KEYENCE
LJ-X8000 プロバイダ
ユーザーズ ガイド

Version 1.0.0

July 2, 2020

備考：

【改版履歴】

バージョン	日付	内容
1.0.0	2020-07-02	初版.

【動作確認機種】

機種	バージョン	注意事項
LJ-X8000		

目次

1. はじめに

本書は，株式会社 KEYENCE の LJ-X8000 からデータの取得を行う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 LJ-X8000 プロバイダと呼称します．

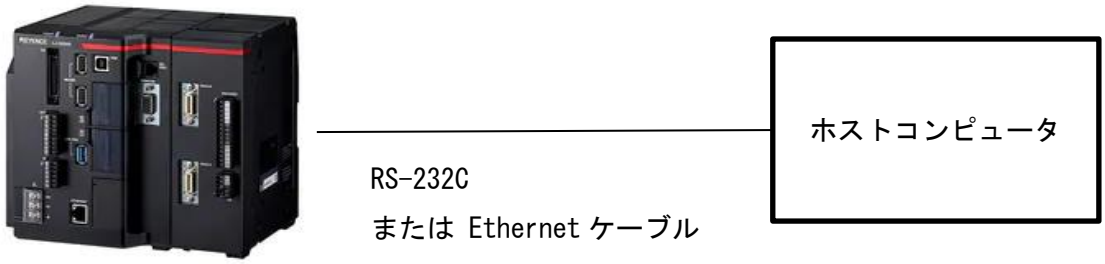

図 1-1 構成図

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．）

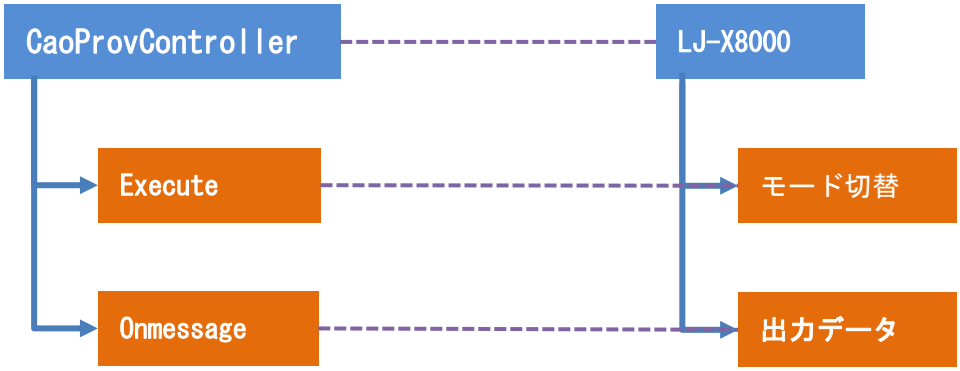

図 1-2 プロバイダの構成とデバイス情報との対応図

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. LJ-X8000 とクライアント PC との接続

LJ-X8000 とクライアント PC を接続するためには,Ethernet ケーブルによる TCP 接続またはモジュラーケーブルを使った COM 接続を行う必要があります.

また本プロバイダとのデータ受信を行うためには，LJ-X8000 側でデリミターの設定や出力設定を行う必要があります.

2.1.1. LJ-X8000 のデリミター設定

本プロバイダでは，LJ-X8000 とのデータ送受信する際のデリミターは,【CR】または【CR+LF】に対応しております. LJ-X8000 で無手順コマンドおよび出力データのデリミター設定を上記以外に設定している場合は，設定を変更してください. 設定方法については，LJ-X8000 シリーズのユーザーズマニュアルを参照してください.

2.1.2. 出力データの受信方法

本プロバイダでは,LJ-X8000 で出力設定したデータを受信する機能があります. COM 接続で接続する場合は RS-232C(無手順通信)に出力設定をし，Ethernet で接続する場合は，Ethernet(無手順通信)に出力設定をしてください. 設定方法については LJ-X8000 シリーズのユーザーズマニュアルを参照してください.

2.1.3. トリガー入力について

LJ-X8000 はトリガー入力を受け付けるとヘッダから撮像を開始しデータの出力を行います. 本プロバイダでは,LJ-X8000 に対してトリガーを発行することができます(3.2.2.4.1 参照)が本プロバイダからのトリガー入力を受け付けるためには，LJ-X8000 側の設定でトリガー設定を外部トリガー設定し，トリガーモードで RS-232C または Ethernet にチェックを入れる必要があります. トリガー設定については，LJ-X8000 シリーズのユーザーズガイドを参照してください.

3. コマンドリファレンス

3.1. メソッド/プロパティ一覧

表 3-1 メソッド/プロパティ一覧

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				
	VariableNames	P	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.12
	OnMessage	E	メッセージ受信イベント	P.15
CaoVariable				
	Value	P	値の取得/設定	P.15

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．LJ-X8000 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 LJ-X8000 シリーズのコントローラと接続を行います．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.KEYENCE.LJ-X8000",     // 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)
)
```

[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．

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
Conn=<通信パラメータ>	○	接続先情報を指定します	ETH or COM
Timeout=	--	通信タイムアウトをmsで指定します	1-65535 デフォルト値：2000
Delimiter=	--	LJ-X8000に設定したデリミターを指定します(2.1.1参照). 0:CR 1:CR+LF	0-1 デフォルト:0

使用例

```
// 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("LJ-X8000",
                                        "CaoProv.KEYENCE.LJ-X8000",
                                        "",
                                        "Conn=ETH:192.168.10.10,Timeout=5000,delimiter=1");
```

3.2.1.1.1. CONNオプション

以下にConnオプションの接続パラメータ文字列を示します. ここで角括弧("[]")内は省略可能なことを,各パラメータの解説中の下線部はオプションを指定しなかった時のデフォルト値をそれぞれ示します.

TCP/IPで接続する場合

"Conn=ETH:<接続先IP>[:<接続先ポート>[:<ローカルIP>[:<ローカルポート>]]]"

<接続先IP> : 接続先IPアドレスを***.***.***.***の形式で指定します.
この項目は必ず指定してください.

<接続先ポート> : 接続先ポート番号を指定します. 8500

RS232Cで接続する場合

"Conn=COM:<COM Port>[:<BaudRate>[:<Parity>:<DataBits>:<StopBits>[:<Flow>]]]"

<COM Port> : COMポート番号. '1'-COM1, '2' - COM2, ...

<BaudRate> : 通信速度. 4800, 9600, 19200, 38400, 57600, 115200

<Parity> : パリティ. 'N'-NONE, 'E'-EVEN, 'O'-ODD

<DataBits> : データビット数. '7'-7bit, '8'-8bit

<StopBits> : ストップビット数. '1'-1bit, '2'-2bit
<Flow> : フロー制御. '0'-None, '1'-Xon/Xoff, '2'-ハードウェア制御
OR をとって指定できます.

3.2.1.1.2. AddController 時の注意事項

AddController 時に LJ-X8000 からデバイスのバージョン情報を取得しています. バージョン情報の取得が失敗した場合は, 0x80111000 のエラーコードが返却され, AddController が失敗します. デバイスのバージョンを取得できた場合は, CaoController 内で保持しており, 後述する CaoVariable クラスの@DEVICE_VERSION 変数, @DEVICE_MODEL でデバイスとの通信を行わずに取得する際に使用されます.

3.2.2. CaoController クラス

3.2.2.1. VariableNames プロパティ

接続可能な変数名リストを取得します. 本プロパティで取得した変数名は, 後述する AddVariable メソッドの第一引数に使用することができます.

使用例

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

3.2.2.2. 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

```
// 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];
```

3.2.2.3. 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 3.3.1 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(省略可能)
)
```

3.2.2.4. 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.

一部コマンドは運転モード状態によりコマンドは成功しているが LJ-X8000 側で実行されないことがあります. 動作については, 各コマンドの説明をご参照ください.

コマンド	説明	参照
Trigger	トリガーを発行します.	P.12
ClearError	エラー情報をクリアします.	P.13
ChangeMode	運転モードを切り替えます.	P.13
GetMode	現在の運転モード状態を読み出します.	P.13
ChangeOperationScreen	運転画面を切り替えます.	P.14
SendRowCommand	パラメータのデータをそのままデバイスに転送します.	P.14

3.2.2.4.1. Trigger コマンド

LJ-X8000 にトリガー発行のコマンドを送信します. トリガーを発行することで LJ-X8000 に設定しているヘッダが 1 回だけ撮像を行います. LJ-X8000 がトリガーを受け取るためには, LJ-X8000 本体のトリガー設定で外部トリガーを選択している必要があります.

項目	型説明
引数	特になし
戻り値	特になし

使用例

```
// トリガーの発行をします.
controller.Execute("Trigger");
```

3.2.2.4.2. ClearError コマンド

LJ-X8000 のエラー情報をクリアするコマンドを送信します. このコマンドは設定モード/運転モード関係なく実行できます.

項目	型説明
引数	特になし
戻り値	特になし

使用例

```
// エラー情報をクリアします.
controller.Execute("ClearError");
```

3.2.2.4.3. ChangeMode コマンド

LJ-X8000 の運転状態モードを切り替えるコマンドを送信します. オプションに設定する値によって設定モードか運転モードに切り替わります.

モードにより LJ-X8000 がコマンドを受け付けた際の動作が異なります. 本プロバイダで実装しているコマンドに関しては対象のコマンド説明をご参照ください.

項目	型説明		
引数	VT_UI1	切り替えるモードを指定します.	
		設定値	モード
		0	設定モード
		1	運転モード
戻り値	特になし		

使用例

```
// 運転モードに切り替える
controller.Execute("ChangeMode",1);
```

3.2.2.4.4. GetMode コマンド

LJ-X8000 の現在の運転状態を取得するコマンドを送信します. 本コマンドは設定/運転モードで実行することが可能になります. このコマンドは設定モード/運転モード関係なく実行できます.

項目	型説明		
引数	特になし		
戻り値	VT_UI1	現在の運転状態を取得します.	
		取得値	モード
		0	設定モード
		1	運転モード

使用例

```
// 現在の運転状態モードを取得します
Byte mode = controller.Execute("GetMode") as Byte;
```

3.2.2.4.5. ChangeOperationScreen コマンド

LJ-X8000 の運転画面を切り替えるコマンドを送信します．タブ種別とタブ番号を指定することで運転画面を切り替えます．本コマンドは運転モード中のみ実行されます．運転状態モードを運転モードにしてから実行してください．

項目	型説明		
引数	VT_UI1	タブ種別を指定します．	
		指定値	種別
		0	ツール一覧タブ
		1	S**タブ
	VT_UI1	タブ種別により設定範囲が異なります． タブ種別がツール一覧タブの場合は，1 を指定して下さい． 範囲：0 ~ 9	
戻り値	特になし		

使用例

```
// S00タブに運転画面を切り替える
object[] opt = new object[]{ 1, 0 };
controller.Execute("ChangeOperationScreen", );

// ツール一覧タブに運転画面を切り替える
object[] opt = new object[]{ 0, 1 };
controller.Execute("ChangeOperationScreen", );
```

3.2.2.4.6. SendRawCommand コマンド

LJ-X8000 にパラメータで指定したデータをそのまま送信します．

項目	型説明	
引数	VT_BSTR	送信するデータを文字列型で指定してください．
戻り値	VT_BSTR	LJ-X8000 からの応答データを文字列型で返します．

使用例

```
// SendRawCommandでVIコマンドを実行する
String opt = "VI"
String retValue = controller.Execute("SendRawCommand",opt) as String;
```

3.2.2.5. 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. 受け取ることが可能なイベントについては 0 を参照してください.

3.2.3. CaoVariable クラス

3.2.3.1. Value プロパティ

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. 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.15
@VERSION	DLL バージョンを取得します.	○	-	P.16
@DEVICE_VERSION	デバイスのバージョンを取得します.	○	-	P.16
@DEVICE_MODEL	デバイスの型式番号を取得します.	○	-	P.16

3.3.1.1. @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

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

3.3.1.2. @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

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

3.3.1.3. @DEVICE_VERSION

デバイスのバージョンの取得をします. デバイスのバージョン取得は, AddController 時に取得したデータを CaoController で保持しておりその値が取得されます.

データ型

型説明	
VT_BSTR	デバイスのバージョンを 14 文字の文字列で取得します.

使用例

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

3.3.1.4. @DEVICE_MODEL

デバイスの型式番号を取得します. デバイスの型式番号取得は, AddController 時に取得したデータを CaoController で保持しておりその値が取得されます.

データ型

型説明	
VT_BSTR	デバイスの型式番号

使用例

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

3.4. イベント一覧

LJ-X8000 で出力データ設定を行い，出力データをプロバイダが受け取った場合，OnMessage イベントとして受け取ることが可能です.

各メッセージの内容は Value プロパティで取得することができます.

3.4.1. 出力データ

LJ-X8000 からの出力データを文字列で返却します.

データ型

メッセージ番号	メッセージ内容	
1	VT_BSTR	LJ-X8000 コントローラからの出力データ

3.4.2. 切断メッセージ

LJ-X8000 からのデータ受信確認時に異常が見られた場合に切断されたと判断しメッセージが発行されます．このメッセージが発行された場合は，再接続を行ってください.

また COM 接続時は，異常の検知ができないためメッセージの発行がされません．COM 通信時に断線しているか確認したい場合は，CaoController::Execute コマンドの GetMode コマンドを使いデータの受信ができない場合に断線している判断できるのでその際は再接続を行ってください.

データ型

メッセージ番号	メッセージ内容	
99	VT_I4	エラーコード番号

4. LJ-X8000 プロバイダによるサンプルプログラミング

LJ-X8000 プロバイダでは，以下の手順でクライアント PC と LJ-X8000 のコントローラと接続することができます.

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

LJ-X8000 に接続した後は，CaoController の Execute メソッドを使用する，または LJ-X8000 から設定した出力データを受け取ることができます.

4.1. LJ-X8000 から出力データを受信するサンプルプログラミング

ここでは例として LJ-X8000 で出力設定したデータを受信していくサンプルプログラミングについて解説します．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	TCP/IP で接続する
	接続先 IP アドレスは 192.168.10.10
	接続先ポート番号は 8500
処理内容	LJ-X8000 からの出力データを受信する
	受信データをカンマ区切りで分割する.

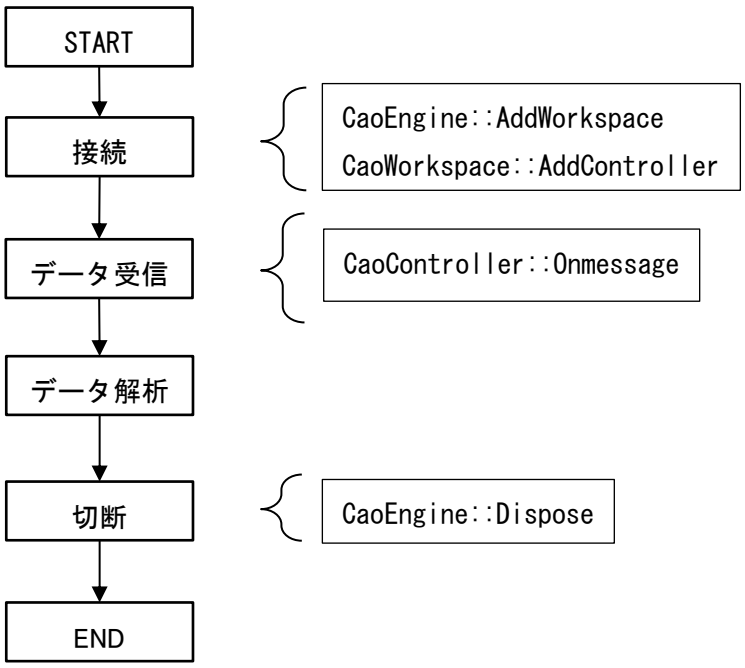

図 4-1 出力データ受信の流れ

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 SampleApp.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;

public void Main()
{
    // 接続
    this.Connect();
    // コントローラにOnmessageイベントを受信した際の動作を追加
    m_caoController.OnMessage += new
ORiN2.ManagedCAO.OnMesssageEventHandler(OnMessageEvent);
    // OnMessageイベントを受け付けるために10秒間待機
    Thread.Sleep(10000);

    // 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("LJX8000",
        "CaoProv.KEYENCE.LJ-X8000",
        "",
        "Conn=ETH:127.0.0.1:8500, Timeout=1000");
}

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

/// <summary>
/// OnMessage受信イベント
/// </summary>
private void OnMessageEvent(object sender, ORiN2.ManagedCAO.OnMessageEventArgs e)
{
    // メッセージの内容を文字列型で取得
    string messageData = e.Message.Value as string;
    // メッセージ内容をカンマ区切りで分割する
    string[] messageArray = messageData.Split(',');
}
```

4.1.1.1. 接続

LJ-X8000 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．

```
    // 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_varD0100 = null;
// CaoVariable オブジェクト用の変数
private ORiN2.ManagedCAO.CCaoVariable m_varD1100 = 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.

```
// 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.

```
// 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. LJ-X8000プロバイダでは, Connオプションで通信先のデバイスを指定します. 以下にコード例を示します.

```
// CaoController オブジェクトの生成
this.m_caoController = this.m_caoWorkspace.AddController("LJX8000",
        "CaoProv.KEYENCE.LJ-X8000",
        "",
        "Conn=ETH:127.0.0.1:8500, Timeout=1000");
```

4.1.1.2. データの受信

(1) LJ-X8000プロバイダでは, LJ-X8000シリーズからの出力データを受信した際にOnMessageイベントが発行されます. OnMessageイベントを受信するためにはCaoControllerオブジェクトのOnMessageイベントハンドラにイベント受信時の動作を設定します. 以下にコード例を示します.

```
// コントローラにOnmessageイベントを受信した際の動作を追加
m_caoController.OnMessage += new
ORiN2.ManagedCAO.OnMesssageEventHandler(OnMessageEvent);
```

(2) OnMessageイベントが発行されると上記で登録したイベントが実行され,取得データはOnMessageのValueプロパティで取得することができます. 以下にコードの例を示します.

```
/// <summary>
/// OnMessage受信イベント
/// </summary>
private void OnMessageEvent(object sender, ORiN2.ManagedCAO.OnMessageEventArgs e)
{
    // メッセージの内容を文字列型で取得
    string messageData = e.Message.Value as string;
    // メッセージ内容をカンマ区切りで分割する
    string[] messageArray = messageData.Split(',');
}
```

4.1.1.3. 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.

```
    // 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;
```

5. LJ-X8000 プロバイダエラーコード

本プロバイダには，メソッド，プロパティ，イベントごとに独自エラーコードが存在します.

ORiN2 の共通エラーについては，「ORiN2 プログラミングガイド」のエラーコードの章を参照してください.

5.1. CaoWorkspace::AddController 時のエラーコード一覧

表 5-1 AddController 時のエラーコード表

エラー番号	説明
0x80110000	指定Connオプションで疎通確認が失敗しました. 指定のConnオプションがLJ-X8000シリーズで設定されているIPアドレス，ポート番号またはCOMポートか確認してください.
0x80111001	Connオプションの指定方法が間違っています. Connオプションが正しく指定できているか確認してください.
0x80111002	Timeoutオプションの指定方法が間違っています. Timeoutオプションの指定内容を確認してください.
0x80111003	Delimiterオプションの指定方法が間違っています. Delimiterオプションの指定内容を確認してください.

5.2. CaoController::Execute 時のエラーコード一覧

表 5-2 CaoController::Execute 時のエラーコード表

エラー番号	説明
0x80121001	指定オプションの引数が足りていません. 実行するコマンドのオプション引数をご確認ください.
0x80121002	オプション指定方法が間違っています. 実行するコマンドのオプションをご確認ください.
0x80121005	想定外のデータを受信しました. 通信環境等の見直しをしてください.
0x801200**	デバイスからエラーコード「**」が返却されました. **=実行コマンドに対するデバイスからのエラーコード ※デバイスからのエラーコードに関しては「LJ-X8000シリーズ ユーザーズガイド」の9章にある制御用通信コマンドの一覧をご参照ください.

付録 A. 通信プロトコルコマンド対応表

CaoControllre::AddController メソッド	LJ-X8000 無手順通信コマンド
疎通確認	VI
CaoControllre::Execute メソッド	LJ-X8000 無手順通信コマンド
Trigger	T1
ClearError	CE
ChangeMode(運転モード指定時)	RO
ChangeMode(設定モード指定時)	SO
GetMode	RM
ChangeOperationScreen	VW