村田製作所
無線センシングソリューションゲートウェイ
プロバイダ
ユーザーズ ガイド

Version 1.0.0

February 8, 2023

備考:

【改版履歴】

バージョン	日付	内容
1.0.0	2022-03-28	初版.
	2022-10-04	プロバイダ名および対応機種を修正.
	2023-02-08	誤記を修正しました.

【対応機種】

機種	バージョン	注意事項
無線センシングソリューション用ゲートウェイ		

【動作確認機種】

機種	バージョン	注意事項
無線センシングソリューション用ゲートウェイ		接続確認を行ったセンサは以下の通り LBAC0ZZ1MT-247<466> LBAC0ZZ1LZ-123<448> LBAC0ZZ1AN-783<578> LBAC0ZZ1MU-270<464> LBAC0ZZ1TF-474<556>

目次

1. はじめに

本書は，村田製作所の無線センサユニットゲートウェイと接続し，無線センサユニットゲートウェイ経由で無線センサユニットから測定情報を受信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無線センサユニットゲートウェイプロバイダと呼称します．

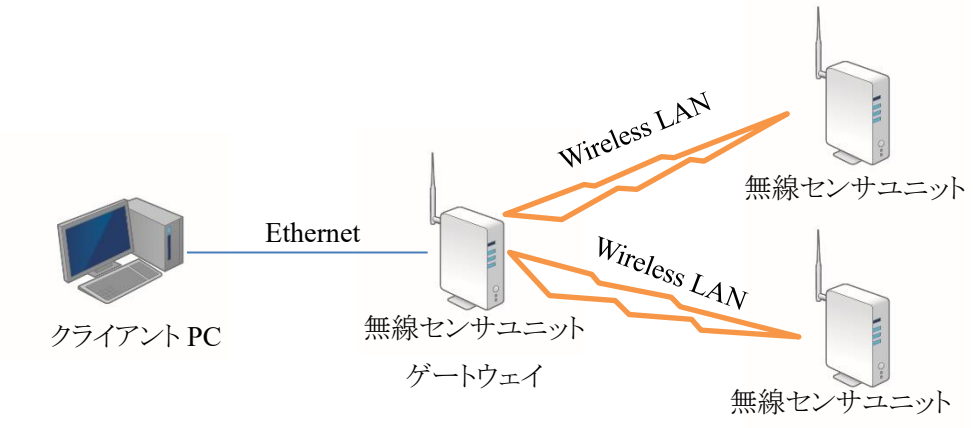

図 1-1 構成図

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. 無線センシングソリューションゲートウェイとクライアント PC との接続

無線センシングソリューションゲートウェイとクライアント PC は UDP によって通信を行います.

接続に際しては無線センシングソリューションゲートウェイとクライアント PC を直接接続するか, ネットワークハブを中継して接続します.

ルーターを超えて通信を行う場合は無線センシングソリューションゲートウェイからの電文を正しく受信できるように NAT の設定を行う必要があります.

2.2. PC 開発環境のセットアップ

2.2.1. 無線センシングソリューションゲートウェイプロバイダの手動インストール

無線センシングソリューションゲートウェイ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, regsvr32 コマンドを実行してください. 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 無線センシングソリューションゲートウェイプロバイダ

ファイル名	CaoProvMurataWirelessGateway.dll
ProgID	CaoProv.Murata.WirelessGateway
レジストリ登録	regsvr32 CaoProvMurataWirelessGateway.dll
レジストリ登録の抹消	regsvr32 /u CaoProvMurataWirelessGateway.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.9
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.10
	GetVariableNames	M	接続可能な変数名リストの取得	P.10
	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.10
	AddVariable	M	変数オブジェクトの追加	P.11
	OnMessage	E	メッセージ受信イベント	P.11
CaoVariable				
	Index	P	変数番号の取得	P.12
	Name	P	変数名の取得	P.12
	Value	P	値の取得/設定	P.12

[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.

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．無線センシングソリューションゲートウェイプロバイダでは，AddController メソッド実行時に渡されたパラメータを参照し，該当する無線センシングソリューションゲートウェイと接続を行います．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)
"CaoProv.Murata.WirelessGateway",      // 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
CONN	○	無線センシングソリューションゲートウェイの IP アドレスとポート番号を指定します．	-	-

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(
    "Gateway", "CaoProv.Murata.WirelessGateway", "",
    "CONN=UDP:192.168.0.1:55061");
```

3.2.1.1.1. CONN オプション

CONN オプションで接続先の無線センシングソリューションゲートウェイの IP アドレスおよびポート番号を指定します.

各要素は":"で区切られます.

各要素の意味は以下の通りです.

プロバイダ起動時に無線センシングソリューションゲートウェイに向けて測定開始コマンドを送信するため, 自局の IP アドレスおよび自局のポート番号を省略しても測定情報を受信できますが, 無線センシングソリューションゲートウェイを再起動した場合, WsnManager で設定した送信先 IP および送信先ポートに向けて測定情報を送信するようになりますので, 自局の IP アドレスおよび自局のポート番号を WsnManager で設定した送信先 IP および送信先ポートにあわせて設定することをお薦めします.

項目番号	意味	必須	省略時の値
0	プロトコル. "UPD"固定	○	-
1	接続先の IP アドレス	○	-
2	接続先のポート番号	-	55061
3	自局の IP アドレス	-	任意
4	自局のポート番号	-	自動

例）CONN=UDP:192.168.1.100:55061

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, 該当のコントローラが, CaoWorkspace クラスの保持するコントローラコレクションにおいてコントローラを識別するために使用されます.

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. GetVariableNames メソッド

接続可能な変数名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（未使用）
)
```

オプション

無線センシングソリューションゲートウェイプロバイダではオプションを使用しません.

使用例(C#)

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

3.2.2.4. 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];
```

3.2.2.5. 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 に示すもののみ使用できます.

以下に，AddVariable の仕様を示します.

書式

```
AddVariable
(
"<変数名>",                  // 変数名
"<オプション>"               // オプション文字列(未使用)
)
```

3.2.2.6. OnMessage イベント

無線センシングソリューションゲートウェイが無線センサユニットから受信した測定情報を 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. この番号は該当の変数を, CaoController クラスの保持する変数コレクションで識別するために使用されます.

使用例(C#)

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

3.2.3.2. Name プロパティ

CaoContrller 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. Value プロパティ

プロバイダの情報を取得します. 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

無線センシングソリューションゲートウェイプロバイダにはシステム変数のみが存在します.

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. 無線センシングソリューションゲートウェイプロバイダではシステム変数で取得できる値はプロバイダの情報を示す静的データです. 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

3.3.2. CaoController クラス変数

変数名	説明	Value		参照
		get	put	
@MAKER_NAME	メーカー名を取得します.	○	-	P.12
@VERSION	DLL バージョンを取得します.	○	-	P.13

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;
```

3.4. イベント一覧

無線センシングソリューションゲートウェイが無線センサユニットから受信した測定情報を OnMessage イベントとして受け取ることが可能です.

無線センサユニットからの測定情報は Number が 0 のメッセージとして受け取れます.

3.4.1.1. 測定情報メッセージ

無線センシングソリューションゲートウェイが無線センサユニットから受信した測定情報を通知するメッセージです.

Value プロパティデータ型

型説明		
VT_ARRAY | VT_VARIANT		
0	VT_UI2	測定情報を送信した無線センサユニットの ID です.
1	VT_I2	測定情報を受信したときの電波強度(RSSI)です.
2	VT_UI1	センサ種別です. 詳細は 3.4.1.1.1 を参照してください.

型説明		
3	VT_VARIANT | VT_ARRAY	測定値です. 内部のデータの並び順や値の型はセンサ種別ごとに異なります. 測定値のない要素は VT_EMPTY が格納されます. 詳細は 3.4.1.1.2~3.4.1.1.11 を参照してください.
4	VT_UI1 | VT_ARRAY	測定値の単位です. 要素数は測定値と同様となり, 測定値と同じ要素番号の単位が測定値の単位となります. 単位の意味は xxx を参照してください.

使用例(C#)

```
/// <summary>
///  メッセージ受信時の処理メソッド例
/// </summary>
/// <param name="sender">イベント送信元</param>
/// <param name="e">イベント引数</param>
private void OnMessage(object sender, OnMessageEventArgs e)
{
    object[] dataArray = e.Message.Value as object[];
    if (dataArray != null)
    {
        ushort unitId = Convert.ToUInt16(dataArray[0]);
    }
}
```

3.4.1.1.1. センサ種別の一覧

センサ種別は以下の通りです.

センサ種別	センサ種別名	製品名称末尾 3 文字[2]
0x01	温湿度	1AN
0x07	3 温度/熱電対	1EM/1PF
0x10	電流・パルス	1MU
0x13	電圧・パルス	1RU
0x12	CT	1NT/1MT
0x09	振動	1LZ
0x18	振動(速度)	1TF
0x1B	3 電流	1ZU

[2] 製品名称 LBAC0ZZxxx-***の xxx の部分

0x1C	3 電圧	1ZV
0x1D	3 接点	1ZS

3.4.1.1.2. 温湿度(0x01)の測定値

温湿度の測定値は以下の通りです.

要素番号	値の型	値の意味
0	VT_R8	電源電圧
1	VT_R8	温度
2	VT_R8	湿度

3.4.1.1.3. 3 温度/熱電対(0x07)の測定値

温度/熱電対の測定値は以下の通りです.

要素番号	値の型	値の意味
0	VT_R8	電源電圧
1	VT_R8	温度 1
2	VT_R8	温度 2
3	VT_R8	温度 3

3.4.1.1.4. 電流・パルス(0x10)の測定値

電流・パルスの測定値は以下の通りです.

要素番号	値の型	値の意味
0	VT_R8	電源電圧
1	VT_R8	カウンタ
2	VT_R8	電流
3	VT_R8	パルスカウント
4	VT_R8	前回電流
5	VT_R8	前回パルスカウント

3.4.1.1.5. 電圧・パルス(0x13)の測定値

電圧・パルスの測定値は以下の通りです.

要素番号	値の型	値の意味
0	VT_R8	電源電圧
1	VT_R8	カウンタ
2	VT_R8	電圧
3	VT_R8	パルスカウント

4	VT_R8	前回電圧
5	VT_R8	前回パルスカウント

3.4.1.1.6. CT(0x12)の測定値

CT の測定値は以下の通りです.

要素番号	値の型	値の意味
0	VT_R8	電源電圧
1	VT_R8	カウンタ
2	VT_R8	電流 1
3	VT_R8	電流 2
4	VT_R8	借電
5	VT_R8	前回電流 1
6	VT_R8	前回電流 2
7	VT_R8	前回借電

3.4.1.1.7. 振動(0x09)の測定値

振動の測定値は以下の通りです.

要素番号	値の型	値の意味
0	VT_R8	電源電圧
1	VT_R8	ピーク周波数 1
2	VT_R8	ピーク加速度 1
3	VT_R8	ピーク周波数 2
4	VT_R8	ピーク加速度 2
5	VT_R8	ピーク周波数 3
6	VT_R8	ピーク加速度 3
7	VT_R8	ピーク周波数 4
8	VT_R8	ピーク加速度 4
9	VT_R8	ピーク周波数 5
10	VT_R8	ピーク加速度 5
11	VT_R8	加速度 RMS
12	VT_R8	尖り度
13	VT_R8	温度

3.4.1.1.8. 振動（速度）(0x18)の測定値

振動(速度)の測定値は以下の通りです.

要素番号	値の型	値の意味
0	VT_R8	電源電圧
1	VT_R8	ピーク周波数 1
2	VT_R8	ピーク加速度 1
3	VT_R8	ピーク周波数 2
4	VT_R8	ピーク加速度 2
5	VT_R8	加速度 RMS
6	VT_R8	速度ピーク周波数 1
7	VT_R8	ピーク速度 1
8	VT_R8	速度ピーク周波数 2
9	VT_R8	ピーク速度 2
10	VT_R8	速度ピーク周波数 3
11	VT_R8	ピーク速度 3
12	VT_R8	速度 RMS
13	VT_R8	尖り度
14	VT_R8	温度

3.4.1.1.9. 3 電流 (0x1B)の測定値

3 電流の測定値は以下の通りです.

要素番号	値の型	値の意味
0	VT_R8	電源電圧
1	VT_BSTR	シリアルナンバー
2	VT_R8	電流値 1
3	VT_R8	電流値 2
4	VT_R8	電流値 3

3.4.1.1.10. 3 電圧 (0x1C)の測定値

3 電圧の測定値は以下の通りです.

要素番号	値の型	値の意味
0	VT_R8	電源電圧
1	VT_BSTR	シリアルナンバー
2	VT_R8	電圧値 1
3	VT_R8	電圧値 2
4	VT_R8	電圧値 3

3.4.1.1.11. 3 接点 (0x1D)の測定値

3 電圧の測定値は以下の通りです.

要素番号	値の型	値の意味
0	VT_R8	電源電圧
1	VT_BSTR	シリアルナンバー
2	VT_R8	検出エッジ 1
3	VT_R8	エッジカウント 1
4	VT_R8	検出エッジ 2
5	VT_R8	エッジカウント 2
6	VT_R8	検出エッジ 3
7	VT_R8	エッジカウント 3

3.4.1.1.12. 単位一覧

以下に単位の一覧を示します.

単位コード	意味	単位
0x01	Duty 比	%
0x02	圧力	Pa
0x03	位置_緯度	
0x04	位置_経度	
0x05	インダクタンス	H
0x06	エネルギー	J
0x07	カウント	Time
0x08	角加速度	rad/s^2
0x09	拡散率	m^2/s
0x0A	角速度	rad/s
0x0B	角度	°
0x0C	加速度	m/s^2
0x0D	輝度	cd/m^2
0x0E	吸収線量	Gy
0x0F	吸収線量率	Gy/s
0x10	屈折率	-
0x11	酵素活性	kat
0x12	酵素活性濃度	kat/m^3
0x13	光束	lm
0x14	光量	cd
0x15	コンダクタンス	S
0x16	サイズ	Byte
0x17	磁界	A/m
0x18	時間	sec
0x19	時刻	Hour
0x1A	時刻	minute
0x1B	時刻	second
0x1C	時刻	millsecond
0x1D	仕事率	W
0x1E	磁束	Wb

単位コード	意味	単位
0x1F	磁束密度	T
0x20	質量	kg
0x21	質量吸収係数	m^2/kg
0x22	質量濃度	kg/m^3
0x23	質量モル濃度	mol/kg
0x24	質量流量	kg/s
0x25	質量流量密度	$kg/m^2 \cdot s$
0x26	周波数	Hz
0x27	照射線量	C/kg
0x28	照度	lx
0x29	数値	-
0x2A	セルシウス温度	℃
0x2B	線量当量	Sv
0x2C	相対湿度	%RH
0x2D	体積	m^3
0x2E	体積エネルギー	J/m^3
0x2F	体積流量	m^3/s
0x30	力	N
0x31	力のモーメント	N・m
0x32	電位差(電圧)	V
0x33	電荷	C
0x34	電界の強さ	V/m
0x35	電荷密度	C/m^3
0x36	電気抵抗	ohm
0x37	電気容量	F
0x38	電束密度	C/m^2
0x39	電流	A
0x3A	電流密度	A/m^2
0x3B	電力量	kWh
0x3C	透磁率	H/m

単位コード	意味	単位
0x3D	導電率	S/m
0x3E	動粘度	m^2/s
0x3F	長さ	m
0x40	日射量	kW/m^2
0x41	熱伝導率	W/(m・K)
0x42	熱容量	J/K
0x43	熱力学温度	K
0x44	熱流密度	W/m^2
0x45	粘度	Pa・s
0x46	濃度	ppm
0x47	波数	m^{-1}
0x48	速さ	m/s
0x49	比エネルギー	J/kg
0x4A	比透磁率	-
0x4B	比熱容量	J/(kg・K)
0x4C	比容積	m^3/kg
0x4D	表面張力	N/m
0x4E	表面電荷	C/m^2
0x4F	物質量	mol
0x50	平面角	rad
0x51	放射輝度	$W/(m^2・sr)$
0x52	放射強度	W/sr
0x53	放射能	Bq
0x54	密度	kg/m^3
0x55	面積	m^2
0x56	面密度	kg/m^2
0x57	モルエネルギー	J/mol

単位コード	意味	単位
0x58	モルエントロピー	J/(mol・K)
0x59	モル体積	m^3/mol
0x5A	モル導電率	$S・m^2/mol$
0x5B	誘電率	F/m
0x5C	力積	N・s
0x5D	立体角	sr
0x5E	量濃度	mol/m^3
0x5F	放射線量	uSv
0x60	人感センサ	-
0x61	磁気センサ	-
0x62	電流	mA
0x63	加速度実効値	m/s^2
0x64	気圧	hPa
0x65	ADC 出力	-
0x66	尖度	-
0x68	単位なし	-
0x69	電力	W
0x6A	流量	L/min
0x6B	体積	L
0x71	角度	rad
0x72	速度	mm/s
0x73	変位	mm
0x74	速度実効値	mm/s
0xFA	シリアル No	-
0xFE	レンジオーバー	-
0xFF	異常	-

4. 無線センシングソリューションゲートウェイプロバイダによるプログラミング

無線センシングソリューションゲートウェイプロバイダでは，以下の手順でクライアント PC と無線センシングソリューションゲートウェイを接続することができます．

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

無線センサユニットに接続した後は，CaoController の OnMessage イベントで**無線センサユニットから送信される測定情報を取得**することができます．

4.1. 温湿度計の測定情報を表示するサンプルプログラミング

ここでは例として温湿度計の測定情報を表示する ManagedCAO を利用した C#の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 アドレスおよびポートはコンソールから読み込む
処理内容	OnMessage 受信時に温湿度計の測定情報だった場合は画面に情報の内容を表示する．

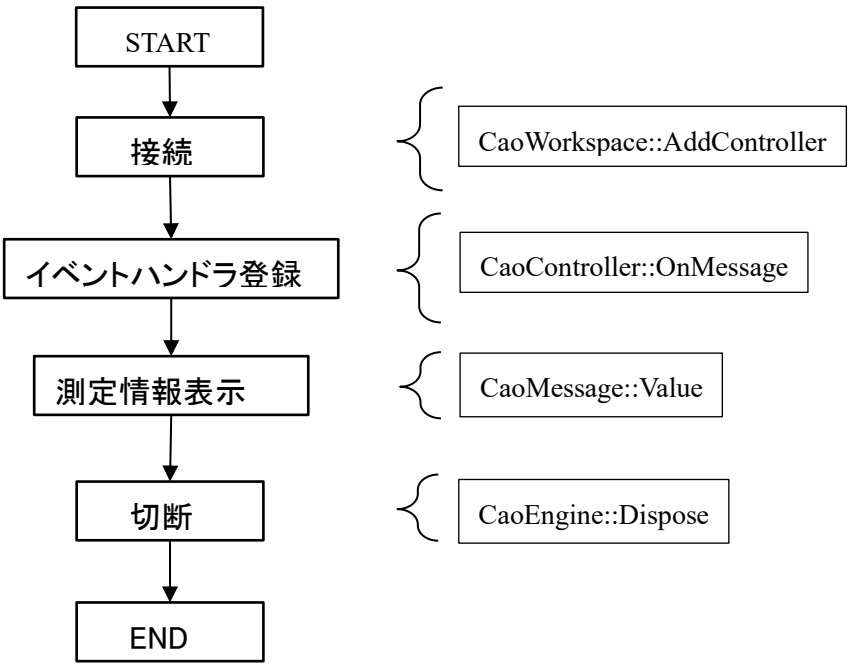

図 4-1 サンプルプログラムの流れ

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 **Program.cs**

```
using System;
using ORiN2.ManagedCAO;

namespace MurataWirelessGatewaySample
{
    /// <summary>
    ///  温湿度計の測定情報を画面に表示するプログラム
    /// </summary>
    class Program
    {
        /// <summary>
        ///  メイン処理
        /// </summary>
        /// <param name="args">実行時引数</param>
        static void Main(string[] args)
        {
```

```
        try
        {
            // CONN=UDP:に続くオプションを入力させる
            string option = "CONN=UDP:";
            Console.Write(option);
            option += Console.ReadLine();

            // CaoEngine の作成
            // using で囲うとスコープから抜けたときに自動的に
            // Dispose が実行されて解放される
            using (CCaoEngine engine = new CCaoEngine())
            {
                // CaoController の作成
                // engine.Workspaces[0]  は自動的に生成される
                CCaoController controller = engine.Workspaces[0].AddController(
                    "Gateway", "CaoProv.Murata.WirelessGateway", null, option);
                // OnMessage イベントハンドラの登録
                controller.OnMessage += Controller_OnMessage;

                Console.WriteLine("Hit any key to exit.");
                Console.ReadKey();

                // OnMessage イベントハンドラの解除
                controller.OnMessage -= Controller_OnMessage;
                //  次の行の}でスコープを抜けるため，Dispose が呼び出される
            }
        }
        catch(Exception ex)
        {
            //  例外の内容をコンソールに出力
            Console.WriteLine(ex);
        }
    }

    /// <summary>
    ///  メッセージ処理関数
    /// </summary>
```

```
/// <param name="sender">送信元のオブジェクト</param>
/// <param name="e">イベント引数</param>
private static void Controller_OnMessage(object sender, OnMessageEventArgs e)
{
    try
    {
        object[] values = e.Message.Value as object[];
        // 目的以外のメッセージは処理しない
        if (e.Message.Number != 0
            || values == null
            || values.Length != 5
            || Convert.ToByte(values[2]) != 0x01)
        {
            return;
        }

        ushort sensorId = Convert.ToUInt16(values[0]);
        short rssi = Convert.ToInt16(values[1]);
        byte sensorTypeId = Convert.ToByte(values[2]);
        object[] sensorValues = values[3] as object[];
        if (sensorValues == null || sensorValues.Length != 3)
        {
            Console.WriteLine("invalid message.");
            return;
        }
        double supplyVoltage = Convert.ToDouble(sensorValues[0]);
        double temperature = Convert.ToDouble(sensorValues[1]);
        double humidity = Convert.ToDouble(sensorValues[2]);

        Console.WriteLine(
            "sensorId:{0:X4}, rssi:{1}, sensorTypeId:{2:X2}, " +
            supplyVoltage:{3}, temperature:{4}, humidity:{5}",
            sensorId, rssi, sensorTypeId, supplyVoltage, temperature, humidity);
    }
    catch (Exception ex)
    {
        // 例外の内容をコンソールに出力
```

```
                Console.WriteLine(ex);
            }
        }
    }
}
```

4.1.1.1. 接続

無線センシングソリューションゲートウェイと接続するためには，以下の手順を取ります．

(1) CaoEngine オブジェクトおよび CaoController オブジェクトを生成します．

CaoEngine オブジェクトの使用を終了するときは CaoEngine.Dispose() を呼び出してオブジェクトを解放する必要があります．

using 節で囲うことでスコープを抜けたときに自動的に Dispose が呼び出されますので，解放忘れを防ぐことができます．

CaoEngine の生成時に CaoEngine.Workspaces[0]に自動的に CaoWorkspace が生成されます．

この CaoWorkspace オブジェクトの AddController 関数を呼び出すことで CaoController オブジェクトが生成され，無線センシングソリューションゲートウェイに接続されます．

使用例(C#)

```
// CaoEngine の作成
// using で囲うとスコープから抜けたときに自動的に
// Dispose が実行されて解放される
using (CCaoEngine engine = new CCaoEngine())
{
    // CaoController の作成
    // engine.Workspaces[0] は自動的に生成される
    CCaoController controller = engine.Workspaces[0].AddController(
        "Gateway", "CaoProv.Murata.WirelessGateway", null, option);
```

(2) CaoController オブジェクトに OnMessage イベントを受け取るイベントハンドラを登録します．

CaoController オブジェクトの OnMessage イベントにイベントを受け取るイベントハンドラを登録します．

登録するイベントハンドラの型は次のようになっています．

```
public delegate void OnMesssageEventHandler(object sender, OnMessageEventArgs e);
```

使用例(C#)

```
controller.OnMessage += Controller_OnMessage;
```

4.1.1.2. イベントの処理

(1) OnMessage イベントを受け取った時の処理を記述します.

無線センサユニットから測定データを受信したとき，CaoController の OnMessage イベントが発生します.

イベント引数の Message プロパティに受信した測定データが格納されていますので，測定データに対して処理を行ってください.

サンプルプログラムでは温湿度計からの測定データのみをコンソールに出力するようになっています.

使用例(C#)

```
/// <summary>
/// メッセージ処理関数
/// </summary>
/// <param name="sender">送信元のオブジェクト</param>
/// <param name="e">イベント引数</param>
private static void Controller_OnMessage(object sender, OnMessageEventArgs e)
{
    try
    {
        object[] values = e.Message.Value as object[];
        // 目的以外のメッセージは処理しない
        if (e.Message.Number != 0
            || values == null
            || values.Length != 5
            || Convert.ToByte(values[2]) != 0x01)
        {
            return;
        }

        ushort sensorId = Convert.ToUInt16(values[0]);
        short rssi = Convert.ToInt16(values[1]);
        byte sensorTypeId = Convert.ToByte(values[2]);
        object[] sensorValues = values[3] as object[];
```

```
                if (sensorValues == null || sensorValues.Length != 3)
                {
                    Console.WriteLine("invalid message.");
                    return;
                }
                double supplyVoltage = Convert.ToDouble(sensorValues[0]);
                double temperature = Convert.ToDouble(sensorValues[1]);
                double humidity = Convert.ToDouble(sensorValues[2]);
                Console.WriteLine(
                    "sensorId:{0:X4}, rssi:{1}, sensorTypeId:{2:X2}, " +
                    supplyVoltage:{3}, temperature:{4}, humidity:{5}",
                    sensorId, rssi, sensorTypeId, supplyVoltage, temperature, humidity);
            }
            catch (Exception ex)
            {
                // 例外の内容をコンソールに出力
                Console.WriteLine(ex);
            }
        }
```

4.1.1.3. 切断

コントローラと切断する場合には，登録したイベントハンドラを解除すると共に，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，ManagedCAO を使用した場合は CaoEngine の Dispose の呼び出しで自動的にすべてのオブジェクトが削除されますので，明示的に削除の処理を記述する必要はありません．

使用例(C#)

```
                // OnMessage イベントハンドラの解除
                controller.OnMessage -= Controller_OnMessage;
                // 次の行の}でスコープを抜けるため，Dispose が呼び出される
            }
```