株式会社ソフィックス
SOFIXCAN ΩEye プロバイダ
ユーザーズ ガイド

Version 1.0.0

July 17, 2024

備考:本プロバイダを使用した SOFIXCAN ΩEye for Windows のデータ取得には,ΩEye の ORiN オプション(有償)の購入が必要となります.

※ORiN オプションの詳細, その他 ΩEye に関する内容は, 下記にお問い合わせください.
・SOFIXCAN ΩEye 公式 ： sofixcan-omega-eye@sofix.co.jp

【改版履歴】

バージョン	日付	内容
1.0.0	2024-07-17	初版

【対応機種】

機種	バージョン	注意事項
SOFIXCAN ΩEye for Windows	3.1E	Rev.3.1E 以降のバージョンのみの対応となります.

【動作確認機種】

機種	バージョン	注意事項
SOFIXCAN ΩEye for Windows	3.1E	その他のバージョンでの動作は保証しません.

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. ΩEye 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 8
2.2.1. ΩEye プロバイダの手動インストール 8

3. コマンドリファレンス 9

3.1. メソッド/プロパティー覧 9
3.2. メソッド・プロパティ 9
3.2.1. CaoWorkspace クラス 9
3.2.1.1. AddController メソッド 9
3.2.2. CaoController クラス 10
3.2.2.1. GetVariableNames メソッド 10
3.2.2.2. Variables プロパティ 11
3.2.2.3. AddVariable メソッド 11
3.2.3. CaoVariable クラス 11
3.2.3.1. Value プロパティ 11
3.3. 変数一覧 12
3.3.1. システム変数とユーザー変数 12
3.3.2. CaoController クラス変数 12
3.3.2.1. @MAKER_NAME 13
3.3.2.2. @VERSION 13
3.3.2.3. @STATUS 13
3.3.2.1. @SYSTEM_INFO 14
3.3.2.2. LAST_RESULT<??> 15
3.3.2.3. RESULTS<??> 15

4. ΩEye プロバイダによるプログラミング 17

4.1. 最新の認識結果データを取得するサンプルプログラミング 17
4.1.1. サンプルプログラム 18
4.1.1.1. オブジェクト作成 19

4.1.1.2. 最新の認識結果を取得 20
4.1.1.3. オブジェクト削除 21
5. ΩEye プロバイダエラーコード 22
付録 A. リクエスト URI 対応表 23
付録 B. プロファイルについて 23

1. はじめに

本書は，ソフィックス社の SOFIXCAN ΩEye デバイスに対してデータのアクセス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 ΩEye プロバイダと呼称します．

(※ΩEye プロバイダは，ΩEye Windows 版のみの対応です．ΩEye ハードウェア版については非対応となります．)

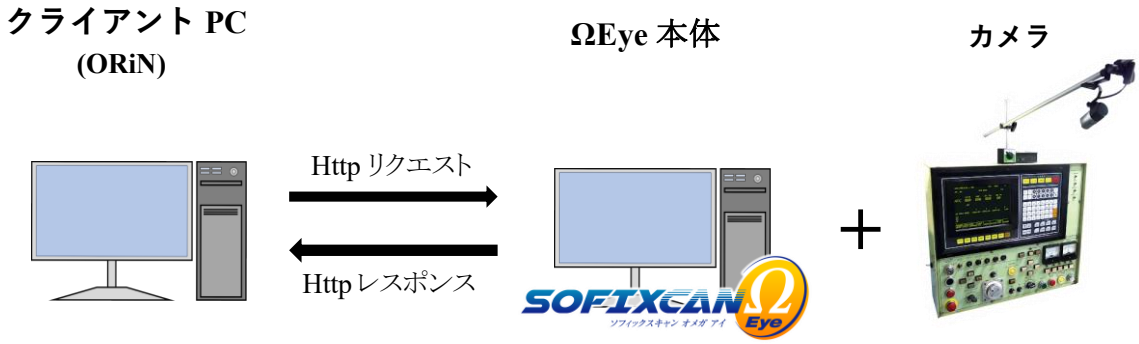


図 1-1 構成図

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．)

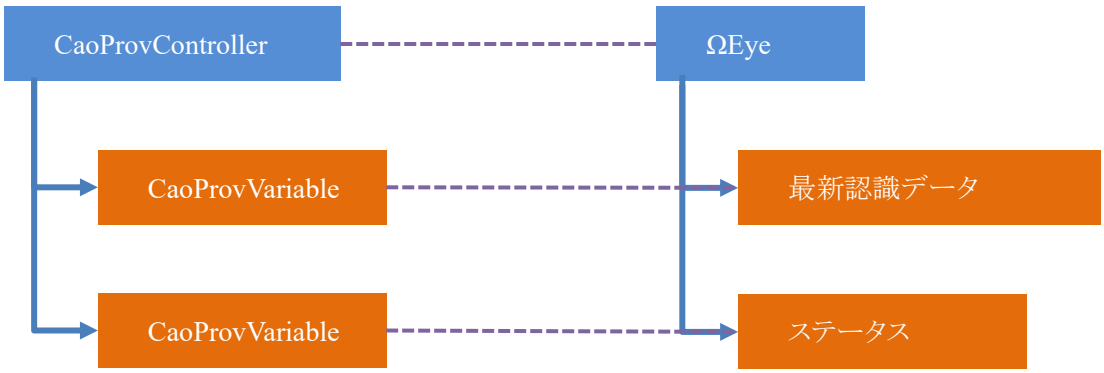


図 1-2 プロバイダの構成とデバイス情報との対応図

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. ΩEye とクライアント PC との接続

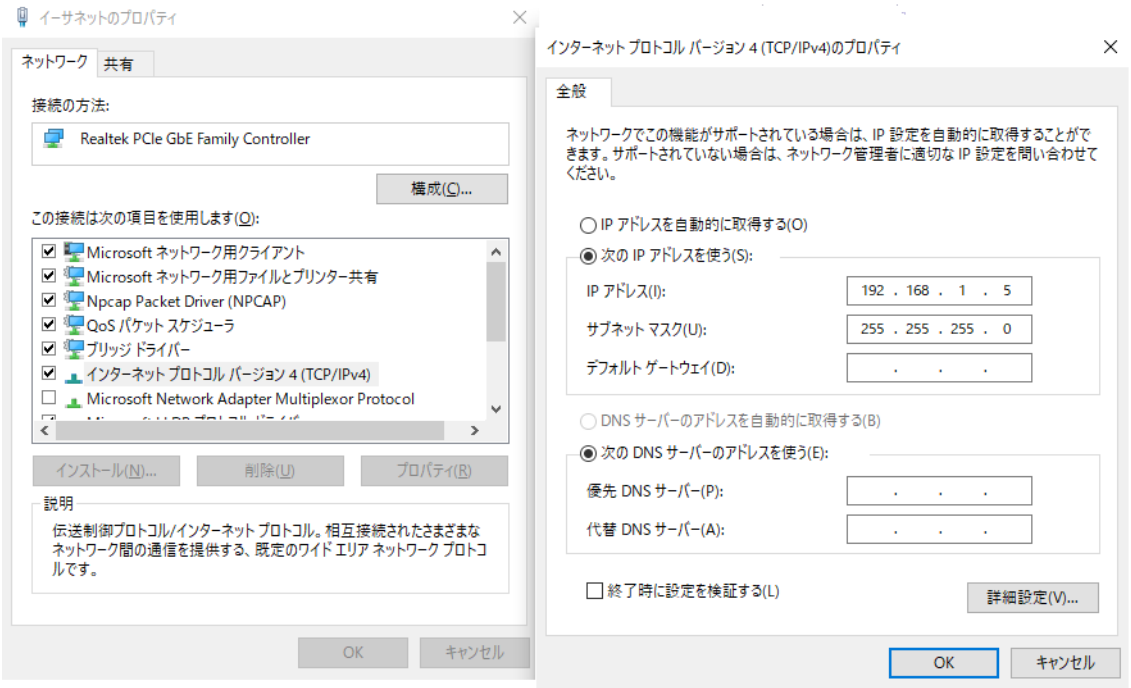
ΩEye と ΩEye プロバイダは ΩEye WebAPI を利用して通信します. ΩEye WebAPI は ΩEye が作成した認識データを, HTTP 通信により外部システムから閲覧取得するための機能です.

下記の手順で ΩEye とクライアント PC を接続してください.

1 SOFIXCAN Ω Eye for Windows インストールマニュアルの 7 章，8 章，9 章を実行し，WebAPI 機能を有効にします.

2 ΩEye をインストールした PC(ΩEye PC)の IPv4 の設定を確認，もしくは，変更します.

3 クライアント PC の IP アドレスを ΩEye PC に設定した IP アドレスと通信できるものに設定します.



※ ΩEye プロバイダはネットワークアダプタの選択や，プロキシの使用有無を指定することは出来ません．クライアント PC の設定に従いプロバイダが自動で決定します．想定外のネットワークアダプタが使用される，プロキシが使用されて通信できないなどが発生した場合は，クライアント PC の設定を見直してください.

2.2. PC 開発環境のセットアップ

2.2.1. ΩEye プロバイダの手動インストール

ΩEye プロバイダを使用するには表 2-1 の方法で登録を行う必要があります．RegistAsm.bat および UnregistAsm.bat は ORiN2SDK をインストールしたフォルダの下の DotNet¥BAT フォルダにあります．

表 2-1 ΩEye プロバイダ

ファイル名	CaoProvSOFIXSOFIXCANOmegaEyeV3.exe
ProgID	CaoProv.SOFIX.SOFIXCAN.OmegaEye.V3
レジストリ登録[1]	RegistAsm.bat CaoProvSOFIXSOFIXCANOmegaEyeV3.exe
レジストリ登録の抹消	UnregistAsm.bat CaoProvSOFIXSOFIXCANOmegaEyeV3.exe

[1] ORiN SDK でインストールした場合は手動で登録／抹消する必要はありません．

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[2]		機能	参照
CaoWorkspace				
	AddController	M	コントローラに接続	P.9
CaoController				
	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.11
	AddVariable	M	変数オブジェクトの追加	P.11
CaoVariable				
	Value	P	値の取得	P.11

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．ΩEye プロバイダでは，AddController メソッド実行時には接続は行いません．CaoVariable クラスの Value プロパティの値取得時に，渡されたパラメータを参照し，該当する ΩEye の Web サーバーへアクセスします．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.SOFIX.SOFIXCAN.OmegaEye.V3",       // 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
)
```

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．

[2] 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	必須	説明	値範囲	デフォルト値
Uri	○	接続先である ΩEye へのアクセス URL を指定します. アクセス URL に関しては, 3.2.1.1.1.アクセス URL を参照してください.	-	-
Timeout	-	応答待機時間を指定します.	0 - 600000	2000

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(“OmegaEye”,
                                                    "CaoProv.SOFIX.SOFIXCAN.OmegaEye.V3",
                                                    "",
                                                    "Uri = http://192.168.0.1/OMEGA-EYE-API");
```

3.2.1.1.1. アクセス URL

ΩEye にアクセスする URL は, ΩEye 本体の IP アドレスの後に"OMEGA-EYE-API"を指定する必要があります. 詳細は, 以下の表を参照してください.

表 3-2 アクセス URL

OS	アクセス URL 例
Windows 版	"http://192.168.0.1/OMEGA-EYE-API"

3.2.2. CaoController クラス

3.2.2.1. GetVariableNames メソッド

接続可能な変数名リストを取得します. 使用可能な変数は, 3.3 変数一覧を参照ください.

書式

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

使用例(C#)

```
// 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(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];
```

3.2.2.3. 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 3.3.2 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(省略可能)
)
```

3.2.3. CaoVariable クラス

3.2.3.1. Value プロパティ

ΩEye と接続し，データを取得します．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するための変数です. 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.

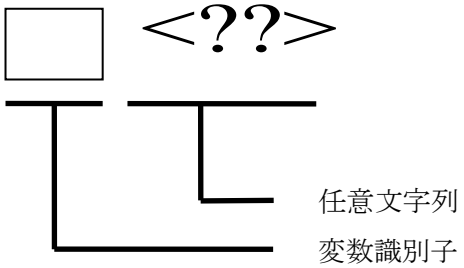
例)プロバイダバージョン, デバイス製造元, 認識アプリの認識処理状態

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. ユーザー変数はその性質上, 複数変数を登録(オプションのみ変更したい場合等に有用)するために変数識別の後に任意の文字列を付与することが可能です.

変数名に任意文字列を付与するための書式を以下に示します.

複数変数共通指定書式



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.13
@VERSION	ΩEye プロバイダのバージョンを取得します.	○	-	P.13
@STATUS	認識アプリの状態を取得します.	○	-	P.13
@SYSTEM_INFO	認識アプリのバージョン情報などを取得します.	○	-	P.14
LAST_RESULT	直近の認識結果データを取得します.	○	-	P.15
RESULTS	指定した件数分の認識結果を, 直近のデータから遡って取得します.	○	-	P.15

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

ΩEye プロバイダのバージョンの取得をします.

データ型

型説明	
VT_BSTR	ΩEye プロバイダ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. @STATUS

認識アプリの認識処理状態を取得します.

データ型

型説明	
VT_BSTR	認識アプリの認識処理状態を取得します. 以下に取得した値の意味を示します. "active" : 認識アプリが起動し認識処理を行っている状態. (撮影方式は「タイマー式」) "stopped" : 認識アプリは起動しているが認識処理は停止している状態. (撮影方式は「タイマー式」) "inactive" : 認識アプリが起動していない状態. "camera_ready" : 撮影方式が「外部トリガー」で起動し, ワンショット認識要求を待ち合わせている状態. “invalid_license” : ORiN オプションが有効ではない. ※ΩEye の ORiN オプションが無効状態のため, オプションの有効化が必要となります. 詳細は下記にお問い合わせください. ・SOFIXCAN ΩEye 公式 : sofixcan-omega-eye@sofix.co.jp

使用例(C#)

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

3.3.2.1. @SYSTEM_INFO

認識アプリのバージョン情報などを取得します.

データ型

型説明		
VT_ARRAY | VT_BSTR		
	0	ソフトウェアバージョンを取得します. *.*.*
	1	設定アプリのカメラ設定画面で設定した個体名称を取得します.

使用例(C#)

```
// 変数追加
ORiN2.ManagedCAO.CCaoVariable var = controller.AddVariable("@SYSTEM_INFO","");
// 値取得
string[] values = var.Value as string[];
// ソフトウェアバージョン
var version = values[0];
```

```
// 個体名称
var name = values[1];
```

3.3.2.2. LAST_RESULT<??>

直近の認識結果データを取得します. LAST_RESULT の後に任意の文字列を入力して変数名を指定してください.

オプション

オプション	必須	説明	値範囲	デフォルト値
SettingId	-	認識結果データのプロファイル番号を指定します. プロファイルの詳細は「プロファイルについて」を参照してください.	0 - 9	0

※ 認識結果データは"measurement_0","measurement_1"というようなプロファイル番号毎のディレクトリ以下に公開されます.

データ型

型説明	
VT_ARRAY | VT_BSTR	認識結果データの CSV 一行分をカンマ区切りで分割し, 配列データとして格納します.

使用例(C#)

```
// 変数追加
ORiN2.ManagedCAO.CCaoVariable var = controller.AddVariable("LAST_RESULT1"," SettingId = 0");
// 値取得
string[] values = var.Value as string[];

for (var i = 0; i < values.Length; i++)
{
    // カンマ区切りのデータ
    var splitData = values[i];
}
```

3.3.2.3. RESULTS<??>

指定した件数分の認識結果を, 直近のデータから遡って取得します. RESULTS の後に任意の文字列を入力して変数名を指定してください.

オプション

オプション	必須	説明	値範囲	デフォルト値
SettingId	-	認識結果データのプロファイル番号を指定します. プロファイルの詳細は「プロファイルについて」を参照してください.	0 - 9	0
Count	-	取得するデータ件数を指定します.	1 - 2147483647	1

※ 認識結果データは"measurement_0","measurement_1"というようなプロファイル番号毎のディレクトリ以下に公開されます.

データ型

型説明		
VT_ARRAY | VT_VARIANT		認識結果データの履歴一覧を取得します. 先頭の行は最新データとなります.
i	VT_ARRAY VT_BSTR	認識結果データの CSV 一行をカンマ区切りで分割し, 一行ごとに配列データとして格納したジャグ配列として CSV を格納します.

※ **i**: データの履歴数分

使用例(C#)

```
// 変数追加
ORiN2.ManagedCAO.CCaoVariable var = controller.AddVariable("RESULTS1", " SettingId = 0,Count = 1");

// 値取得
object[] results = var.Value as object[];

for (int i = 0; i < results.Length; i++)
{
    // 認識結果データ
    string[] resultData = results[i] as string[];

    for (int j = 0; j < resultData.Length; j++)
    {
        // カンマ区切りのデータ
        string splitData = resultData[j];
    }
}
```

4. ΩEye プロバイダによるプログラミング

ΩEye プロバイダでは, 以下の手順でクライアント PC と ΩEye を接続することができます.

- CaoEngine の作成
- CaoWorkspace の作成
- CaoController の作成
- CaoVariable の作成
- CaoVariable の Value プロパティの取得実行

4.1. 最新の認識結果データを取得するサンプルプログラミング

ここでは例として ΩEye の最新の認識データを取得す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	Http 通信で接続する
	接続先 IP アドレスは 192.168.0.1
	ΩEye は Windows 版
処理内容	ΩEye の最新の認識結果を読み込む

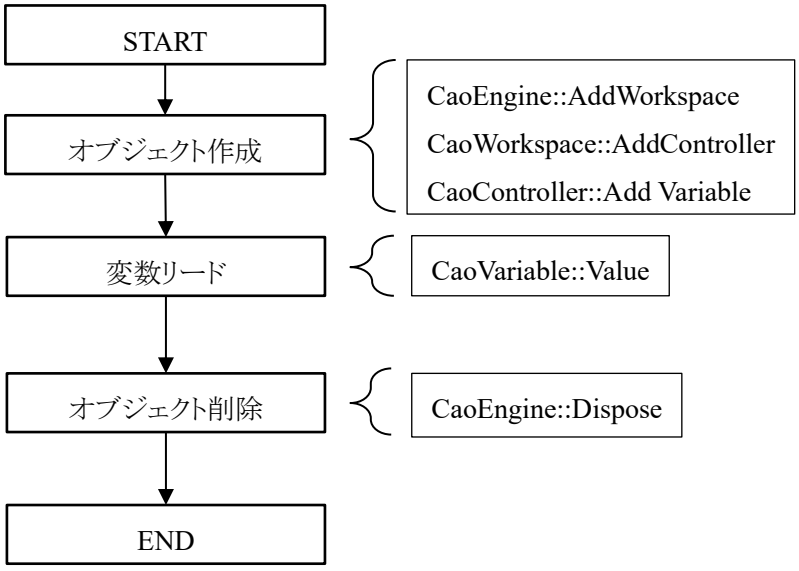


図 4-1 処理の流れ

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

```
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_varLastResult = null;

public void Main()
{
    // オブジェクト作成
    this.CreateObject();

    // 変数リード
    string[] values = this.m_varLastResult.Value as string[];

    for (var i = 0; i < values.Length; i++)
    {
        // カンマ区切りの各データを取得
        var splitData = values[i];
    }

    // オブジェクト削除
    this.DeleteObject();
}

// オブジェクト作成メソッド
private void CreateObject ()
{
    // 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("OmegaEye",
                                                             "CaoProv.SOFIX.SOFIXCAN.OmegaEye.V3",
                                                             "",
```

```
                                                    "Uri = http://192.168.0.1/OMEGA-EYE-API");
    // CaoVariable オブジェクトの生成
    this.m_varLastResult = this.m_caoController.AddVariable("LAST_RESULT1"," SettingId = 0");
}

  // オブジェクト削除メソッド
private void DeleteObject ()
{
    this.m_caoEngine.Dispose();
    this.m_caoEngine = null;
}
```

4.1.1.1. オブジェクト作成

データを取得するために前準備として，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．
データ取得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_varLastResult = 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. ΩEye プロバイダでは, 接続先の ΩEye を指定するために ΩEye サーバーの URL 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("OmegaEye",
                                                          "CaoProv.SOFIX.SOFIXCAN.OmegaEye.V3",
                                                          "",
                                                          " Uri = http://192.168.0.1/OMEGA-EYE-API ");
```

(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, 変数名と使用するためのパラメータを設定します. ここで最新の認識結果を取得するために変数名に"LAST_RESULT1", パラメータはデフォルト値を使用するため、未指定で指定します. 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_varLastResult = this.m_caoController.AddVariable("LAST_RESULT1","");
```

4.1.1.2. 最新の認識結果を取得

ΩEye から最新の認識結果を取得します. ΩEye プロバイダは変数リード時に ΩEye との接続を行い, 値を取得します. 以下にコード例を示します.

使用例(C#)

```
// 変数リード
string[] values = this.m_varLastResult.Value as string[];
for (var i = 0; i < values.Length; i++)
```

```
{
    // カンマ区切りの各データを取得
    var splitData = values[i];
}
```

4.1.1.3. オブジェクト削除

後処理として，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;
```

5. ΩEye プロバイダエラーコード

本プロバイダには, 0x8011****でマスクした以下の独自エラーコードが存在します. (表 5-1 独自エラーコード表参照)

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.

表 5-1 独自エラーコード表

エラー番号	説明
0x80110001	必須オプションが指定されていません.
0x80110002	解析できないオプションが指定されました.
0x80110003	値範囲外のオプションが指定されました.
0x80111001	ΩEye からの異常応答が発生しました.
0x80111002	タイムアウトエラーが発生しました.
0x80110301	未対応の変数が指定されました.
0x80110302	Set 動作未対応変数で Set 動作が行われました.

また, 本プロバイダは, HTTP のレスポンドコードを「0x8010****」でマスクして返します.

レスポンスコードの内容については SOFIXCAN ΩEye WebAPI 仕様書を参照してください.

付録A. リクエスト URI 対応表

変数名	get_Value	put_Value
@STATUS	get_state_orin	---
@SYSTEM_INFO	get_system_info_orin	---
LAST_RESULT	get_latest_result_orin	---
RESULTS	get_results_orin	---

付録B. プロファイルについて

認識に使用する設定情報をまとめたファイルをプロファイルと呼びます. ΩEye Windows 版はこのプロファイルを最大 10 個保持し, 各プロファイル内で『カメラ選択』『チェックポイント設定』などを行います.