数にクラスコードを 8 ビット左シフト（256 倍）したものを指定してください.</td></tr>
<tr><td></td><td>VT_BSTR</td><td>クラス名が返却されます.</td></tr>
</table>

3.2.2.8. OnMessage イベント

コントローラのエラー通知や状態の変化を OnMessage イベントとして受け取ることが可能です. 受け取れるイベントについては 3.4 を参照してください.

3.2.3. CaoVariable クラス

3.2.3.1. Index プロパティ

変数番号を Long 型(4 バイト整数型)で取得します. この番号は親のオブジェクトの変数のコレクションから該当の変数を取得/削除するために使用されます.

Index プロパティは変数コレクションの要素が追加/削除されても変更されることはありません.

使用例(C#)

```
// Index 取得
int index = caoVar.Index;
```

3.2.3.2. Name プロパティ

CaoController クラスの AddVariable メソッドで指定された変数名を取得します.

使用例(C#)

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

3.2.3.3. Help プロパティ

変数のヘルプ文字列を取得します.

ECHONET Lite プロバイダではシステム変数に割り当てられている ECHONETLite ノードまたは ECHONET Lite 機器のプロパティの説明が取得できます.

使用例(C#)

```
// Help 取得
string help = caoVar.Help;
```

3.2.3.4. Value プロパティ

接続した ECHONET Lite ノードまたは ECHONETLite 機器からプロパティの値を取得/設定します. 変数名によって動作が異なります. 詳細は, 3.3.変数一覧を参照してください.

3.3. 変数一覧

各クラスで使用可能な変数一覧を定義します. なお変数は, CaoVariable クラスのオブジェクトを指します.

3.3.1. システム変数とユーザー変数

プロバイダにはシステム変数とユーザー変数という 2 種類の変数が存在します.

システム変数

その変数を保持するオブジェクト内で唯一の情報にアクセスための変数です. システム変数はしばしば静的データである場合があります. システム変数は名前の先頭に"@"がついています.

例)プロバイダバージョン, デバイス製造元, シリアル番号

ユーザー変数

変数を作成する際にどのような情報にアクセスするのかを，オプション文字列を使用することでユーザーが指定できる変数です．ユーザー変数は任意の名前を付けることができます．（ただし，”@”で開始する名前はシステム変数として認識されるため使用できません．）

3.3.2. CaoController クラスシステム変数

変数名	説明	Value		参照
		get	put	
@MAKER_NAME	メーカー名を取得します．	○	-	P.18
@VERSION	DLL バージョンを取得します．	○	-	P.18
その他	ECHONET Lite ノードまたは ECHONET Lite 機器ごとに定義されているシステム変数から値を取得/設定します．	○	○	P.19

3.3.2.1. @MAKER_NAME

メーカー名の取得をします．

データ型

型説明	
VT_BSTR	メーカー名を取得します．

使用例(C#)

```
// 変数追加
ORiN2.ManagedCAO.CCaoVariable var = controller.AddVariable("@MAKER_NAME","");
// 値取得
string value = var.Value as string;
```

3.3.2.2. @VERSION

DLL のバージョンの取得をします．

データ型

型説明	
VT_BSTR	DLL のバージョンを取得します． *.*.*

使用例(C#)

```
// 変数追加
ORiN2.ManagedCAO.CCaoVariable var = controller.AddVariable("@VERSION","");
```

```
// 値取得
string value = var.Value as string;
```

3.3.2.3. その他のシステム変数

ECHONET Lite ノードまたは ECHONET Lite 機器ごとに定義されているシステム変数です.
使用できるシステム変数はコントローラの GetVariableList 拡張メソッドで取得してください.

3.4. イベント一覧

ECHONET Lite ノードまたは ECHONET Lite 機器からプロパティの値の更新が通知されたとき, OnMessage イベントで値の更新を通知します.
イベントで通知されるメッセージは次のようになります.

プロパティ	説明
Number	変更が通知されたプロパティのプロパティコード.
Value	変更後の値. システム変数が定義されているプロパティの場合はシステム変数の型に値を変換して通知します. それ以外の場合はプロパティの生データを VT_ARRAY | VT_UI1 で通知します.

例） 蓄電池の運転モード(プロパティコード 0xDA(218))が急速充電(65)に切り替わった場合, メッセージの Number が 218, Value が 65 のメッセージが通知されます.

3.5. ECHONET Lite で使用できる型

ECHONET Lite プロバイダで使用できる型は以下の通りです.

表 3-2 ValueType に指定可能な値

指定する型	データの長さ	ORiN での型	Digit 有効	備考
BOOL	1	VT_UI1	×	TrueValue と FalseValue を指定しないと値の取得/設定はできません.
UI1	1 * n	VT_UI1	○	n = 1 のときは単体の値として, n > 1 のときは配列として扱います.

I1	1 * n	VT_I1	○	n = 1 のときは単体の値として, n > 1 のときは配列として扱います.
UI2	2 * n	VT_UI2	○	n = 1 のときは単体の値として, n > 1 のときは配列として扱います.
I2	2 * n	VT_I2	○	n = 1 のときは単体の値として, n > 1 のときは配列として扱います.
UI4	4 * n	VT_UI4	○	n = 1 のときは単体の値として, n > 1 のときは配列として扱います.
I4	4 * n	VT_I4	○	n = 1 のときは単体の値として, n > 1 のときは配列として扱います.
UI8	8 * n	VT_UI8	○	n = 1 のときは単体の値として, n > 1 のときは配列として扱います.
I8	8 * n	VT_I8	○	n = 1 のときは単体の値として, n > 1 のときは配列として扱います.
BSTR	任意	VT_BSTR	×	ASCII コードでエンコードします.
Date	4	VT_DATE	×	先頭 2 バイト:年, 3 バイト目:月, 4 バイト目:日
Time	2	VT_BSTR	×	1 バイト目:時, 2 バイト目:分. HH:mm 形式の文字列として扱います.
Elappsed	5	VT_BSTR	×	経過時間. 先頭 1 バイト:単位(秒:0x41,分:0x42,時:0x43,日:0x44), 後続 4 バイトが uint で値. {値}{単位}の形の文字列で取得します. ("1234s"など). 読み取り専用です.
ObjectCode	3	VT_UI4	×	オブジェクトのコードを表します.
PropertyMap	17	VT_UI1 | VT_ARRAY	×	プロパティマップ用特殊データ構造です. 読み取り専用です.

4. ECHONET Lite プロバイダによるプログラミング

ECHONET Lite プロバイダでは，以下の手順でクライアント PC と ECHONET Lite ノードまたは ECHONET Lite 機器を接続することができます.

- CaoEngine の作成
- CaoWorkspace の作成
- CaoController の作成

ECHONET Lite ノードまたは ECHONET Lite 機器に接続した後は， CaoVariable オブジェクトを生成することで，ECHONET Lite ノードまたは ECHONET Lite 機器の情報にアクセスすることができます.

4.1. 蓄電池に接続して動作状態を取得するサンプルプログラミング

ここでは例として蓄電池の動作状態を取得し，動作中であれば停止させるサンプルプログラムを示します. 表 4-1 にサンプルプログラムの要件を，図 4-1 にサンプルプログラムの流れをそれぞれ記述しています.

表 4-1 サンプルプログラムの要件

要件	説明
接続先	UDP で接続する
	接続先 IP アドレスは 192.168.1.2
処理内容	蓄電池の動作状態を読込む.
	動作中であれば蓄電池を停止させる

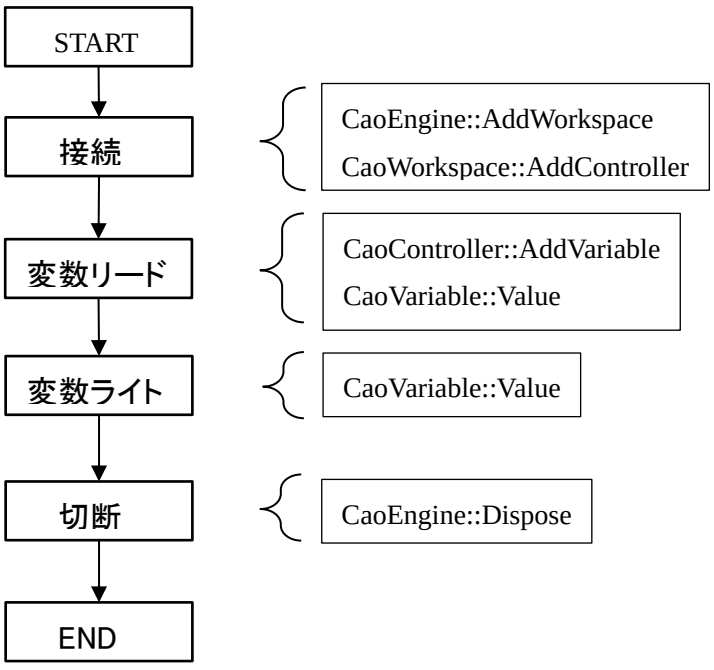

図 4-1 蓄電池に接続して動作状態を取得するプログラムの流れ

以降の節から具体的なコードを示します.

4.1.1. サンプルプログラム

以下にサンプルプログラムの全体像を示します.

Sample **Sample.cs**

```
// オブジェクト
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
    {
      // 接続
      this.Connect();
      // 値の取得
      bool operationStatus = Convert.ToBoolean(m_varOperationStatus.Value);
      // 動作状態を OFF に設定
      if (operationStatus)
      {
        m_varOperationStatus.Value = false;
      }
    }
    finally
    {
      // 切断
      this.Disconnect();
    }
}

// 接続メソッド
private void Connect()
{
    // CaoEngine オブジェクトの生成
```

```
        this.m_caoEngine = new ORiN2.ManagedCAO.CCaoEngine();
        // CaoWorkspace オブジェクトの生成
        this.m_caoWorkspace = this.m_caoEngine.AddWorkspace("NewWrks", "");
        // CaoController オブジェクトの生成
        this.m_caoController = this.m_caoWorkspace.AddController(
          "ECHONET", "CaoProv.ECHONETLite", "",
          "ObjectCode=0x027D01,Address=192.168.1.2");
        // CaoVariable オブジェクトの生成
        this.m_varIoeratuibStatus = m_caoController.AddVariable(“@OperationStatus”, “”);
    }

    // 切断メソッド
    private void Disconnect()
    {
        If (this.m_caoEngine != null)
        {
          this.m_caoEngine.Dispose();
        }
        this.m_caoEngine = null;
    }
```

4.1.1.1. 接続

ECHONET Lite ノードまたは ECHONET Lite 機器と接続するためには，以下の手順を取ります．

(1) オブジェクトを保持するための変数を用意します．コントローラ接続に必要なオブジェクトは，CaoEngine オブジェクトと CaoWorkspace オブジェクトと CaoController オブジェクトです．CaoWorkpace オブジェクトは，CaoController オブジェクトを CaoWorkspaces から取得する場合には変数を用意する必要はありません．また変数にアクセスするための CaoVariable オブジェクトも必要になります．以下に C#でのコード例を示します．

使用例(C#)

```
    // CaoEngine オブジェクト用の変数
    private ORiN2.ManagedCAO.CCaoEngine m_caoEngine = null;
    // CaoWorkspace オブジェクト用の変数
    private ORiN2.ManagedCAO.CCaoWorkspace m_caoWorkspace = null;
    // CaoController オブジェクト用の変数
    private ORiN2.ManagedCAO.CCaoController m_caoController = null;
```

```
// CaoVariable オブジェクト用の変数
private ORiN2.ManagedCAO.CCaoVariable m_ varOperationStatus = null;
```

(2) CaoEngine オブジェクトを生成します. CaoEngine オブジェクトは new キーワードを使って生成します.

使用例(C#)

```
// CaoEngine オブジェクトの生成
this.m_caoEngine = new ORiN2.ManagedCAO.CCaoEngine();
```

(3) CaoWorkspace オブジェクトを取得もしくは生成します．CaoEngine オブジェクトを生成すると，デフォルトで CaoWorkspaces オブジェクトと CaoWorkspace オブジェクトを 1 つずつ生成しています．以下に CaoWorkspace オブジェクトを新しく生成するコード例とデフォルトの CaoWorkspace を示します.

使用例(C#)

```
// CaoWorkspace オブジェクトの生成
this.m_caoWorkspace = this.m_caoEngine.AddWorkspace("NewWrks", "");
```

(4) CaoController オブジェクトを生成します．CaoController オブジェクトを生成するには，使用するプロバイダ名と使用するためのパラメータを設定します. ECHONET Lite プロバイダでは，オブジェクトコードおよびノードのアドレスまたは識別番号をオプションで指定します. 以下にコード例を示します.

使用例(C#)

```
// CaoController オブジェクトの生成
this.m_caoController = this.m_caoWorkspace.AddController(
        "ECHONET", "CaoProv.ECHONETLite", "",
        "ObjectCode=0x027D01,Address=192.168.1.2");
```

(5) CaoVariable オブジェクトを生成します．CaoVariable オブジェクトを生成するには，コントローラの AddVariable メソッドを呼び出します．以下にコード例を示します.

使用例(C#)

```
    // CaoVariable オブジェクトの生成
    this.m_varIoeratuibStatus = m_caoController.AddVariable(“@OperationStatus”, “”);
```

4.1.1.2. 動作状態の取得

CaoVariable の値を取得することにより動作状態プロパティの値を取得します.

使用例(C#)

```
// 値の取得
bool operationStatus = Convert.ToBoolean(m_varOperationStatus.Value);
```

4.1.1.3. 動作状態の設定

CaoVariable に値を設定することによりプロパティに対する書き込みを行います.

使用例(C#)

```
// 動作状態を OFF に設定
if (operationStatus)
{
   m_varOperationStatus.Value = false;
}
```

4.1.1.4. 切断

コントローラと切断する場合には，生成したオブジェクトを消去すると共に，オブジェクトを管理するコレクションクラスから消去するオブジェクトを削除します．ただし，ORiN.ManagedCAO を使用した場合は明示的に削除する必要はありません．以下にコード例を示します．

使用例(C#)

```
// CaoEngine からすべてのオブジェクトを削除
this.m_caoEngine.Dispose();
// CaoEngine の消去
this.m_caoEngine = null;
```

付録A. クラス定義ファイル

ECHONET Lite プロバイダでは機器オブジェクトのクラスを JSON 形式の定義ファイルとして配置することで機器オブジェクトの定義を追加することができます.

機器オブジェクトの定義ファイルは以下の場所に以下の名前で配置してください.

- 配置フォルダ: %ECHONET Lite プロバイダ Bin フォルダ%¥setting
- ファイル名: {クラスグループコード(0x なし 16 進数 2 桁)}{クラスコード(0x なし 16 進数 2 桁)}.json

クラス定義ファイルの内容は以下の通りです.

付録A.1. クラスオブジェクトの構造

例)

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

プロパティ名	値の型	説明
classCode	String	機種のタイプコードを"0x"＋16 進数 4 桁(ファイル名と同じ文字列)で指定してください.
description	説明オブジェクト	説明オブジェクトの構造を参照してください.
properties	プロパティオブジェクトの配列	プロパティオブジェクトの構造を参照してください.

付録A.2. プロパティオブジェクトの構造

例 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
  }
```

例 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"
  }
```

プロパティ名	値の型	説明
propertyCode	String	プロパティのコードを”0x”＋16 進数 2 桁で指定してください.
description	説明オブジェクト	説明オブジェクトの構造を参照してください.
variableName	String	コントローラのシステム変数名を指定してください.
valueType	String	ECHONET Lite で使用できる型を参照してく

		ださい.
attribute	String	変数の属性を”ReadWrite”（読み書き可能），”ReadOnly”（読み取り専用），”WriteOnly”（書き込み専用）のいずれかで指定してください. ECHONET Lite 機器との接続時にプロパティの属性が取得できた場合は ECHONET Lite 機器から取得した属性で上書きされます.
trueValue, falseValue	String	Controller.AddVariable の TrueValue オプション, FalseValue オプションを参照してください.
digit	Number	Controller.AddVariable の Digit オプションを参照してください.

付録A.3. 説明オブジェクトの構造

例）

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

プロパティ名	値の型	説明
value	String	localization プロパティに該当する言語が存在しなかったときの説明を設定してください.
localization	言語リソースオブジェクトの配列	言語リソースオブジェクトの構造を参照してください.

付録A.4. 言語リソースオブジェクトの構造

プロパティ名	値の型	説明
language	String	リソースの言語のロケールを指定してください.
value	String	languate プロパティで指定したロケールでの説明を設定してください.