CaoOPC (OPC Server)

ユーザーズ ガイド

Version 1.5.2

March 12, 2021

【備考】
OPC DA (Data Access) 2.0 の仕様に準拠しています. OPC DA 1.0 仕様も実装されていますがサポート外です.

【改版履歴】

日付	版数	内容
2006-02-24	1.0	初版作成.
2008-01-09	1.1	IDataObject (DA 1.0)を追加.
2010-04-13	1.2	IOPCBrowseServerAddressSpace に対応. 3 CaoOPCConfig を Ver.1.1.9 の内容に修正.
2013-07-30	1.3	アイテムの初期化（初期値設定）機能を追加.
2016-07-26	1.4	同じ親オブジェクトへの参照を持つ変数の対応追加.
2018-05-29	1.5	I8, UI8 型対応.
2019-02-14	1.5.1	ログ出力追加.
2021-03-12	1.5.2	メモリリーク バグ修正.

【依存モジュール】

No.	依存モジュール
1	OPC Core Components 2.00 (OPC Core Components 2.00 Redistributable 2.30.msi) 上記ファイルは Bin フォルダに格納されています。

目次

1. はじめに

本書は CAO(Controller Access Object)のクライアントアプリケーションとして位置付けられる CaoOPC のユーザーズガイドです.

CaoOPC は CAO モジュールを 1 つの FA 機器とみなし, OPC(OLE for Process Control)の機能を提供する OPC サーバです(図 1-1). CAO は様々な FA 機器との通信を可能にするモジュールなので, その上のアプリケーションである CaoOPC も当然様々な FA 機器に接続して使うことができます.

また, CaoOPC は OPC サーバの機能を提供するモジュールなので, 市販されている OPC クライアントソフトウェアを活用することもできます. このことは「CAO プロバイダを実装すれば, CAO クライアントアプリケーションに加えて, OPC クライアントアプリケーションも使える」ことを意味し, よりバラエティに富んだシステム開発が可能になります. まさに, これが CaoOPC の目的です.

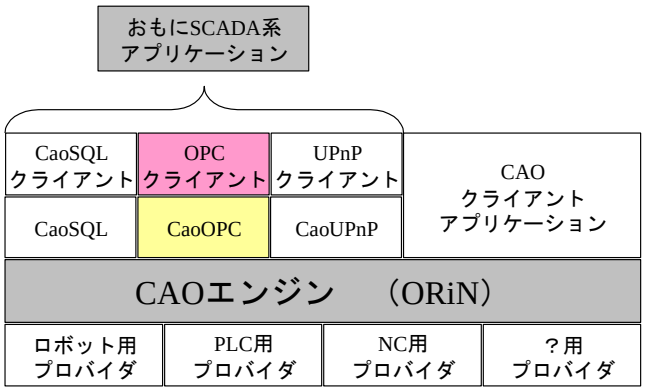

図 1-1 CAO アプリケーションの形態

OPC は主として「プロセスデータをクライアントに提供するサーバソフトウェア」として位置付けられており, CAO の「FA 機器が持つ全てのリソースをモデル化する」という位置付けとは異なります. したがって, CaoOPC は主に CAO の変数クラスだけを使って実装されています. このことから, 必然的に CaoOPC を使ったアプリケーションは SCADA 系アプリケーションに限定されます.

しかし, 逆に SCADA 系アプリケーションを開発する場合は, CaoSQL や CaoOPC を使うと CAO を直接使うのに比べて非常にシステム構築が容易になります. CaoSQL は SCADA 系アプリケーションに必須のデータベースアクセスの機能など高度な機能を提供し, CaoOPC は市販の OPC クライアントソフトウェアを活用できるようにします.

本書では, 主に CaoOPC の使い方を解説します. OPC の仕様については, OPC Foundation のホームページ などを参照して下さい. また, CaoOPC とは逆に OPC サーバを CAO クライアントアプリケーションから使う場合は,「OPC プロバイダユーザーズガイド 」を参照して下さい.

2. CaoOPC の概要

2.1. 概要

CaoOPC の構成を下図に示します. CaoOPC は OPC クライアントと CAO を接続するミドルウェアであり, OPC クライアントからの要求を CAO の機能を使って処理する OPC サーバです. CaoOPC は1つの独立したプロセス(アウトプロセスサーバ)で, 複数の OPC クライアントに対してサービスを提供します.

CaoOPC ver.1.0 は OPC DA(Data Access) 2.0 の仕様に対応しているので, DA2.0 の機能を使う OPC クライアントも使うことができます. CaoOPC の ProgID は“OPC.CAO”です.

CaoOPCConfing ツールは, CaoSQLConfig を利用し, CAO の CaoController オブジェクト, CaoVariable オブジェクトを作成するのに必要なパラメータ, CaoOPC の動作オプションを設定するツールです. このツールで設定された内容は, レジストリと csq ファイルに保存され, CaoOPC の起動時に参照します.

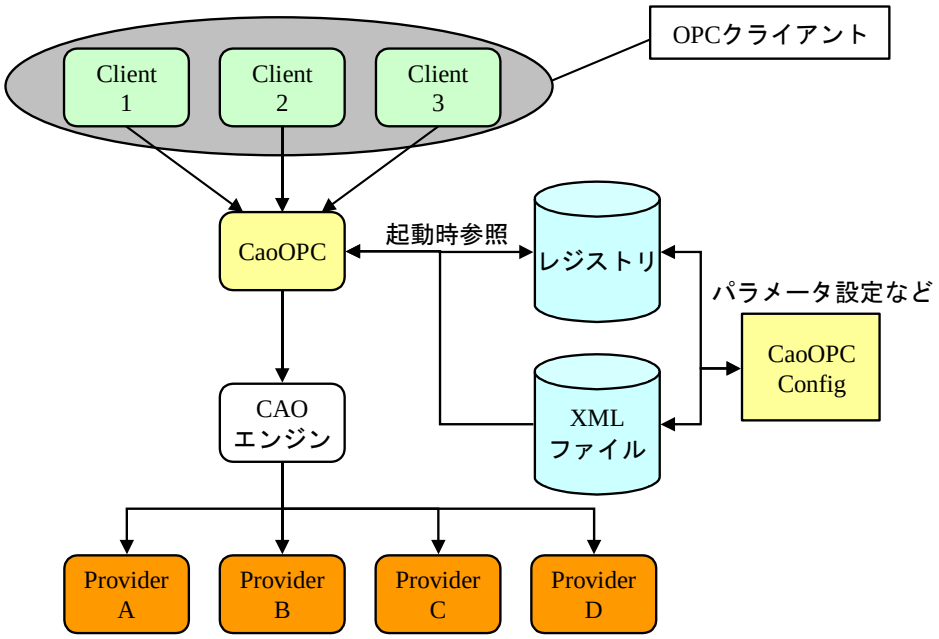

図 2-1 CaoOPC 概要図

ここで，例えば Provider として OPC 用 CAO プロバイダを使った場合，OPC クライアントと OPC サーバの間に本システムが挿入されることになります(図 2-2). 前述したように，OPC クライアントが CaoOPC を使うメリットとしては，CAO プロバイダが提供するロボットなど様々な FA 機器との接続が可能となることです. しかし，OPC 用 CAO プロバイダを使ってその 2 つの間に挿入した場合でも，「インターネットを介した OPC サーバとの接続」が可能になるというメリットがあります. これは，ORiN の CAP(Controller Access Protocol)の技術を利用した接続形態です.

このように，単に OPC クライアントと OPC サーバの間に本システムを挿入するというだけの使い方でも ORiN の技術を活用できるというメリットがあります. (さらに言えば，CRD プロバイダを使えば，CRD(XML)形式のファイルにも OPC クライアントからアクセスできるようになります.)

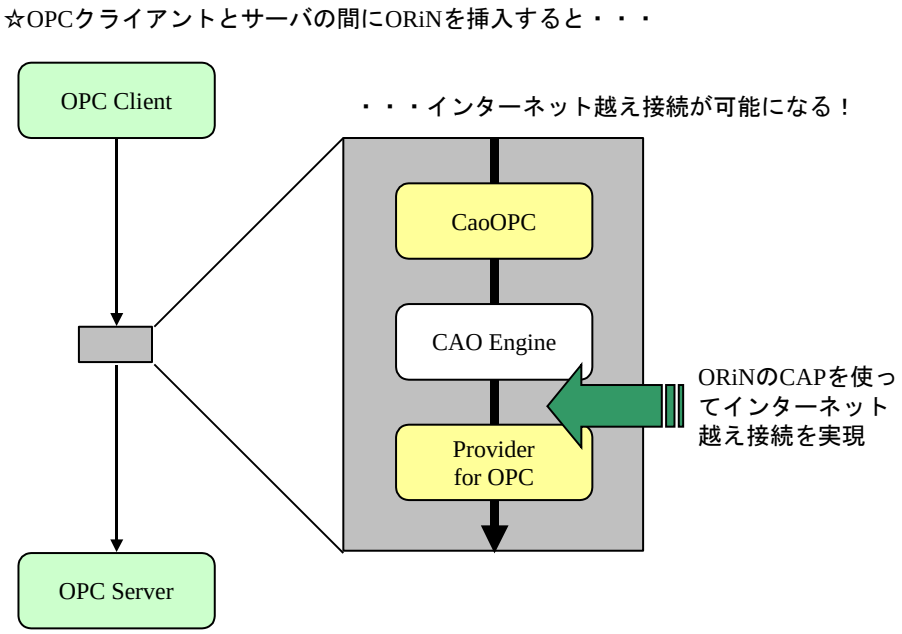

図 2-2 OPC クライアントとサーバ間に挿入

2.2. OPC インタフェース サポート状況

OPC の DA2.0 カスタムインタフェースのサポート状況を表 2-1 に示します.

表 2-1 OPC インタフェースサポート状況

OPC インタフェース	サポート状況	備考
+ OPC サーバオブジェクト		
IOPCCommon	○	
IOPCServer	○	
IOPCServerPublicGroups	×	Option
IOPCBrowseServerAddressSpace	△	Option[1]
IOPCItemProperties	○	[2] [3]
IConnectionPointContainer	○	
IPersistFile	×	Option
+ OPC グループオブジェクト		
IOPCGroupStateMgt	○	
IOPCPublicGroupStateMgt	×	Option
IOPCSyncIO	○	
IOPCAsyncIO	○	Old Interface
IOPCItemMgt	○	
IDataObject	○	Old Interface
IOPCAsyncIO2	○	
IConnectionPointContainer	○	
+ OPC アイテム列挙オブジェクト		
IEnumOPCItemAttributes	○	

[1] BrowseAccessPath メソッドのみ, 未対応です. また, BrowsOPCItemIDs メソッドの文字列フィルタには, ***?**のみ使用可能です.
[2] デフォルトコントローラ のアイテム, もしくは フルパス表記 のアイテムのみ可能です.
[3] 使用可能なプロパティ ID は「2.7. プロパティ ID 一覧 」を参照してください.

2.3. データタイプ サポート状況

CaoOPC は，VARIANT 型で CAO から値を取得しますので，配列を含めて対象とするプロバイダが扱えるデータ型をそのまま取得することができます．ただ，CaoOPC 内部で値変化のイベントを発生させる際に値が変化したか否かの判定に，VarComp 関数(Automation API)を使っていますので，その時点で配列は正しく判定されません．

また，OPC 仕様ではクライアントが取得データ型を指定できますので，CaoOPC 内部で型変換させる必要がありますが，その処理に VariantChangeType 関数(Automation API)を使っていますので，この時点でも配列型を正しく変換することはできません．

2.4. アイテム ID とアクセスパス

CaoOPC は AccessPath をサポートしています．AccessPath は CaoOPCConfig ツールで設定されるコントローラの Name(CaoOPC コントローラ名)[4]が用いられます．AccessPath の指定の仕方は，通常の IOPCItemMgt::AddItems 関数での指定に加えて，下記のようにアイテム ID に含めることができます．

アイテム ID = [<AccessPath>!]<ItemID>

例えば，AccessPath が“PLC1”で，ItemID が“W100”の場合，“PLC1!W100”と表現することができます．この例では CaoOPC は“W100”を CaoVariable の変数名として使用します．しかし，元々 CaoVariable の変数名にデリミタ“!”が含まれている場合も考えられますので，特に理由がない場合は AccessPath は AddItems 関数で指定して下さい．[5]

アイテム ID は CaoOPCConfig で設定したものの他に，動的に追加することもできます．この場合，アイテムのネイティブ型は VT_VARIANT になり，クラスはコントローラクラスになります．[6]

2.5. 動作モデル

CaoOPC の動作モデルを図 2-3 に示します．プロセス内部では，アイテムリストに登録されたアイテムの値を UpdateRate で指定された周期で常に CAO から読込んでキャッシュしています．ここで，取得した値とキャッシュした値が異なる場合はキャッシュの値を更新し，OnDataChange イベントを発生させます．[7]

通常 OPC_DS_CACHE オプションで Read するとキャッシュされた値を取得しますが，OPC_DS_DEVICE オプションで Read した場合は，UpdateRate の周期とは関係なく Read 関数呼び出した時点で CAO から読込みます．ただし，共にアイテムが Active 状態でない場合は更新されません．

また，逆に値を書込む場合は，クライアントが Write 関数を呼び出した時点で CAO に対して書き込みを行います．

[4] Controller Name(Cao コントローラ名)ではないことに注意して下さい．
[5] デリミタ“!”を含むアイテム ID を指定する場合，コントローラ名の省略(デフォルトコントローラ名)は使用できません．
[6] 動的に追加したアイテムは，CaoOPC 終了時に開放されます．
[7] 値変更判定に VarCmp を使用しているため，VarCmp が認識できない型(VT_I1, VT_UI2,VT_UI4 等)では OnDataChange イベントは発生しません．

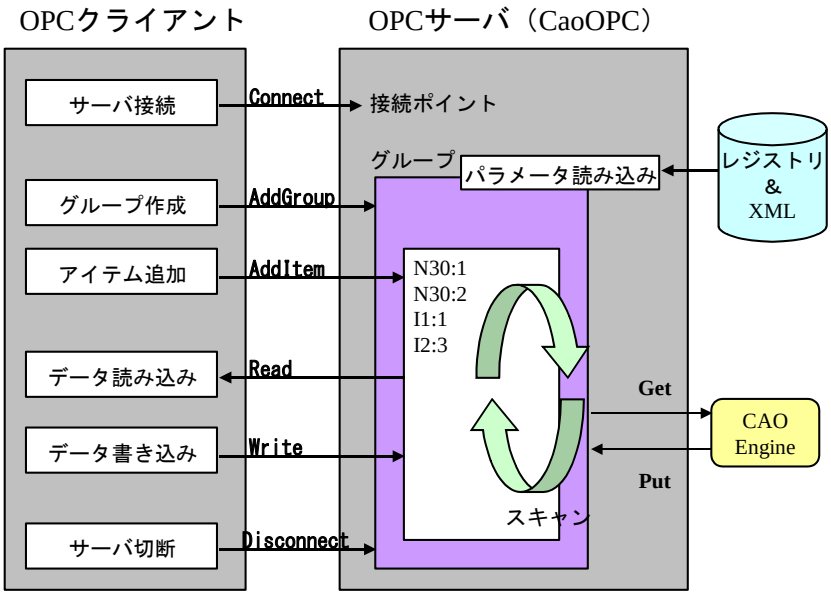

図 2-3 CaoOPC ワーキングモデル

2.6. LocaleID

LocaleID とは言語を表す DWORD 型の整数であり，主なものに表 2-2 あります.

表 2-2 主な LocaleID

ロケール名	言語コード	16 進	10 進
日本語	ja	0x0411	1041
英語(U.S.)	en-us	0x0409	1033
ドイツ語(ドイツ)	de	0x0407	1031

CaoOPC の LocaleID は CaoOPCConfig ツールでデフォルト値を設定することができます. また，IOPCCommon::SetLocaleID 関数でも設定することができます.

IOPCCommon::QueryAvailableLocaleIDs 関数では常に 3 要素のリストを返し，その値は LOCALE_SYSTEM_DEFAULT(システムのデフォルトロケール ID:2), LOCALE_USER_DEFAULT(ユーザのデフォルトロケール ID:1), LOCALE_NEUTRAL(0)になります.

OPC サーバから Group オブジェクトを作成するときは，IOPCServer::AddGroup 関数の引数で指定した値がそのグループのデフォルト値になります. この値も，IOPCGroupStateMgt::SetState 関数で再設定することもできます. このグループのメンバであるアイテムの値の変化検出時にこのグループの LocaleID が使われます[8].

[8] 具体的には，Platform SDK の Automation ライブラリ(oleaut32.lib)の VarCmp 関数の第 3 引数で使われています.

2.7. プロパティ ID 一覧

以下に IOPCItemProperties メソッドで取得することのできるプロパティ ID を示します.
各プロパティについては OPC の仕様書を参考にして下さい.

表 2-3 プロパティ ID 一覧その 1

ID	データ型	
1	VT_I2	"Item Canonical DataType"
2	<varies>	"Item Value"
3	VT_I2	"Item Quality"
4	VT_DATE	"Item Timestamp"
5	VT_I4	"Item Access Rights"
6	VT_R4	"Server Scan Rate"

表 2-4 プロパティ ID 一覧その 2

ID	データ型	
100	VT_BSTR	"EU Units"　e.g. "DEGC" or "GALLONS"
101	VT_BSTR	"Item Description"
102	VT_R8	"High EU"
103	VT_R8	"Low EU"

2.8. サンプルプログラム

ここでは, CaoOPCを使った簡単なサンプルを示します. プログラミング言語はMicrosoft Visual Basic6.0で記述しています. この場合, OPC Automation ラッパーDLL が登録されていないと動作しないので注意して下さい.

(1) 参照設定の追加

まずは, OPC Automation ラッパーDLL の参照を追加します. メニューの「プロジェクト」→「参照設定」を選択してください.

次に「参照」ボタンを押下し, “OPCDAAuto.dll”を追加してください.

この DLL は, ORiN2¥CaoOPC¥Bin に含まれています.

図 2-4 参照設定の追加

(2) コードの記述

テキストボックスを 1 つ持つフォームを作成し，以下のソースコードを記述してください．

List 2-1 Form1.frm

```
Dim WithEvents objMyOpcGroup As OPCGroup

Private Sub Form_Load()
    Dim myOpcServer As OPCServer
    Dim myOpcItems As OPCItems
    Dim myOpcGroups As OPCGroups
    Dim myOpcGroup As OPCGroup

' connect server
    Set myOpcServer = New OPCServer
    myOpcServer.Connect "OPC.CAO"    ' CaoOPC の ProgID は"OPC.CAO"固定.

' add group
    Set myOpcGroups = myOpcServer.OPCGroups
    Set myOpcGroup = myOpcGroups.Add("MyGroup")
    myOpcGroup.UpdateRate = 10       ' 更新周期を 10ms に設定.
    Set myOpcItems = myOpcGroup.OPCItems
    Dim ItemServerHandles() As Long
    Dim ItemServerErrors() As Long
    Dim RequestedDataTypes(1) As Integer
    Dim AccessPaths(1) As String
    Dim ClientHandles(1) As Long
    Dim OPCItemIDs(1) As String

' add item
    OPCItemIDs(1) = "ItemA"     ' アイテム名を設定.
    ClientHandles(1) = 1
    RequestedDataTypes(1) = vbDouble
    AccessPaths(1) = "DS1"      ' コントローラリストから選択.
    myOpcItems.AddItems 1, OPCItemIDs, ClientHandles, ItemServerHandles, _
        ItemServerErrors, RequestedDataTypes, AccessPaths
    myOpcGroup.IsActive = True  ' 読み出し開始.
    myOpcGroup.IsSubscribed = True  ' イベントを有効に設定.
    Set objMyOpcGroup = myOpcGroup
```

```
End Sub

Private Sub objMyOpcGroup_DataChange(ByVal TransactionID As Long, _
        ByVal NumItems As Long, ClientHandles() As Long, _
        ItemValues() As Variant, Qualities() As Long, _
        TimeStamps() As Date)
    Text1.Text = ItemValues(1)        ' "ItemA"の値が変化するたびに更新される.
End Sub
```

このプログラムを実行すると，アイテムの値が変化するたびにイベントが発生し，テキストボックス内の値が変化します.

3. CaoOPCConfig

3.1. 概要

CaoSQLConfig の引数に“OPC”を付けて実行(“CaoSQLComfig.exe OPC”)することで CaoOPC 用 CaoSQLConfig(以下 CaoOPCConfig)を起動します. CaoOPCConfig は, マシン内の CaoOPC の動作パラメータ設定や, CAO のコントローラオブジェクト, CAO の変数オブジェクトを作成する際に必要なパラメータを設定するツールです. ここで設定した情報はレジストリとcsqファイルに記録され, CaoOPCの起動時に読込まれます.

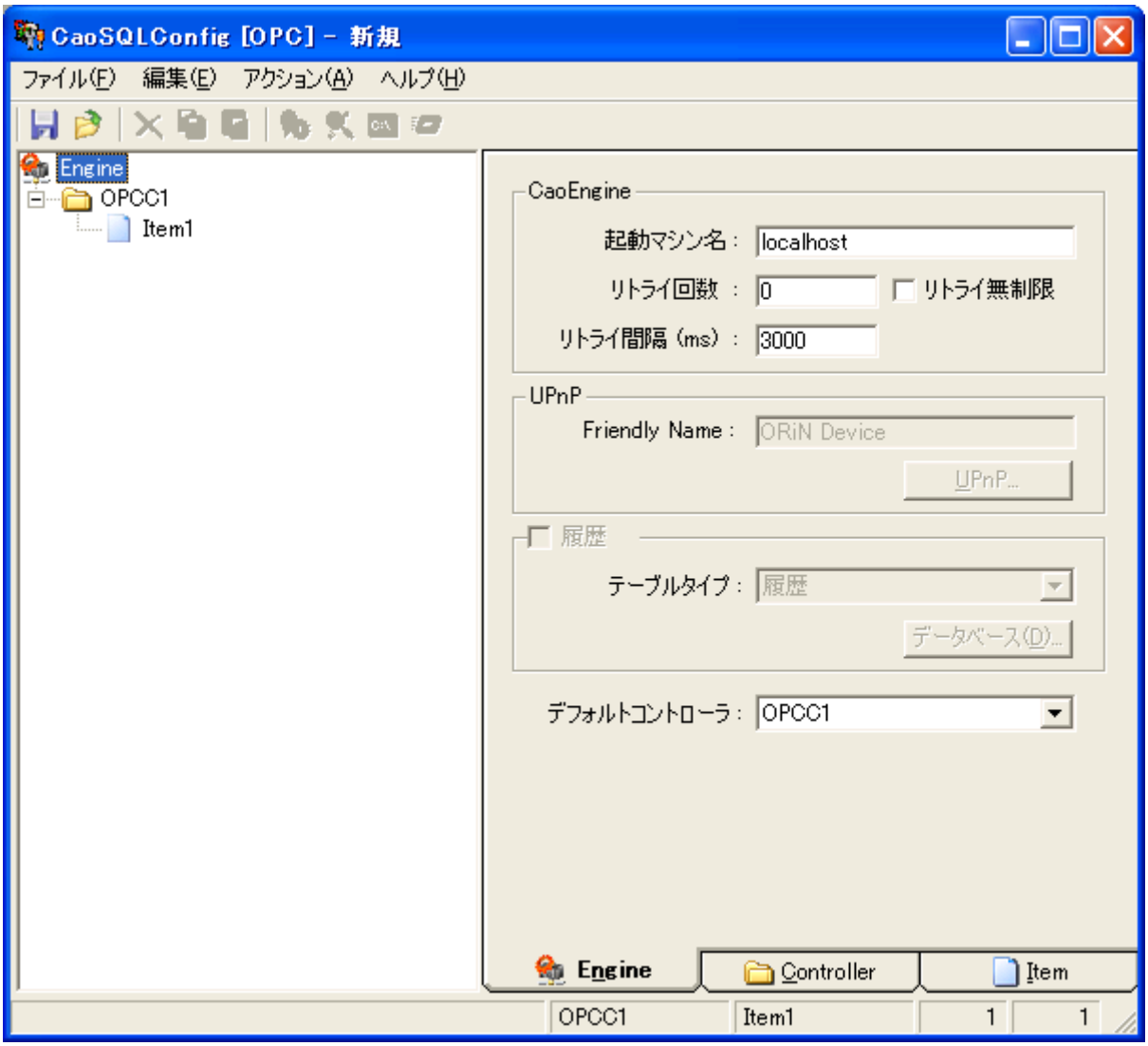

図 3-1 CaoOPCConfig 画面

3.2. 操作方法

3.2.1. タブ入力

画面右のタブはツリービューのノードを選択状態にすることによってその情報を示します．エンジンのノードを選択するとタブはエンジンの情報を表示します．コントローラのノードを選択するとタブはそのコントローラの情報を表示します．アイテムのノードを選択するとタブはそのアイテムの情報を表示します．

注意することとして，入力されたデータの整合性のチェックはおこなっていないため，実際に CaoOPC を起動した時にエラーが発生し，CaoOPC のリストから外れてしまう可能性がありますので CaoOPCConfig でデータを入力する時はデータの確認を慎重に行うことが必要です．

3.2.1.1. エンジンタブ

エンジンタブでは図 3-2 に示した形式で情報を設定・表示します．エンジンの情報は自由に設定できます．CaoOPC ワークスペース名はデフォルトの名前が適応され，またデフォルトコントローラ名に関しては現在存在しているコントローラ名のみ選択ができます．

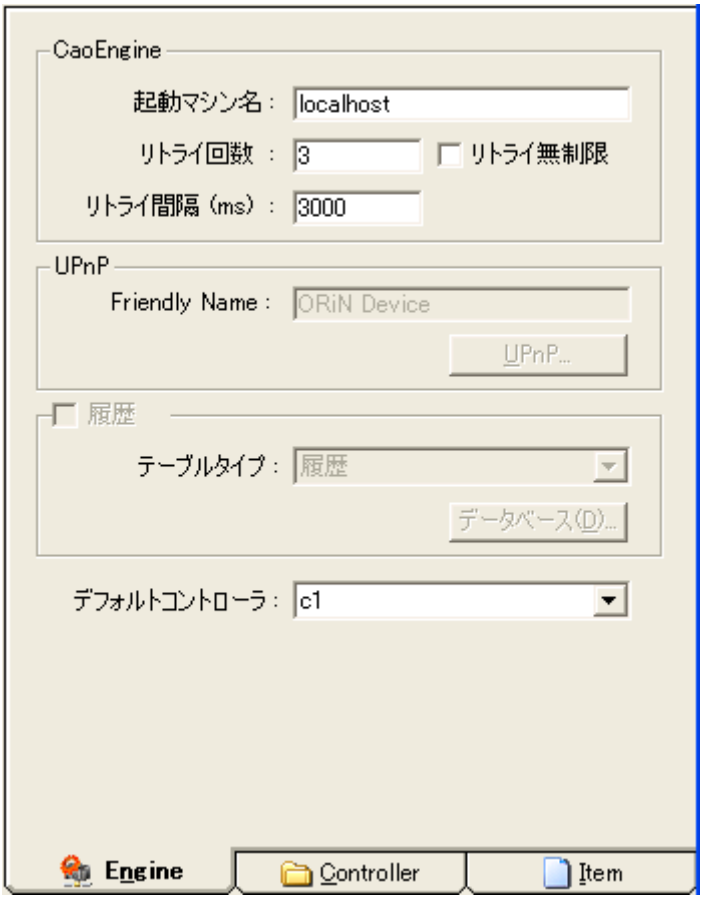

図 3-2 エンジンタブ

ここで設定可能な項目は以下の通りです.

[起動マシン名] – 起動マシン名

CAO.exe の起動コンピュータ名を入力します. 何も入力されていない場合はローカル起動します.

[リトライ回数, リトライ無制限] – リトライ回数, リトライ無制限

チェックボックスを入れると CaoController のオープンで失敗した場合, リトライすることができます. リトライ回数は無制限をチェックする, 又は回数を入力します.

[リトライ間隔 – リトライ間隔(ms)

リトライ間隔は一度失敗してから, リトライをするまでの間隔をミリ秒単位で入力します.

[デフォルトコントローラ] – デフォルトコントローラ

AccessPath を指定しなかった場合, ここで設定したコントローラが使用されます.

3.2.1.2. コントローラタブ

コントローラタブでは図 3-3 に示した形式で情報を設定・表示します.

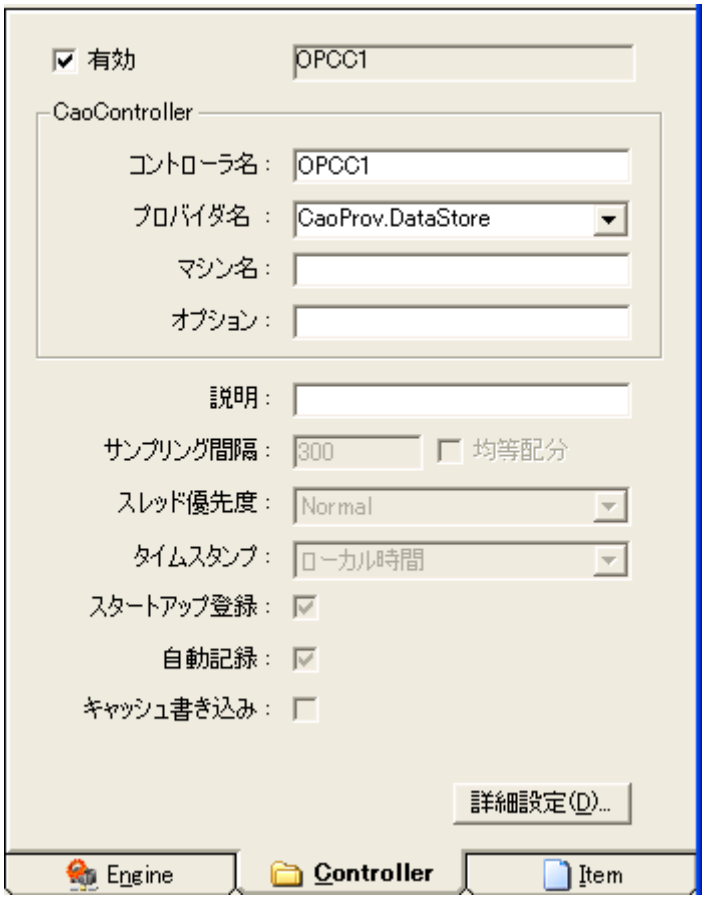

図 3-3 コントローラタブ

コントローラの情報は自由に設定できますが,「CaoOPCコントローラ名」だけは登録された情報の中でユニークな情報を持つことが要求される為, メニューの名前変更でしか変更できません. またコントローラ名に関しては大文字と小文字の区別はされず, 数値のみの名前は許可しません. また, 記号の内, ¥$#は使用できません.

ここで設定可能な項目は以下の通りです.

[コントローラ有効] – 有効

このコントローラの有効/無効設定です. 無効に設定された場合は CaoOPC 実行時に起動されません.

[CaoOPC コントローラ名]

CaoOPC コントローラ名を設定します.

このコントローラ名は AccessPath として使用されます.

[Cao コントローラ名, プロバイダ名, コンピュータ名, オプション]

– コントローラ名, プロバイダ名, マシン名, オプション

AddController をおこなう際に渡すパラメータです. 入力した値がそのまま使用されます.

[詳細情報] – 説明

コントローラに関する詳細な情報を設定できます. XML ファイルから登録されるので, 不揮発な情報を設定するのに向いています.

3.2.1.3. アイテムタブ

アイテムタブでは図 3-4 に示すような情報を設定・表示する. コントローラタブと同様にアイテム名のみはメニューの名前変更でないと変更は行えません. またアイテム名に関しては大文字と小文字の区別はされず, 数値のみの名前は許可しません. また, 記号の内, **¥$#:!** は使用できません.

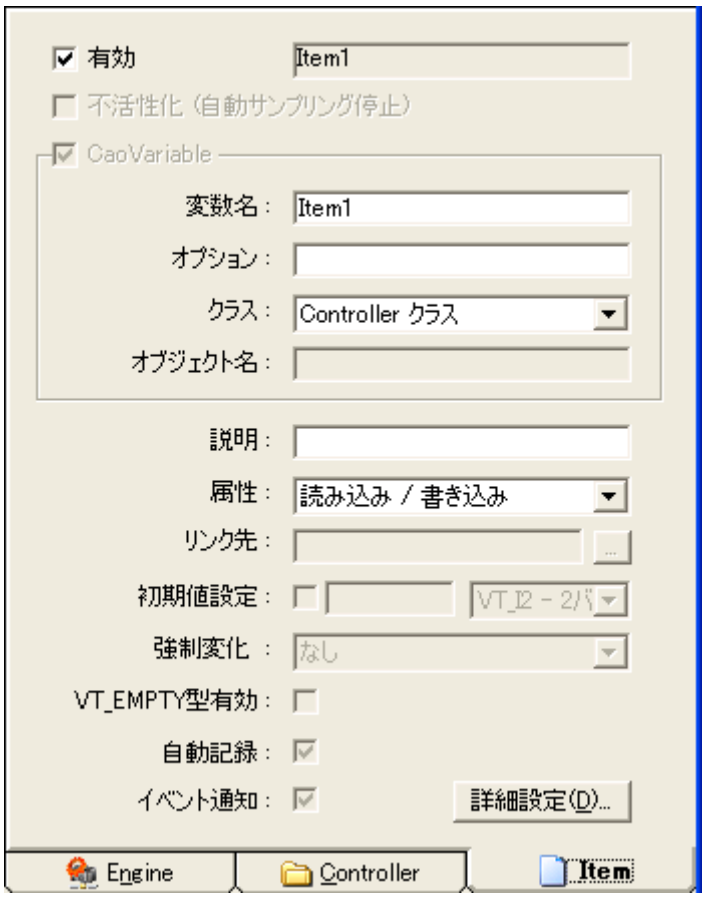

図 3-4 アイテムタブ

ここで設定できる項目は以下の通りです.

[アイテム有効] – 有効

このアイテムを有効/無効にする設定です. 無効に設定された場合はサンプリングされません.

[アイテム名] – 名前

アイテム名を設定します. このアイテム名は ItemID として使用されます.

[変数名] – 変数名

CAO の変数オブジェクト名を指定します. 大文字・小文字は区別されます. CaoVariable オブジェクトの取得(ICaoController::GetVariable)時に使用されます.

[オプション] – オプション

CaoVariable オブジェクトの作成時に付加したいオプションを指定します. CaoVariable オブジェクトの取得(ICaoController::GetVariable)時に使用されます.

[クラス] – クラス

変数クラスの種類を選びます. CaoVariable オブジェクトの取得(ICaoController::GetVariable)時に使用されます.

[属性] – 属性

アイテムの属性を設定します.

[初期化設定] – 値の初期値設定

アイテムの値に初期値を設定します. 実機に対してはほとんど必要ありませんが, コントローラに DataStore プロバイダなどを設定している場合に利用するといいでしょう. CaoVariable オブジェクトの取得(ICaoController::GetVariable)時に[属性]が書き込み可能なアイテムに対してのみ設定されます.

[詳細情報] – 説明

アイテムに関する詳細な情報を設定できます. ここで設定した情報は XML ファイルに登録されるので, 不揮発な情報を設定するのに向いています. CaoVariable オブジェクトの取得(ICaoController::GetVariable)時に使用されます.

[アイテム詳細設定] – 詳細設定

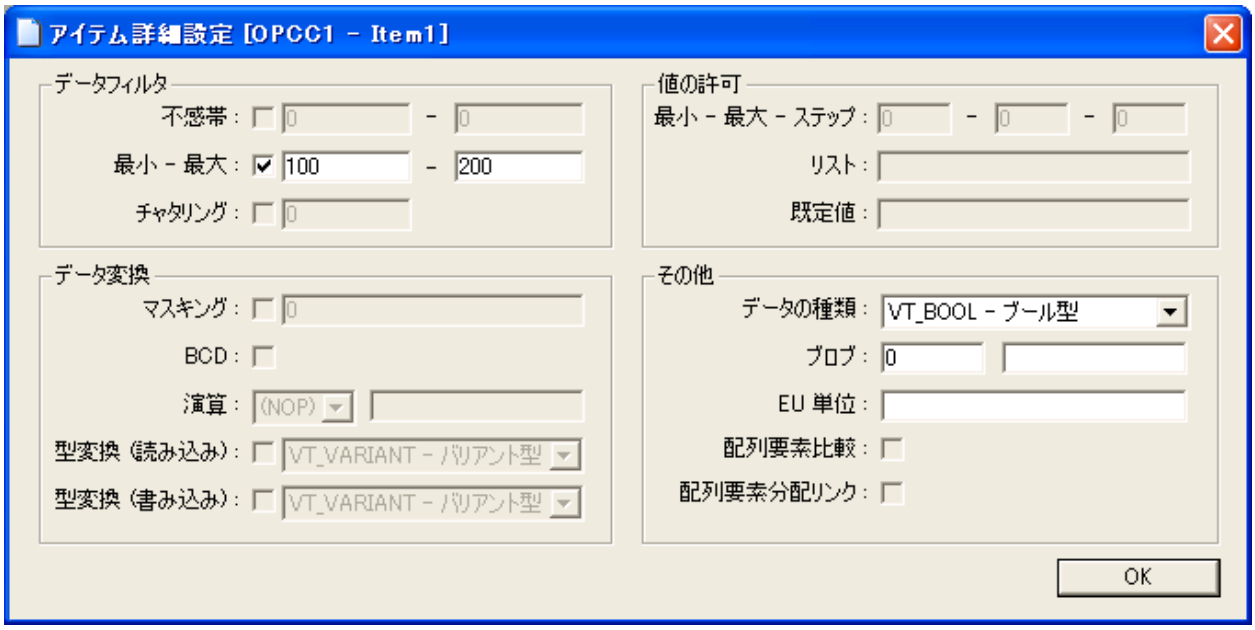

図 3-5 アイテム詳細設定ダイアログ

[Low EU, High EU 設定] –最小-最大

Low EU(VT_R4)と High EU(VT_R4), 有効無効の設定します.

Low EU/High EU は, 値変更判定時に以下の式で使用されます.

| (last value - current value) | > (Deadband percent[9] / 100) * (HighEU - LowEU)

[9] Deadband Percent は, OPCGroup のプロパティ値です.

[データタイプ設定] – データの種類

アイテムのネイティブ型(Canonical DataType)を設定します. 設定可能な型は表 3 の通りです.

表 3-1 アイテムのデータタイプ

データ型	バイト数	説明
VT_BOOL	1	1:True, 1 以外:False
VT_I1	1	符号付整数
VT_UI1	1	符号なし整数
VT_I2	2	単精度(16 ﾋﾞｯﾄ)符号付整数
VT_UI2	2	単精度(16 ﾋﾞｯﾄ)符号なし整数
VT_I4	4	倍精度(32 ﾋﾞｯﾄ)符号付整数
VT_UI4	4	倍精度(32 ﾋﾞｯﾄ)符号なし整数
VT_R4	4	単精度(32 ﾋﾞｯﾄ)浮動小数
VT_R8	8	倍精度(64 ﾋﾞｯﾄ)浮動小数
VT_CY	8	通貨型 小数点以下 4 桁の固定小数点の数値で, スケールが 10000 倍の 8 バイト符号付き整数で格納
VT_BSTR	可変	文字列 VT_UI4 で示された文字数の後に UNICODE 文字と NULL ターミネータを付加
VT_DATE	8	VT_R8 と同じ 1899/12/30 からの通算日時
VT_VARIANT	可変	VARIANT 型

[BLOB サイズ, BLOB 設定] – ブロブ

BLOB のサイズ(VT_I4)と BLOB 文字列(VT_BSTR)を設定します.

[EU Unit 設定] – EU 単位

EU Unit の値(VT_BSTR)を設定します.

3.2.2. メニュー

CaoOPCConfig における基本的な操作はメニューバーから選択します. 一部ツリービューを右クリックして選択できるメニューもありますが, メニューバーから選択できる機能と全く同等です.

3.2.2.1. ファイルメニュー

作成したデータのセーブやロード等の処理を行います.

開く,保存,名前をつけて保存..で使用されるファイルの拡張子は“**csq**”です.

インポート,エクスポートで使用されるファイルの拡張子は“**csx**”です.

CSV ファイルからインポート, CSV ファイルにエクスポートで使用されるファイルの拡張子は“csv”です.

[新規作成] – 新規

新規に csq ファイルを作成します.

[ファイルから読込み] – 開く…

指定した csq ファイルから設定情報を読込みます[10].

読込みをキャンセルした場合には読込み結果は反映されません.

[ファイル保存] – 保存

現在の設定内容を csq ファイルに上書き保存します.

[名前を付けて保存] – 名前をつけて保存…

現在の設定内容を csq ファイルに保存します.

[インポート] – インポート...

csx ファイルからノードを選択しているノードに追加します.

[エクスポート] – エクスポート...

選択しているノードを csx ファイルに出力します.

[CSV ファイルからインポート] – CSV ファイルからインポート...

csv ファイルからノードを選択しているノードに追加します.

[CSV ファイルにエクスポート] – CSV ファイルにエクスポート...

選択しているノードを csv ファイルに出力します.

[10] Win9x/Me では OS の制限から 64K 以上のサイズのファイルの読込みの場合はデータに欠損が生じる場合があるので注意して下さい.

[終了] – 終了

CaoOPCConfig のプログラムを終了します.

3.2.2.2. 編集メニュー

アイテムの追加削除などの編集処理を行います.

[コントローラ追加] – コントローラ追加

CaoOPC で読込むコントローラの新規追加を行います. 既存のコントローラと同じ名前のコントローラは追加できません(大文字・小文字の区別をしません). 有効な名前が入力されるとツリーに追加され, その詳細情報は画面右のタブから入力します.

[アイテム追加] – アイテム追加

ツリービューで選択されたノードにアイテムを追加します. 選択されたコントローラに既に登録されているアイテムと同じ名前のアイテムは追加できません(大文字・小文字の区別をしません). 有効な名前が入力されるとツリーに追加され, 詳細情報は画面右のタブから入力します.

[名前変更] –名前の変更

ツリービューで選択されたコントローラ/アイテムの名前を変更します. すでに登録されているコントローラ/アイテムの名前に変更はできません(大文字・小文字の区別をしません).

[削除] – 削除

ツリービューで選択されたノードを削除します. コントローラの場合はコントローラに登録されたアイテムを再帰的に全て削除します. ツリーでアイテムが選択された状態でキーボードから Del キーを押下することによっても同等の処理が行えます.

[ノードのコピー] – コピー

ツリーで選択されたノードをコピーします. コントローラの場合は登録されたアイテムもコピーされます. 但し, トリガ情報はコピーされません. 下記の“貼り付け”操作でコピーした情報のクローンをノードに追加できます.

[貼り付け] – 貼り付け

コピーされた情報を新しい名前を入力してノード貼り付けます.

[ソート] – ソート

選択されているレベルのノードを昇順ソートします.

3.2.2.3. アクションメニュー

CaoOPC.exe の実行や実行時の設定に関する設定を行います. 設定はレジストリに保存されます.

[オプション設定] – オプション

CaoOPC のオプション設定ダイアログが表示されます.

[全体タブ] – 全体

プロセスの優先度や使用言語など, CaoOPC 全般に関する設定を行うことができます.

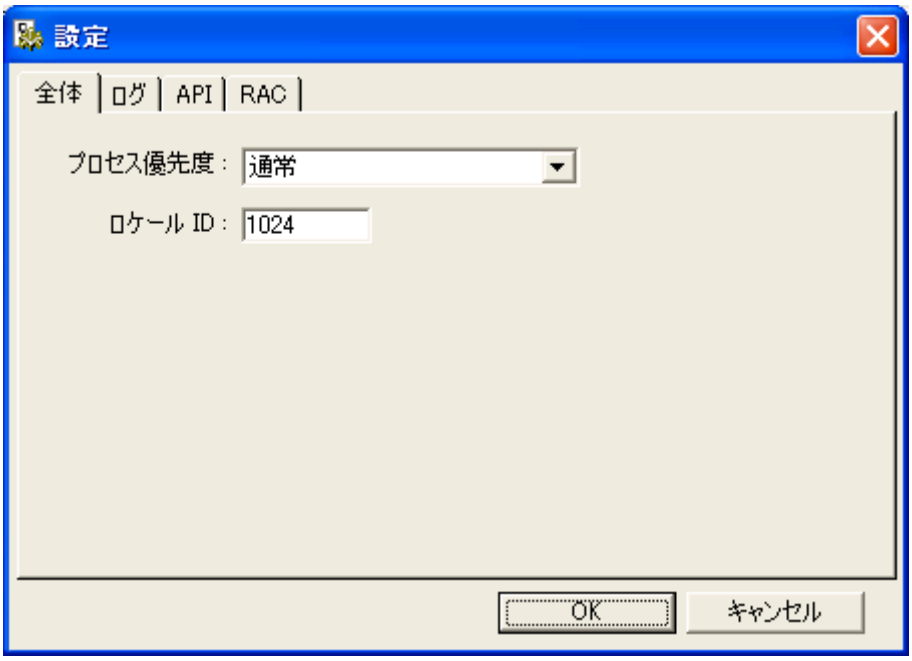

図 3-6 CaoOPC 環境設定（全体）

[プロセス優先度] – プロセス優先度

CaoOPC のプロセス優先度を設定します. 優先度に対する調整は以下の通りです.

リアルタイム > 高 > **通常** > アイドル

[ロケール ID] – ロケール ID

使用する言語 ID を設定します.

[ログ設定タブ] – ログ

CaoOPC のログ出力に関する設定を行うことができます.

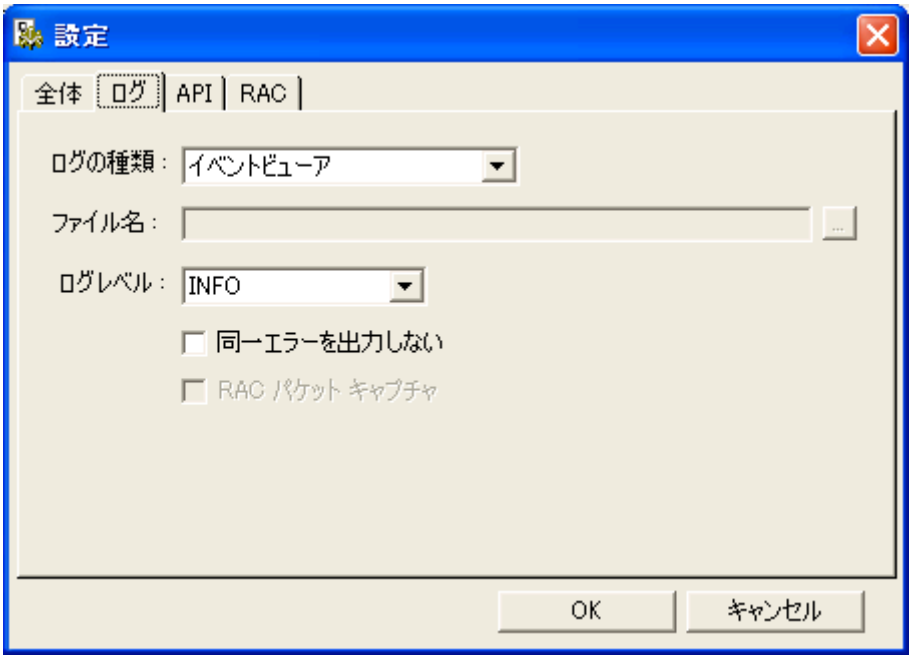

図 3-7 CaoOPC 環境設定(ログ)

[ログタイプ] – ログの種類

CaoOPC.exe のログの出力タイプを選択します.

ログの出力タイプは以下のものを選択することができます.

表 3-2 ログタイプ

出力先	備考
コンソール	コンソールに出力します
メッセージボックス	メッセージボックスに出力します(サービス起動時)
イベント ビューア	イベントビューアに出力します(サービス起動時)
デバッグ ビューア	デバッグ出力します.
テキストファイル	指定したテキストファイルに出力します.

[ファイル名] – ファイル名

Log Type を Text File にした場合にログを出力するファイルパスを設定します.

[ログ出力レベル] – ログレベル

ログの出力レベルを設定します. ログレベルの設定は, 以下の 5 つのレベルから選択することができます. “FATAL”がもっとも深刻度が高く, “DEBUG”に近づくほど深刻度は低くなります. 標準では“INFO”

に設定されています.

FATAL > ERROR > WARN > **INFO** > DEBUG

[同エラー出力抑制] – 同一エラーを出力しない

同じエラーの出力抑制設定を行います. チェックを入れると, 同じエラーが何度発生しても 1 度しかエラーメッセージを出力しません.

[API 設定タブ] – API

設定できる項目はありません.

[RAC 機能設定タブ] – RAC

設定できる項目はありません.

3.2.2.4. ヘルプメニュー

ヘルプやライセンス登録のメニューです.

[バージョン情報]

バージョン情報を表示します.

3.3. レジストリ構成情報

使用するレジストリは CaoSQLConfig と同じですので, 詳しくは『CaoSQL ユーザーズガイド』の「4.3 レジストリ構成情報」を参照して下さい.

4. OPC プロバイダ

4.1. 概要

本章では CAO から OPC(OLE for Process Control)サーバを介して，PLC(Programmable Logic Controller)にアクセスする手段を与える「OPC 接続用の CAO プロバイダ」(以下，OPC プロバイダ)について簡単に解説をおこないます．OPC プロバイダの詳細については，「OPC プロバイダユーザーズガイド 」を参照してください．

このプロバイダにより，CAO 対応ツールは，ロボットのみならず OPC サーバを持つ PLC や表示盤に対してもロボットと同様にアクセスすることができます．

この OPC プロバイダは，OPC サーバが保持する Item の値を CaoVariable オブジェクトから参照することを可能としています．

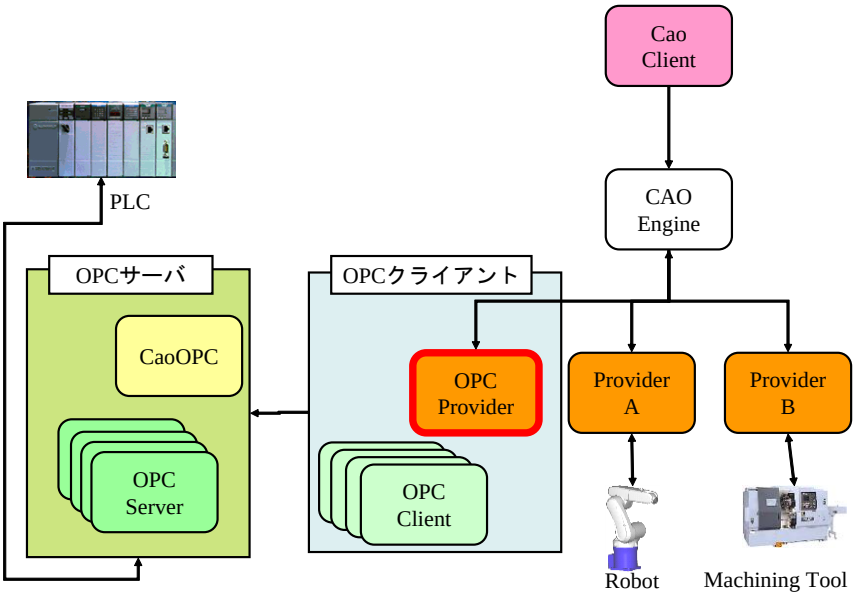

図 4-1 OPC 接続用の CAO プロバイダ

4.2. メソッド・プロパティ

4.2.1. CaoWorkspace::AddController メソッド

CAO の CaoWorkspace クラスの AddController メソッドの仕様を下記に示します.

```
AddController
(
    “<Controller 名>”,                    // コントローラ名
    “CaoProv.OPC ”,                       // プロバイダ名. 固定.
    “<マシン名>”,                         // プロバイダの実行マシン名.
    “<オプション>”                        // オプション文字列(OPC オプション)
)
```

ここで, <プロバイダ名>は固定, <実行マシン名>は他のプロバイダと同じ意味です.

<オプション>の書式は「OPC プロバイダユーザーズガイド」を参照してください.

以下にサンプルを示します.

```
AddController
(
     “OPC1”,
     “CaoProv.OPC”,                                  // プロバイダ名は固定.
     “”,                                             // localhost で起動
     “OPCServer= OPC.DAServer, UpdateRate=1000”      // 更新期間が 1 秒
);
```

4.2.2. CaoController::AddVariable メソッド

CAO の CaoController::AddVariable メソッドの仕様を下記に示します.

```
CaoController:: AddVariable
(
    “<変数名>”,                  // アイテム ID.
    “<オプション>”               // オプション文字列(OPC オプション)
)
```

OPC プロバイダを使う場合は, この引数を次のように設定します.

<変数名> ::=<アイテム ID>

<オプション> ::=[[AccessPath=<アクセスパス>] [, ActiveState=<0|1>] [,RequestType=<受け取り変数型>] [,Source=<データソース>]]

<オプション>の書式は「OPC プロバイダユーザーズガイド」を参照してください.

以下にサンプルを示します.

```
AddVariable
(
     “Item1”,                                                  // アイテム ID が Item1
     “AccessPath=OPCC1,ActiveState=0,RequestType=8,Source=1”   // アクセスパスが OPCC1,
```

```
);                                          アクティブ状態が InActive,
                                            受け取り変数型が VT_BSTR,
                                            データソースが OPC_DS_CACHE
```