CaoSQL

ユーザーズ ガイド

Version 1.9.2

October 2, 2018

【備考】

【改版履歴】

日付	版数	内容
2006-03-09	1.0	初版
2006-08-20	1.1.0	ヒストリ機能で一次元配列データに対応. 書込み時の型変換機能追加
2006-09-22	1.1.1	配列内データ比較機能追加（Compare Arrays オプション）
2007-12-26	1.2.0	配列要素分配リンク機能追加（Extract Linking オプション）
2008-01-30	1.2.1	プロバイダ・セッティング Tips 追加
2008-12-12	1.3.0	トリガアクションに ShellExecute 追加
2009-05-22	1.3.1	コントローラクラス Execute にコマンド追加（$CSQ_RESET_SCANTIME$）
2009-05-26	1.3.2	CaoSQLController::SettingData,CaoSQLItem::SettingData の修正
2009-05-26	1.3.3	セッティング Tips に VT_I4 から二つの VT_I2 へ分解を追加
2009-07-13	1.4.0	ファイルクラス変数追加, アイテム毎の不活性化オプション追加
2009-10-12	1.5.0	二項演算機能に単項演算子（例：NOT, ABS）を追加
2009-11-19	1.5.1	配列型アイテム, エイリアス機能を追加
2009-02-05	1.5.2	アイテムのグループ化機能追加
2010-09-06	1.6.0	リンク先プロパティ選択機能追加
2011-03-19	1.6.1	コントローラクラス Execute にコマンド追加（$CSQ_SCAN_ONETIME$）
2011-03-28	1.6.2	RAC サーバ機能にアイテム ID 指定を追加
2011-06-20	1.7.0	スクリプト実行機能追加
2011-07-03	1.7.1	レジストリ構成情報修正
2011-09-05	1.7.2	RAC サーバ機能のプロパティ取得コマンド追加
2011-10-21	1.7.3	RAC サーバ機能の高速化
2013-02-20	1.7.3	デンソーロボットモード変更（RC8 対応）
2013-12-14	1.8.0	二項演算機能強化（第二演算）
2014-01-13	1.8.1	ヒストリ機能の高速化（値の「文字列変換しない」オプション追加）
2015-05-21	1.8.2	CoAP サーバ機能追加
2015-07-06	1.8.3	ヒストリ機能のバイナリファイル保存機能追加
2015-09-01	1.8.4	ヒストリ機能のパラメータコピーメニューを追加
2015-11-06	1.8.5	ヒストリ機能のシステムログにデータベース情報を追加 アイテム生成時における親オブジェクトのオプション文字列指定を追加
2016-11-22	1.8.6	アイテムを持たないグループを持つコントローラがあった時にアイテムを持たないグループより下にあるアイテムがすべて生成されないバグを修正
2016-12-19	1.9.2	ウォッチドッグタイマの説明修正
2018-10-02	1.9.2	動作確認データベースの追加

【依存モジュール】

No.	依存モジュール
1	MDAC (Microsoft Data Access Components) 2.7
2	Internet Explorer 5.01 SP2 以上

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 CaoSQL のユーザーズガイドです.

CaoSQL は複数の FA 機器からデータを収集し, CaoSQL のクライアントアプリケーション(例えば, 稼動管理や生産指示ソフトなど)へ, その収集したデータを提供するデータ管理用ミドルウェアです.

また, CaoSQL は多種複数の FA 機器からデータを収集しその値を上位アプリケーション等に提供することを主な目的としていますが、それ以外に、収集した値をリレーショナルデータベースに記録することができます. 記録されたデータは Microsoft Access などの汎用フロントエンドソフトでさまざまな解析を行うことができ、例えば,「2002 年 10 月 1 日から 2002 年 10 月 31 日までの一日毎の生産数, その平均, 分散, 標準偏差」がほしい場合も, 簡単にその結果を得ることができます.

CaoSQLは, Access, SQL-Server, ORACLEなど様々な市販のデータベースエンジンで利用することができ[1], それゆえ既存の企業システムとの融合も容易に実現できます. また, CaoSQLは1台のコンピュータ上でも, ネットワークを介した分散環境でも同じようにシステムを構築することができます. 分散環境においては, CaoSQLをカスケード接続して段階的にデータを収集することが可能なので, 工場規模のシステムでもパフォーマンスを損なうことなく, 多くのFA機器を接続することができます.

このように, CaoSQL(とCAO の組合せ)は多くの機器や様々なソフトウェアが関係する課題に対して, 一つのソリューションを与えるシステムとなり得るため, これを活用することで簡単にシステムの開発が可能になります.

本書ではこの CaoSQL の仕様について説明します.

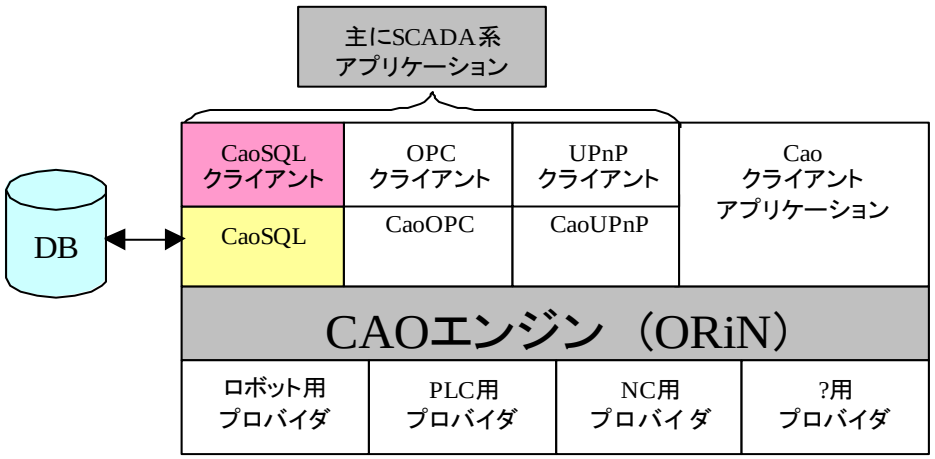


図 1-1 CAO アプリケーションの形態

[1] データベースエンジンによって, パフォーマンスや SQL 言語仕様に若干の違いがあります.

2. CaoSQLの概要

2.1. 概要

下図はCaoSQLの構成です．CaoSQLはCAOのクライアントツールであり，予めCaoSQLConfig.exeで作成した設定ファイルcsqファイル[2]に登録された情報に従ってCAOのCaoControllerオブジェクト，CaoVariableオブジェクトを作成し管理しています．CaoSQLクライアントは，CaoSQLのインタフェースを用いて，それらのデータを取得します．したがって，CaoSQLクライアントはCAOのインタフェースを意識する必要はありません．CaoSQLはデータ（変数）のRead/Writeに特化したミドルウェアなので，CAOのインタフェース仕様に比べると，かなりシンプルな仕様になっています（「オブジェクトモデル」参照）．また，CAOエンジンにはないCaoSQL独自の機能もあります（「CaoSQLの機能」参照）．

CaoSQLConfingツールは，CAOのCaoControllerオブジェクトやCaoVariableオブジェクトに必要なパラメータやCaoSQLの動作オプションを設定するツールです． CaoSQLTesterはCaoSQLのメソッド/プロパティをテストするツールです．CaoSQLCmdツールは，コマンドラインからCaoSQLのメソッド実行や一部のプロパティの取得/設定を行うことができるコンソール・アプリケーションです．CaoSQLLauncherは単純にCaoSQLの起動/終了を行うツールです．CaoSQLConfigについてはCaoSQLConfigを，その他のツールについては「CaoSQLTools ユーザーズガイド」を参照して下さい．

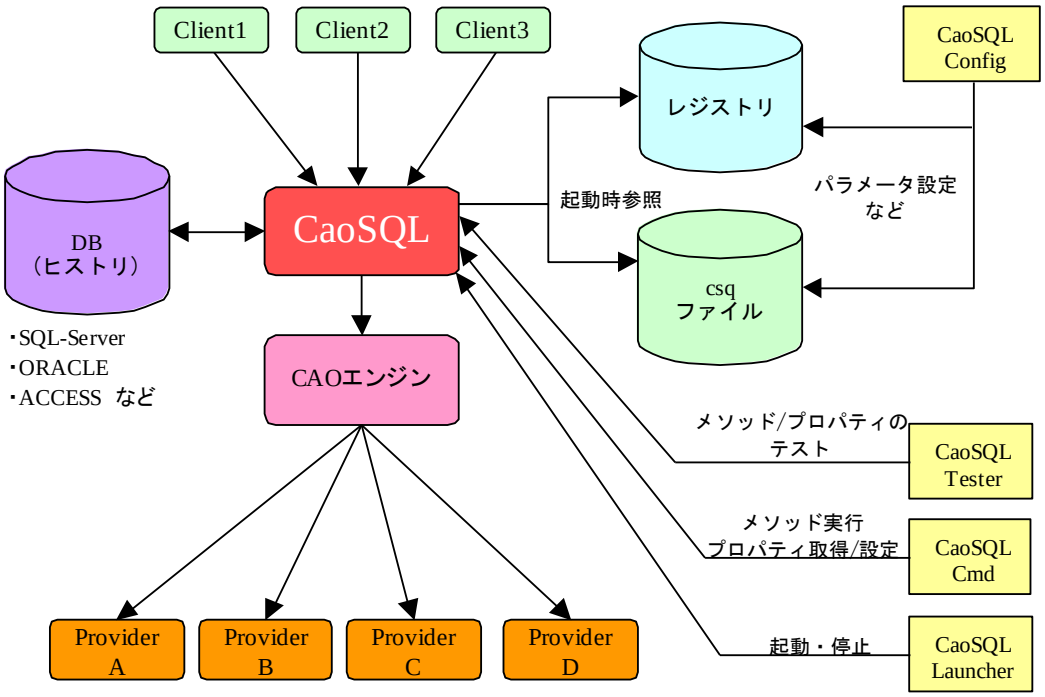


図 2-1 CaoSQL 概要図

[2] CaoSQLConfig にて作成される設定ファイルです．

2.2. オブジェクトモデル

CaoSQLは下図の様な単純なオブジェクトモデルを持っています．CaoSQLEngineが唯一外部から直接生成可能なオブジェクトで，CaoSQLEngineのインタフェースを介してCaoSQLController，CaoSQLHistoryのオブジェクトを操作します[3]．

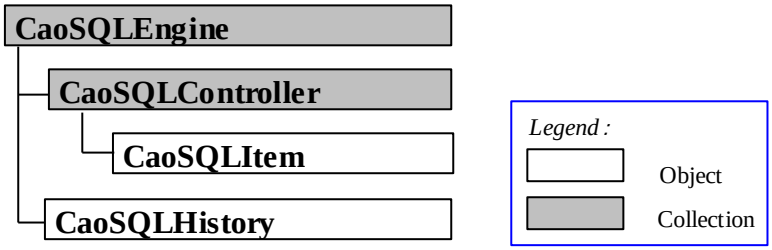


図 2-2 CaoSQL のクラス図

表 2-1 CaoSQL のクラスが提供する機能

クラス名	提供する機能
CaoSQLEngine	CaoSQLController のコレクションクラスです．CaoSQL 全体の設定を保持します．唯一外部から直接生成可能なオブジェクトで，このインタフェースを介して下記のオブジェクトを作成/取得します．
CaoSQLController	CaoSQLItem のコレクションクラス．CAO の CaoController オブジェクトに対応し，その設定を保持します．サンプリングスレッド（ポーリング処理やイベント処理を実行するスレッド）はこの単位で作成されます．
CaoSQLItem	CAO の CaoVariable に対応し，その設定を保持します．CaoSQLConfig で設定したアイテムへの値の読み込みや書き込みなどはこのオブジェクトを使います．
CaoSQLHistory	CaoSQL のヒストリ機能で使用するデータベースの設定内容を保持します．データベースに対して直接 SQL 文を実行することもできます．

2.2.1. CaoSQLControllerのタイプ

CaoSQLController には通常コントローラとエイリアスコントローラの 2 つのタイプがあります．通常コントローラは上記オブジェクトモデルにもあるとおり，CAO の CaoController オブジェクトに対応した処理が行われます．エイリアスコントローラは，エイリアスアイテムをまとめるコントローラです．このコントローラに対してプロバイダやマシンを指定することはできません．

[3]各クラスの関数一覧は，付録 5.5.付録A.1 かCaoSQLObjectsを参照して下さい．

2.2.2. CaoSQLItemのタイプ

CaoSQLItem はコントローラとは違い，通常アイテムとエイリアスアイテムの他に配列型アイテムの設定があります．通常アイテムはコントローラ同様，CAO の CaoVariable に対応した処理が行われます．

エイリアスアイテムは，登録されている通常アイテム，配列型アイテムを別名アイテムとして登録ができ，値をリンクすることができるアイテムです．

配列型アイテムは複数(1 つ以上)のアイテムの値を配列として読み込みや書き込みを行えるアイテムです．

2.3. 動作モデル

CaoSQL のプロセス内部では CaoSQLController 毎に CaoController オブジェクトを一つ保持し，スレッド（サンプリングスレッド）を起動します．

起動されたそれぞれのスレッドは，CaoSQLController オブジェクトに属している CaoSQLItem オブジェクト内の CaoVariable の値を，指定されたサンプリング間隔で周期的に取得し，その値をキャッシュします．

CaoSQL クライアントは，読込み関数でこの内部のキャッシュを参照します．ここで定期的に取得された値とキャッシュされた値が異なる場合はキャッシュの値を書き換え，必要であればヒストリの書き込み，OnChangeItem イベント，OnChangeState イベントの発生等の処理を行います．但し，値の変化を監視できる変数の型は下記のものに制限されているので注意して下さい．

表 2-2 値変化を監視できる VT 型

VARIANT の型	意味	C 言語の型
VT_EMPTY	（空）	-
VT_NULL	NULL	-
VT_I1	符号付き1バイト整数型	char
VT_UI1	符号なし1バイト整数型	unsigned char
VT_I2	符号付き 2 バイト整数型	short
VT_UI2	符号なし 2 バイト整数型	unsigned short
VT_I4	符号付き 4 バイト整数型	long
VT_UI4	符号なし 4 バイト整数型	unsigned long
VT_R4	単精度浮動小数点型	float
VT_R8	倍精度浮動小数点型	double
VT_CY	通貨型	-
VT_DATE	日付型	DATE(double)
VT_BSTR	(UNICODE)文字列型	BSTR
VT_ERROR	エラー型	-
VT_DISPATCH	IDispatch ポインタ	IDispatch*

VT_UNKNOWN	IUnknown ポインタ	IUnknown*

これら以外の変数の型に関しては値変化に伴う処理，ヒストリ書込み，リンク，OnChangeItem イベントの発生等の処理は一切行われませんが，キャッシュの値は随時更新されます．この更新されたタイミングで取得時刻(DateTime プロパティ)も更新されます．

しかし，CaoSQLがCAOからの値の取得に失敗した場合は，アイテムのステータスにエラーが設定され，キャッシュ値および取得時刻は更新されません[4]．

また，逆に値を書き込む場合は，クライアントが書き込み関数を実行した時点で CaoSQL が CAO に対して書き込みを代行します．

これらの処理はスレッド単位で実行されるので，ある応答の遅いコントローラがあっても，ほとんど別のコントローラに影響を及ぼすことはありません．さらにこのスレッドの優先度は，CaoSQLConfigツールを使って明示的に指定することもできます（詳細は「CaoSQLTools ユーザーズガイド」を参照して下さい．）

コントローラのスレッド，およびアイテムのステータスの一覧は，CaoSQLObjectsを参照して下さい．

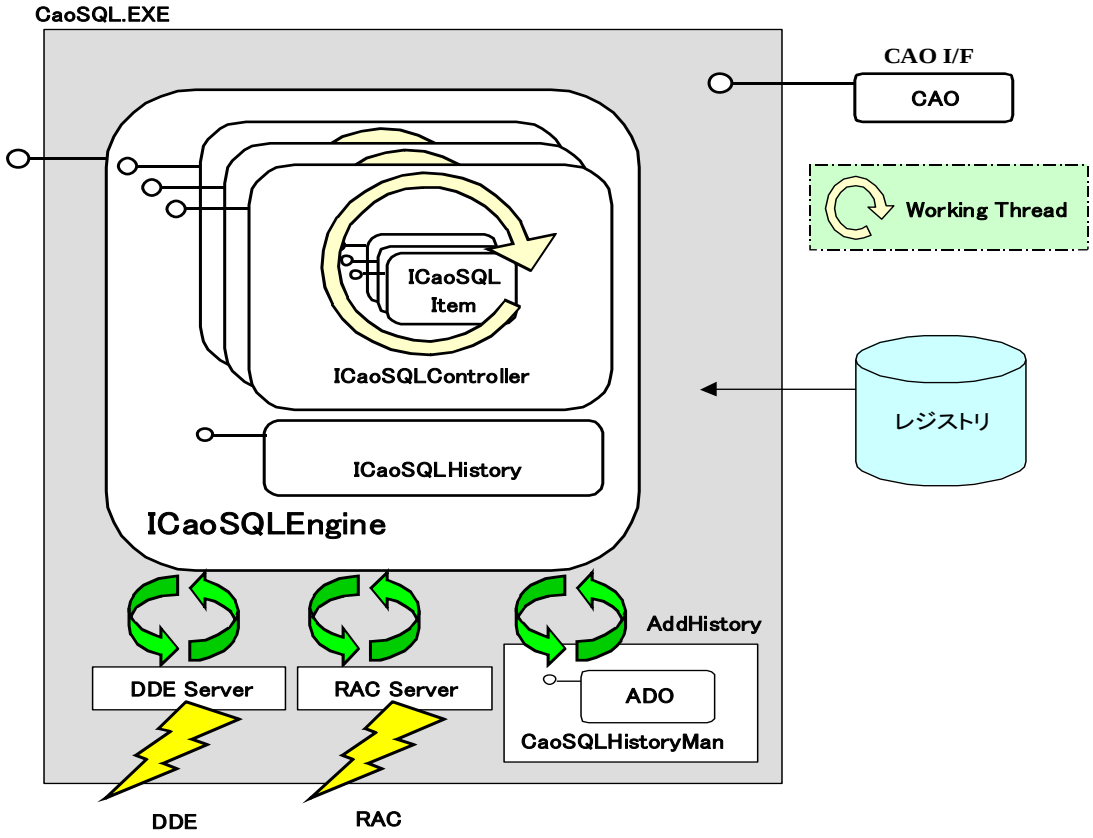


図 2-3 CaoSQL ワーキングモデル

[4] コントローラオブジェクト内の全アイテムの取得に失敗した場合，他のコントローラの処理を優先するために 1 周期分遅延させるというペナルティが自動的に科せられます．
均等配分(distribute equally)オプションが有効な場合は，1 アイテムの値取得毎に（サンプリング間隔／アイテム数）時間の遅延が入ります．
サンプリングオフ（Startup off）の場合は，読込んだ時点でキャッシュ値を更新します．

2.4. サンプルプログラム

2.4.1. Visual Basic

ここでは, CaoSQL 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 を使用して記述しますが, その他の言語でも同様の手続きで実装することが可能です.

まず, CaoSQL エンジンのインスタンスを生成するには, 次のように記述します.

```
Dim csqlEng As Object
Set csqlEng = CreateObject("CaoSQL.CaoSQLEngine")
```

ここで, "CaoSQL.CaoSQLEngine"というのが CaoSQL のプログラム ID です. 参照設定で CaoSQL が登録してある場合は, 以下のように New 演算子でオブジェクトを生成することもできます.

```
Dim csqlEng As New CaoSQLEngine
```

次に, このオブジェクトを使ってコントローラ名"RC1"のアイテム名"I1"の値を変数 x に代入するには次のように記述します.

```
Dim x As Variant
x = csqlEng.Controller("RC1").Item("I1").Value
```

ここで, Controller プロパティや Item プロパティや Value プロパティはそれぞれのクラスのデフォルトプロパティですので, 次のように記述することもできます.

```
x = csqlEng("RC1")("I1").Value
```

逆に, 同じアイテムの値を 10 に設定する場合は, 以下のように記述します.

```
csqlEng("RC1")("I1").Value = 10
```

上記の例は分かりやすいように記述しましたが, 実際にそのように記述すると, CaoSQLController や CaoSQLItem クラスのインスタンスがその都度, 生成されプロパティ参照後に削除されるので効率がよくありません. したがって, 実際のアプリケーションでは, 一度きりの利用でない限り, 以下の例のように一旦オブジェクトのインタフェースを保存して再利用すべきです.

```
Dim x As Variant
Dim csqlCtrl As CaoSQLCmd
Dim csqlItem As CaoSQLItem
Option Explicit


Set csqlCtrl = csqlEng.Controller("RC1")
Set csqlItem = csqlCtrl.Item("I1")
Debug.Print "I1 = " & csqlItem.Value
```

2.4.2. SQL

CaoSQL はアイテムの値の変化を検知してデータベースに記録する機能があります. これを使用すると後で SQL 言語を使って様々なデータ分析ができます. ここでは, CaoSQLHistory インタフェースを使って, どのように SQL を実行するかを示します.

CaoSQLHistory インタフェースには, CaoSQLConfig ツールの Database 設定ダイアログで設定した内容を取得するプロパティがあります. もちろん, ヒストリテーブル名も取得できます. また, これらの他に SQL 文をそのまま実行し結果のレコードセットを返す Execute メソッドがあります. この Execute メソッドを使えば, データベースの Open や Close というような煩わしい処理をせずにヒストリテーブルの解析ができます.

例えば,「コントローラ名“RC1”のアイテム名“I1”のレコードセット」を抽出したい場合は, 次のように書きます.

```
Dim csqlHis As CaoSQLHistory
Dim rs As ADODB.Recordset

Set csqlHis = csqlEng.CaoSQLHistory
Set rs = csqlHis.Execute("SELECT * FROM caosql_history _
                          WHERE controller_name = 'RC1' AND item_name = 'I1'")
```

ここで, 下線を引いた“caosql_history”の値は, CaoSQLConfig ツールで指定したテーブルと同じテーブルに設定する必要があります. これを, プログラムで実現したい場合は, 以下のように, TableName プロパティを使います. このとき, テーブル名の前後の空白が必要なので注意して下さい.

```
Set rs = csqlHis.Execute( "SELECT * FROM " & csqlHis.TableName & _
                          " WHERE controller_name = 'RC1' AND item_name = 'I1'" )
```

以下にその他の SQL サンプルを示します.

表 2-3 ヒストリテーブルへの問合せ例

問合せ例	SQL 文(下線部は任意に変更)
コントローラ毎のレコード数	SELECT controller_name, COUNT(*) FROM caosql_history GROUP BY controller_name
あるコントローラのアイテム毎のレコード数	SELECT controller_name, item_name, COUNT(*) FROM caosql_history WHERE controller_name = 'RC1' GROUP BY controller_name, item_name
あるアイテムだけのレコードセット	SELECT * FROM caosql_history WHERE controller_name = 'RC1' AND item_name = 'I1'
最新データ(flags が 1)だけのレコードセット	SELECT * FROM caosql_history WHERE flags = 1
あるコントローラのアイテム毎の最大値, 最小値	SELECT controller_name, item_name, MIN(item_value), MAX(item_value) FROM caosql_history WHERE controller_name = 'RC1' GROUP BY controller_name, item_name
あるコントローラのアイテム毎の合計と平均と標準偏差	SELECT controller_name, item_name, SUM(CAST(item_value AS decimal)), AVG(CAST(item_value AS decimal)), STDEV(CAST(item_value AS decimal))

	FROM caosql_history WHERE controller_name = 'RC1' GROUP BY controller_name, item_name
ある指定時間内のレコードセット	SELECT * FROM caosql_history WHERE item_time BETWEEN '2002/03/18 15:09:35' AND '2002/03/18 15:09:40'
あるコントローラの全アイテム値の平均より大きいアイテムだけのレコードセット[5]	SELECT * FROM caosql_history WHERE item_value > (SELECT AVG(CAST(item_value AS decimal)) FROM caosql_history WHERE controller_name = 'RC1')
あるコントローラの全レコード削除	DELETE FROM caosql_history WHERE controller_name = 'RC1'

[5] WHERE 句の中で集計関数は使えないことに注意して下さい.

2.4.3. Active Server Page (ASP)

ORiN2 SDK によってインストールされた CaoSQL には，ASP のサンプルプログラムが以下のフォルダにあります．

サンプルプログラムのパス：“<ORiN2 インストールディレクトリフォルダ>¥CaoSQL¥Sample¥ASP”

サンプルプログラムは以下の 3 つのファイルで構成されています．

(1) A.asp

・ サーバの CaoSQLConfig で設定した各コントロール表示して，アイテムを取得します．

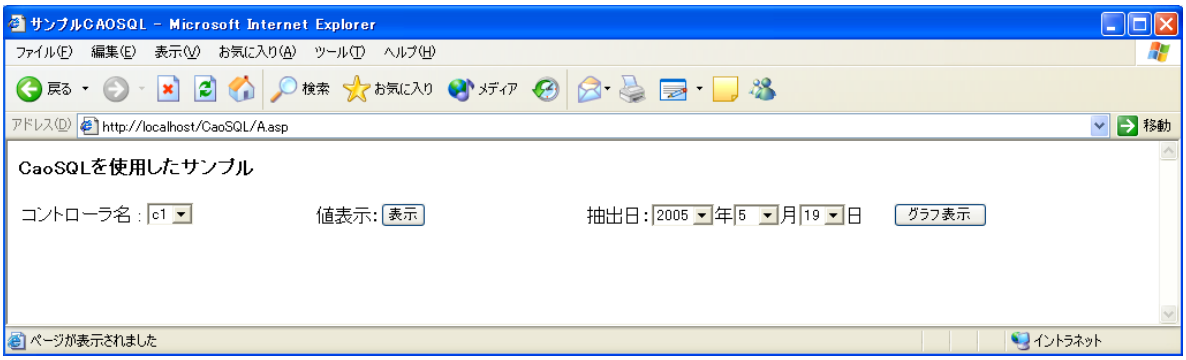


図 2-4 A.asp

(2) B.asp

・ CaoSQL の History オブジェクトを使用して，DB からデータを取ってきます．

・ History の Execute に条件を抽出する SQL 文を投げてやって，レコードを取得します．

・ それを元に，OfficeXPweb コンポーネントを使用してグラフに表示します．

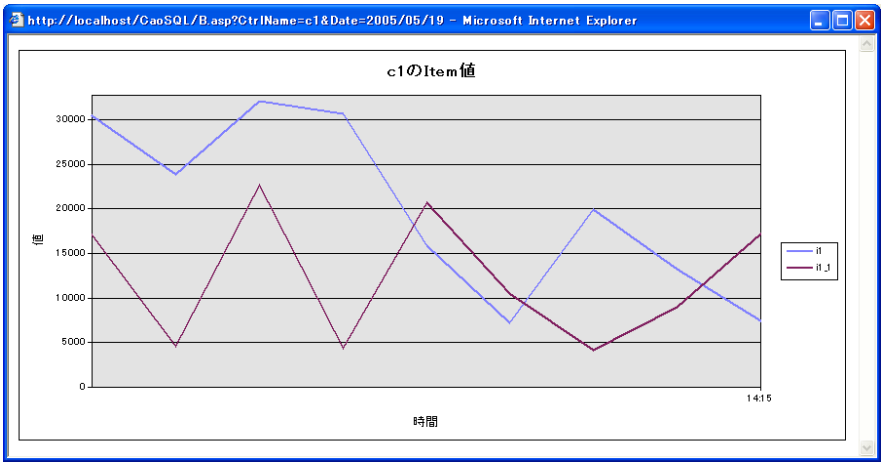


図 2-5 B.asp

(3) C.asp

・ A.asp から呼ばれて，指定されたコントローラのアイテムの値を表に出力します．

・ 自動更新でないので，更新ボタンを押すごとに更新させます．

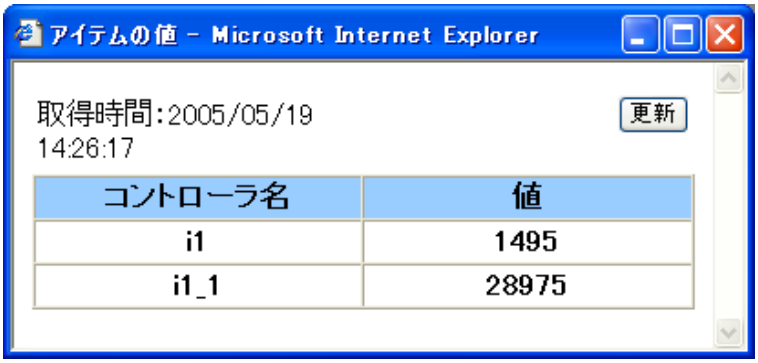


図 2-6 C.asp

2.4.3.1. インストール手順

(1) コンポーネントのインストール

ORiN version 2 をインストールします．

IIS (Internet Information Service) をインストールします．

OfficeXPweb コンポーネントをインストールします．

図 2-7 Office XP web コンポーネントの入手先

ファイル名	入手先 URL
owc10.exe	http://www.microsoft.com/downloads/details.aspx?FamilyID=982b0359-0a86-4fb2-a7ee-5f3a499515dd

(2) IIS で仮想ディレクトリの作成

IIS の管理ツールで仮想ディレクトリを作成します．

仮想ディレクトリの参照先を，<ORiN2 インストールディレクトリ>¥CaoSQL¥Sample¥ASP に設定します

(3) CaoSQL のセキュリティ設定

（オプション）サービスとして起動する場合はコマンドプロンプトで "CaoSQL.exe /service"を実行して下さい．

DCOM 設定ツール"dcomcnfg.exe"を起動し[CaoSQL]のプロパティダイアログを開きます．

作成した仮想ディレクトリのセキュリティポリシーにあった設定をします．（[アクセス権], [起動アクセス

権]など）

(4) 動作確認

IE ブラウザで"http://localhost/<仮想ディレクトリ名>/A.asp"を開きます.

サンプルがうまく動作しないときは DCOM の設定などを再度確認して下さい.

2.5. エラー処理

ここでは CaoSQL で内部での CAO の IF のエラーが起きた場合，どのような処理が行われ，またエラーをクライアント側でそれを把握するかを解説します.

2.5.1. CaoSQLEngineでエラーが発生した場合の処理

CaoSQLEngine でのエラーは CaoEngine の取得失敗，CaoSQLConfig の設定情報の読み込み失敗などがあります. これらは CaoSQL の根幹の動作に関わるエラーであり，エラーをログ出力して CaoSQL.exe 自体の実行が停止します.

2.5.2. CaoSQLControllerでエラーが発生した場合の処理

[初期化時のエラー]

CSQ ファイルの読み込み失敗などの初期化エラーがこれにあたります. この場合 CaoSQLController は復帰することはできません. エラーをログへ出力し，CaoSQLController は生成されません.

[起動時のエラー]

CaoController 生成やスレッドの生成時のエラーがこれにあたります. この場合 CaoSQLController のインスタンスは存在しますが，スレッドの動作は行われていません. エラーはログに出力され，CaoSQLController のステータスは Deactive に設定されます.

2.5.3. CaoSQLItemでエラーが発生した場合の処理

[初期化時のエラー]

CSQ ファイルの読み込み失敗などの初期化エラーであり，CaoSQLItem オブジェクトは生成されません. エラーはログに出力されます.

[起動時のエラー]

CaoVariable 取得時のエラー等がこれにあたります. この場合エラーはログに出力され，さらにその状態は CaoSQLItem のステータスに保存されます.

[稼動時のエラー]

基本的にはエラーが発生してもリトライし，サンプリングスレッドには影響しません. つまり，スレッドが停止したりプロセスが終了したりして CaoSQLItem の管理オブジェクトが解放されない限り，サンプリングは繰り返されます. (エラーをログへ出力)また Item はサンプルの度にアイテムの状態をステータスとして

記録され, その状態は CaoSQLItem インタフェースから State プロパティを使って状態を取得することが可能です.

表 2-4 CaoSQLItem の State

State	詳細
csItemSucceeded	アイテムの値は正常に取得/設定されました.
csItemFailed	アイテムの値の取得/設定に失敗しました.
csItemUndefined	アイテムの値は不定状態です.

2.5.4. CaoSQLHistoryでエラーが発生した場合の処理

[初期化時のエラー]

ここでのエラーは CSQ ファイルの読込み失敗などの初期化エラーです. エラーはログに出力されます.

[起動時のエラー]

DataBase への接続失敗等がこれにあたります. この場合エラーはログに出力され, 自動切替え機能が設定されている場合は 2nd データベースへ接続を試みます.

[稼動時のエラー]

テーブルへのレコード追加失敗等がこれにあたります. この場合エラーはログに出力され, 自動切替え機能が設定されている場合は 2nd データベースへ接続を試みます.

3. CaoSQLの機能

3.1. 機能一覧

CaoSQL には以下の機能があります. これらの機能を有効に使えば, 最小限のプログラミングで複数の FA 機器とのデータリンクが実現できます.

表 3-1 CaoSQL 機能一覧

機能	解説
【プロセス全体の機能】	
ログ (エラーロギング)	オブジェクトの生成や削除, CaoSQL のスタートやストップ, エラーなど動作状況を概観できるログを出力する機能です. 出力先はイベントビューワ, デバッグビューワ, コンソールの 3 種類を選択できます.
ヒストリ機能 (データロギング)	アイテムの値が変化したとき, またはSnapshotメソッドを明示的にコールしたときに, アイテムの値をデータベースに記録する機能です. (セカンダリDB切替え機能参照)
セカンダリ DB 切替え機能	ヒストリ機能を使ってアイテムの値をデータベースに記録している途中にデータベースサーバの異常で記録できない場合に, 自動的にセカンダリデータベースに切替えて記録を継続する機能です. これにより, たった一つのデータも取りこぼすことなく記録ができます.
DDE サーバ機能	DDE サーバ機能です. この機能を使って, DDE クライアントになるアプリケーション(例えばエクセル)から, プログラミングレスでアイテムの値を取得することができます.
RAC サーバ機能	RAC サーバ機能です. RAC プロバイダを使用し, ネットワークを介して RAC コマンドで CaoSQL とデータ交換を行えます.
CoAP サーバ機能	CoAP サーバ機能です. CoAP プロバイダを使用し, ネットワークを介して CoAP プロトコルで CaoSQL とデータ交換を行えます.
【コントローラ単位の機能】	
カスケード接続	この機能により分散環境上で使用するときに, CaoSQL プロバイダを使用することで, 各マシンの CaoSQL で収集した情報を段階的に一つのマシンに集約することができます. (備考) 同様な機能は RAC プロバイダなどを使用して実現することもできます.
サンプリング機能	コントローラ単位で非同期にデータをサンプリングしキャッシュすることで, クライアントアプリケーションからの値要求に高速に応答することができます.
ウォッチドック機能	設定したサンプリング周期に遅延が発生したときに設定した処理を実行す

	ることができます.
トリガ機能	ある一定の条件が成立したときに, サンプリングスレッドに対して設定した処理を実行することができます.
自動接続復帰機能	コントローラのステータスでエラー状態が続いた場合に, 関連する CAO オブジェクトを再作成します. 接続中にデバイスがシャットダウンした場合に自動復帰させたい場合に使います.
アイテムの動的追加機能	CaoSQL 起動後にアイテムを動的に登録・削除する機能です. 削除を行えるのは動的に登録したアイテムのみです.
スクリプト実行機能	VB スクリプトを実行する機能です. 起動時に CaoSQLConfig で設定された VB スクリプトを読み込み, サンプリング毎にスクリプト内のサブルーチン Main を実行します.
【アイテム単位の機能】	
配列要素抽出機能	アイテムの値が配列である場合に, 必要な要素だけを抽出することができます. 抽出したい要素のインデックス番号をカンマで区切り, ()で括ってアイテム名に付加します. データ加工順位:(読込み時)1.
マスキング	アイテムの値をマスキングします. 必要なビットだけを抽出したい場合にマスクを設定すると自動的にマスキングされます. この機能が有効なのは配列でない 1 バイト, 2 バイト, 4 バイトの符号付き・符号なし整数のみです. データ加工順位:(読込み時)2, (書込み時)3.
BCD 変換	読込み時・書き込み時にアイテムの値を BCD(Binary Code Decimal)変換します. この機能が有効なのは配列でない 1 バイト, 2 バイト, 4 バイトの符号付き・符号なし整数のみです. データ加工順位:(読込み時)3, (書込み時)2.
二項演算機能	別のアイテムと二項演算する機能です. 加算(+), 減算(-), 掛算(*), 割算(/), 割算の剰余(MOD), 論理積(AND), 論理和(OR), 排他的論理和(XOR)の 8 つの演算が行えます. データ加工順位:(読込み時)4.
データ型変換機能	アイテムのデータ型を指定した型に変換する機能です. 配列にも対応しています. データ加工順位:(読込み時)5, (書込み時)1.
不感帯設定	値の変化とみなさない範囲を指定します. この範囲内であれば, CaoSQL は値が変化したとみなさないので, それに同期した機能(例えば, イベント通知やアイテムリンクなど)の挙動が変化します. この機能が有効なのは配列でない数値型のデータのみです.

感帯設定	値の変化とみなす範囲を指定します. この範囲外であれば, CaoSQL は値が変化したとみなさないので, それに同期した機能(例えば, イベント通知やアイテムリンクなど)の挙動が変化します. この機能が有効なのは配列でない数値型のデータのみです.
チャタリング除去機能	チャタリング除去のため, 設定回数同じ値をサンプリングできたときにのみ値を更新する機能です.
アイテムリンク機能	アイテムの値が変化したとき, そのアイテムから別のアイテムにリンクが設定してあると, そのリンク先のアイテムに値を伝播させる機能です. これは, あたかも IO の結線をソフト的に実施するような機能です.
不活性化機能	コントローラのサンプリング周期にアイテムの読込み, または書き込みを行わない設定をアイテム毎に設定できます.
配列型アイテム機能	アイテムの読み込みをした場合, リンクに設定しているアイテムへ配列要素の値を伝播させます. 書き込み設定の場合は, リンク設定しているアイテムの値を要素とした配列を書き込みます. この機能は Bit などを扱う場合などに便利です.
エイリアス機能	通常のアイテム, 配列型アイテムを別名として登録することができ, 値をリンクする機能です. エイリアスとして登録したアイテムの値を,さらに伝播することや, データ加工することも可能です.

3.2. イベント通知機能

イベント通知機能を使えばイベント駆動型のプログラミングが可能になり, クライアント側でポーリング処理をする必要がないので効率のよいプログラムが作成できます. イベントの発行は, CaoSQL 内部ではサンプリングスレッドで処理されます. このため, クライアント側のイベントハンドラの処理によって, サンプリングが遅れることがありますので注意してください. また, イベントハンドラ内で CaoSQL の関数を再帰的に呼び出した場合も, タイミングによってはロックする可能性があります. したがって, クライアントプログラム内のイベントハンドラはできる限り非同期化してください.

このサンプリングスレッドはコントローラ毎に生成されます. CaoSQLConfig ツールの設定でこのイベントの発生をコントローラ単位でキャンセルできます. また, アイテム単位でもキャンセル設定可能なので, イベントを使わない場合はキャンセルしておきましょう.

値の変化を検出できる型は, VT_EMPTY, VT_NULL, VT_BOOL, VT_I1, VT_UI1, VT_I2, VT_UI2, VT_I4, VT_UI4, VT_R4, VT_R8, VT_CY, VT_DATE, VT_BSTR, VT_ERROR, VT_DISPATCH, VT_UNKNOWN の 17 のデータ型です. 配列データは, **[Compare Arrays]**の設定が有効な場合は配列の各要素を比較し, 無効な場合は常に値が変化したと解釈されます. ただし, この機能は配列の各要素を比較するためパフォーマンスが落ちる場合があります.

3.2.1. OnChangeItemイベント

OnChangeItem イベントは登録されたアイテムの値が変化したときに発行します. OnChangeItem イベントは 2 つの引数があり, それぞれアイテム名(BSTR)と値(VARIANT)です. アイテム名はコントローラ毎に重複せず(大文字・小文字区別なし)設定されています.

3.2.2. OnChangeStateイベント

OnChangeState イベントは登録されたコントローラのステータスの値が変化したときに発行します. OnChangeState イベントは 1 つの引数があり, それはステータス値(long)です.

ステータス値については表 3-2, 付録0 を参照して下さい.

このステータスの変化を受けて, クライアントアプリケーションでは, 1 スキャン毎に処理を実行することができます.

表 3-2 コントローラのステータス値一覧

ステータス値	状態説明
0	スレッド未初期化
1	スレッドアクティブ
2	スレッドディアクティブ
3	スレッドエラー発生
4	スレッド終了

3.3. トリガ機能

トリガ機能とは, ある一定の条件が成立したときに, サンプリングスレッドに対してアクションを設定することができる機能のことです. 例えば, あるコントローラ内のアイテムは, ある条件が成り立った時にだけサンプリングすればいい場合があります. こういった場合にトリガを設定することで, トリガの設定がないと必要のないアイテムでも毎回サンプリングされるというようなオーバーヘッドを低減させることができ, さらにはその他のサンプリング間隔に影響を及ぼす可能性を抑えることができます.

トリガ機能は以下のような設定項目を持ちます. (設定方法は「CaoSQLTools ユーザーズガイド」を参照して下さい)

表 3-3 トリガ機能設定項目

設定項目	設定内容	
トリガアイテム	条件を監視する CaoSQL アイテム名	
アクション	Start	全アイテムのサンプル開始
	Stop	トリガアイテム以外のサンプルを停止
	1 Time	全てのアイテムを一回だけサンプリングする

	ShellExecute	指定されたプログラムを起動する
条件式	＝＝，！＝，＜，＜＝，＞，＞＝	
閾値	トリガアイテムの持つ値と比較する値	

ここでトリガ機能の“Stop”はCaoSQLControllerのメソッドによるStopとは動作が全く異なることに注意して下さい. トリガ機能による“Stop”アクションではスレッド自体停止されず, 指定してあるトリガアイテムのサンプリングは行われて続けています.

これに対し, CaoSQLController::Stopはスレッド自体停止する, 時間制限的な処理(例えばラインが稼動する時間に合わせてStart/Stopする)等に有効です.

3.4. DDEサーバ機能

この DDE サーバ機能を使って DDE クライアントアプリケーション(例えば Excel)から，プログラミングレスで CaoSQL のアイテム値を取得することができます.

CaoSQL は DDEML を使用した DDE サーバ機能を**読み込み**に限定してサポートしています．また，ホットリンクにも対応していますので DDE クライアントに値変更を通知することも可能です．CaoSQL の DDE 機能を使用するには以下の情報から必要なアイテムを指定します.

表 3-4 DDE サーバ情報

項目	設定	備考
DDE サーバ名	CaoSQL	“CaoSQL”固定
トピック名	コントローラ名	CaoSQL で設定済みの CaoController 名
アイテム名	アイテム名	CaoSQLで設定済みのCaoController名に登録した Item名．Read属性に限ります．

以下に，サンプルとして Excel2000 を DDE クライアントとした場合の設定方法を示します.

(1) DDE サーバ機能を有効にする.

CaoSQLConfig を起動し，「アクション」→「オプション」→「API」から，DDE サーバ項目にチェックを入れて，OK ボタンを押して下さい.

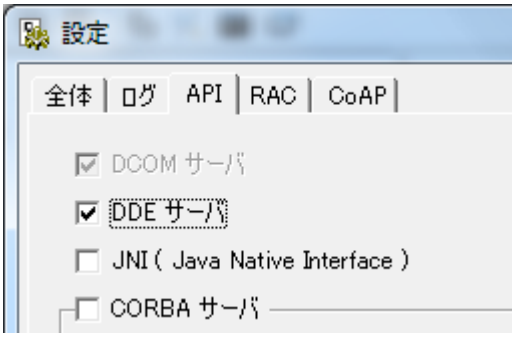


図 3-1 DDE サーバ機能の設定

(2) CaoSQL を起動する.

CaoSQLLauncher 等を使用し，CaoSQL を起動します．但し CaoSQL をサービスとして起動する場合は，サービスコントロールマネージャの設定から，「デスクトップとの対話をサービスに許可」のチェックを入れる必要があります．Windows XP の場合，「管理ツール」→「サービス」→「プロパティ」から，WindowsNT4.0 の場合，「サービス」の「スタートアップ」から，Windows2000 の場合は「サービス」→「プロパティ」で表示されるログオンタブから設定できます.

図 3-2 Windows XP でのサービス設定

(3) Excel を起動し，アイテムを登録する.

CaoSQLConfigにはExcelで使用する場合，簡単にアイテムを登録できるようにクリップボードへDDE登録文字列をコピーする機能があります．(「図 3-3」のDDE文字列のコピーを参照)これをExcelのSheetにペーストし，DDEサーバのアイテムを参照させます.

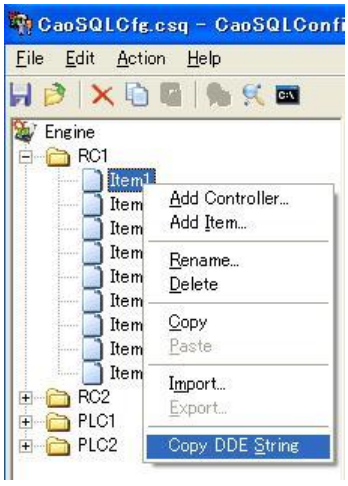


図 3-3 DDE 文字列のコピー

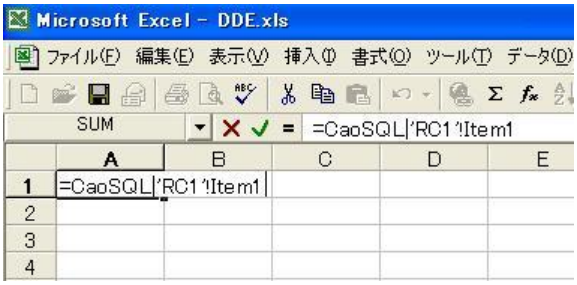


図 3-4 DDE 文字列を Excel にペースト

以上で設定は完了です. Excel2000 を DDE クライアントとした場合, DDE サーバはホットリンクを行うので, アイテムに値変化がおきたときに DDE クライアントに通知し, リンクしたセルに値が反映されます.

3.5. ネットワークDDEサーバ機能

DDE サーバ機能をネットワーク上で使用できるようにする機能です. CaoSQL は DDEShare プログラム (ddeshare.exe)を利用してネットワーク DDE の機能を提供します.

例えば, ネットワーク DDE を使用してサーバマシンで動作している CaoSQL のデータを別のクライアントマシンの Excel を使ってデータを取得することができます. 以下に, ネットワーク DDE を使用する際の設定手順を示します.

(1) DDEShare の設定

ネットワーク DDE を使用するには, DDE 共有の設定をする必要があります. スタートメニューの「ファイル名を指定して実行」から「DDEShare.exe」を指定して DDEShare を起動します.

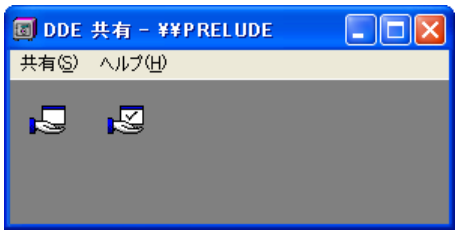


図 3-5 DDEShare のメイン画面

この DDEShare の設定はサーバとクライアントでそれぞれ設定を行います. 「Administrator」でログインして設定しなければ設定が有効になりません.

(2) DDE 共有の作成

DDEShare のメニューから「DDE 共有…」を選択すると, DDE 共有一覧を表示されたダイアログが表示されます.

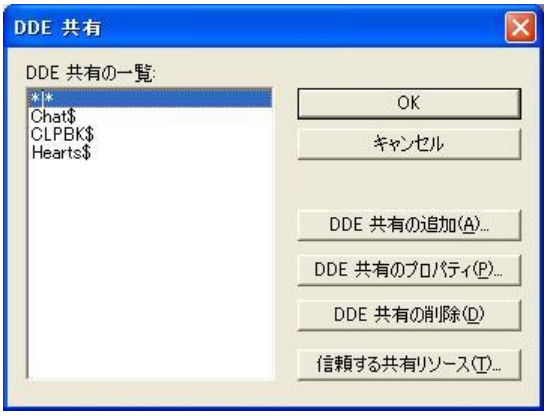


図 3-6 DDE 共有ダイアログ

新しく共有を作成する場合は,「DDE 共有の追加」を選択します. 一度作成した共有を編集したい場合は,「DDE 共有のプロパティ」で編集することができます.

図 3-7 DDE 共有の追加ダイアログ

このダイアログから「アクセス許可…」でアクセス権を設定することができます. ネットワーク DDE を使用する場合,「信頼する共有リソース」の設定も行う必要があります. ただし, この設定はクライアントマシン側で行います. (この設定を行う場合も「Administrator」で設定する)

(3) クライアントアプリケーションで DDE を使用する

クライアントアプリケーション(ここでは Excel で説明)で DDE を使用して, サーバからデータを取得します. 前節で説明したのと同じように Excel のセルに DDE 文字列を入力することで, DDE を使用することができます.

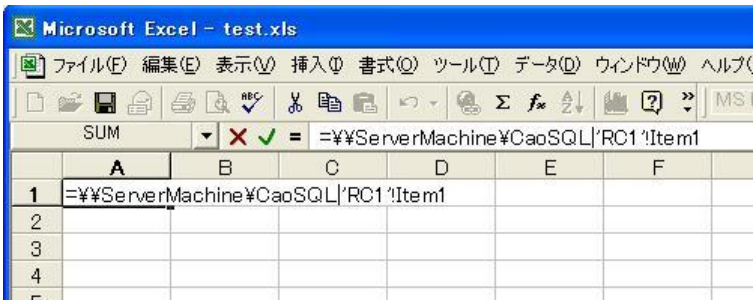


図 3-8 DDE 文字列の入力

ここで，ネットワーク DDE を使用する場合の文字列の形式を次に示します．

「＝‘¥¥マシン名¥アプリケーション名’|‘トピック名’!‘アイテム名’」

この形式は DDE 共有の設定によって変わってくるので注意して下さい．(下記参照)

【参考】

上記の設定例は，CaoSQL に限らず全てのアプリケーションに対して有効です．逆に，例えば CaoSQL の「DS1」というコントローラ名のアイテムだけを有効にしたい場合は，次のように設定して下さい．

- 共有名：「NetDS1$」
- アプリケーション名： (静的)「CaoSQL」，(古いスタイル／新しいスタイル)空白
- トピック名： (静的)「DS1」，(古いスタイル／新しいスタイル)空白

このときの DDE 文字列は下記のように設定します．NDDE$は固定値です．

‘¥¥<マシン名>¥NDDE$’|’NetDS1$’!’Item1’

【注意事項】

ネットワークDDEを動作させるためには上記以外にも注意しなくてはならない場合がありますので，次のサイトなどを参考にして下さい．

http://www.angelfire.com/biz/rhaminisys/ddeinfo.html

また，最低限下記のことは確認して下さい．

(1) サーバマシンで CaoSQL が起動していますか？

(2) Network DDE と Network DDE DSDM サービスはサーバ／クライアントの双方で起動していますか？

(3) 少なくともローカルでは動作しますか？

3.6. ログ機能(エラー ロギング)

ログ機能とはオブジェクトの生成や削除，CaoSQL のスタートやストップなどのなんらかの動作が行われた状態で，その記録する機能です．出力先は，コンソール，メッセージボックス，イベントビューア，デバッグビューア，テキストファイルの 5 種類を選択でき，この設定も CaoSQLConfig で行うことができます．

CaoSQL のログ出力タイミングは以下の通りです．

・ CaoSQL の初期化，終了エラー時.
・ コントローラのサンプリングスレッド，起動/終了，稼動/停止時.
・ コントローラ，アイテムのパラメータの誤りを発見した時.
・ CaoController オブジェクトの生成に失敗した時.
・ CaoController 起動時にタイムアウトした時.
・ CaoVariable オブジェクトの生成，値取得/設定の失敗時.
・ ヒストリ機能でデータベースへの接続に失敗した時.
・ ヒストリ機能でデータベースへの書き込みに失敗した時.
・ DDE サーバの起動に失敗した時.
・ DDE ホットリンク機能で Advise に失敗した時.
・ トリガが実行された時.

さらにエラーログ出力の場合，同じエラーが繰り返し出力される状態が考えられる為，CaoSQLConfig の**[アクション]→[オプション]→[ログ]**から**[同一エラーを出力しない]**のチェックを外すことによって，以下のエラーが繰り返し出力されないようにすることができます．但し，一度でも正常に処理が行われるとエラー状態がリセットされ，次回のエラーはログ出力されるようになります．この機能により，連続エラー出力によりログ出力が流れてしまうことを防ぐことが可能です．

・ CaoVariable の値取得/設定の失敗時.
・ ヒストリ機能でデータベースの書き込みに失敗した時.

3.7. CaoSQLのカスケード接続機能

ここでは CaoSQL 分散環境を構築する方法を説明します．

CaoSQL の分散環境を実現することにより，効率良く必要なデータを一台のマシンに収集することが可能となります．

当然 CAO の IF を使用することで同様の機能の実現は可能ですが，CaoSQL というミドルウェアを使用することで，以下のような恩恵を受けることができます．

・ CaoSQL プロバイダはコントローラのデータをキャッシュしていて，そのデータを参照することで，ロボットコントローラに不必要な負荷をかけずに済みます．
・ CaoSQL という統一した IF を使用することで，クライアントアプリケーションの実装負担を軽減できます．

CaoSQL 分散環境を実現する手段として以下の3つの構築方法を説明します．

・ CaoSQL プロバイダ
・ RAC プロバイダ
・ e-CAP プロバイダ
・ CoAP プロバイダ

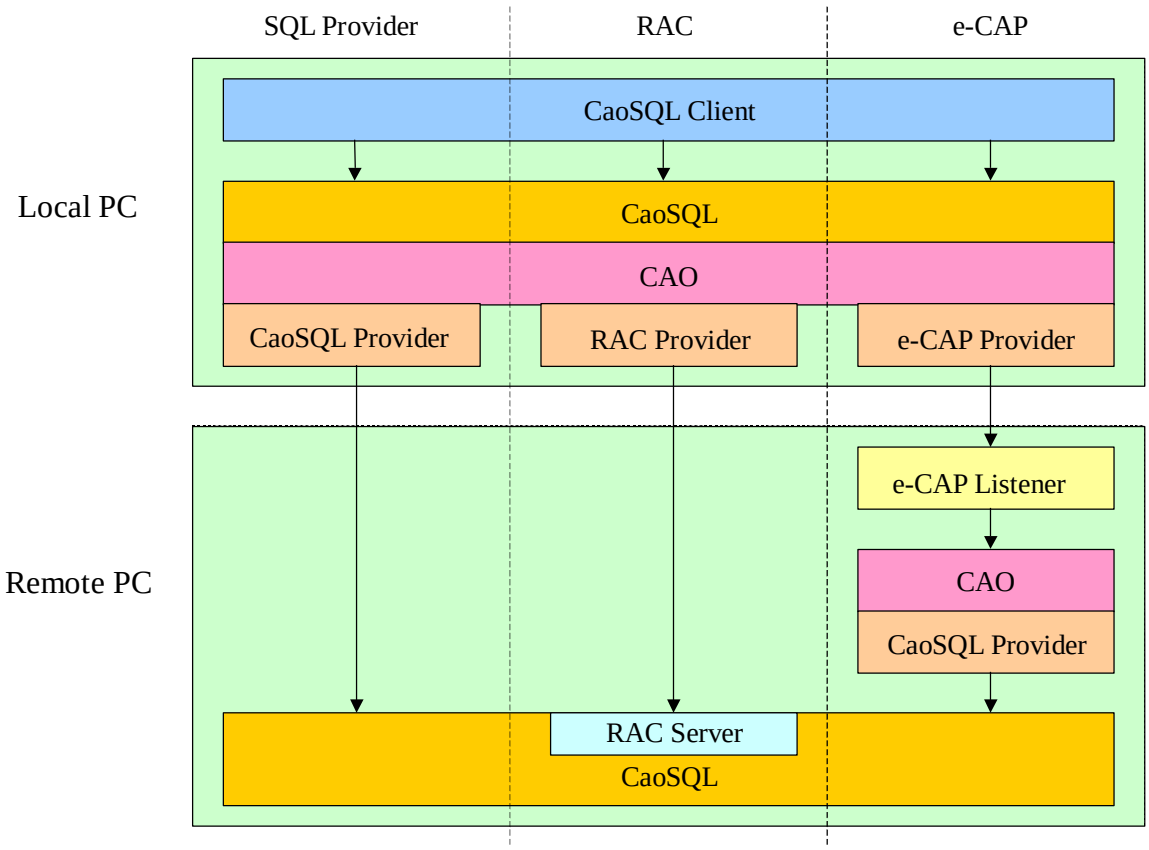


図 3-9 CaoSQL 分散環境設定概要

3.7.1. CaoSQLプロバイダ

CaoSQLプロバイダはあるマシンで稼動するCaoSQLの一つのコントローラをCAOのインタフェースを使用しCaoControllerとして取得することが可能で，またそのItemをCaoVariableとして取得することも可能です．詳しいCaoSQLプロバイダの使用方法については「CaoSQLTools ユーザーズガイド」を参照して下さい

CaoSQLプロバイダはDCOMを使用してリモートマシンのCaoSQLにアクセスするため，カスケード接続を使う為には，Microsoft社のDCOMモジュールに付属のdcomcnfg.exeツールを使ってセキュリティ関連の設定をしなくてはなりません[6]．この設定の目的は，「CaoSQL同士で通信を可能にし，且つCaoSQLが複数起動しないようにする」ことです．

まずは通信を可能にするために，dcomcnfg.exeで認証の設定を行います．CaoSQLプロバイダは，CaoSQLのイベントを使わないので基本的には下位の層の認証を「なし」に設定します．(図 3-10)ただし，9x系ではDCOMの仕様でリモートからのCaoSQLの起動ができないので，予めCaoSQLを起動しておく必要があります．[7]

[6] dcomcnfg.exeの詳細な利用方法は，「ORiN2 プログラミングガイド」を参照して下さい．
[7] このよう場場合はCaoProvExecツールが便利です．CaoProvExecツールについては「CAOプロバイダ作成ガイド」を参照して下さい．

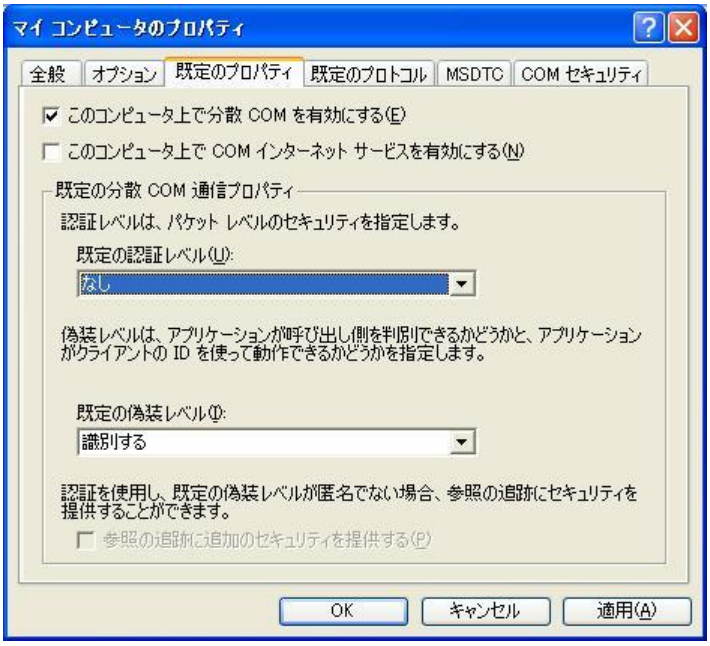


図 3-10 dcomcnfg－「既定のプロパティ」

次に，CaoSQLプロセスが複数起動しないようにプロセスのユーザを統一もしくは明示します． [CaoSQLのプロパティ]→[識別]で「対話ユーザ」を選択しておきます[8]．この設定では，CaoSQLプロセスは常にログオンしているユーザのプロセスで起動されるので複数起動されることはありません．同様に，「このユーザ」を選択してCaoSQLプロセスのユーザを明示的に統一しておくこともできます．

[8]仮に，「起動したユーザ」を選択すると，ログオンユーザがもしCaoSQLを既に起動していた場合，そのユーザと上位から起動したユーザ（この場合User0）が違うので下位で2つのCaoSQLプロセスが起動してしまい正しく通信することができないばかりでなく，負荷が倍になってしまうので注意して下さい．

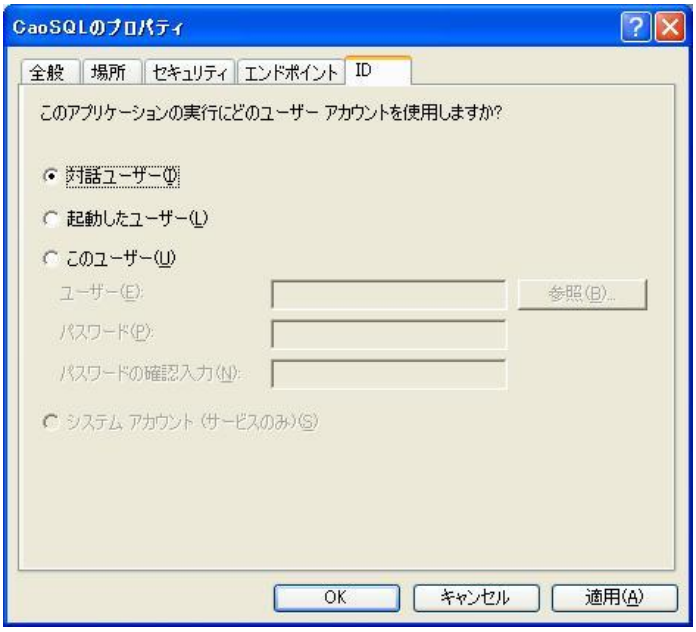


図 3-11 dcomcnfg－「CaoSQL のプロパティ」

3.7.2. RACプロバイダ

RACプロバイダはRACコマンドの送受信を行って，リモートマシンと通信するプロバイダです．詳しいRACプロバイダの使用方法については「RAC プロバイダガイド」を参照して下さい．

リモートマシン上の CaoSQL に RAC サーバ機能を使用することにより，カスケード接続を行うことが可能です．

3.7.3. e-CAPプロバイダ

e-CAP プロバイダは，HTTP を利用してリモートマシンと通信するプロバイダです．このカスケード接続の場合，ファイアーウォールを超えて接続することが可能です．

詳しいe-CAPプロバイダの使用方法については「e-CAP プロバイダガイド」を参照して下さい．

3.7.4. CoAP プロバイダ

CoAP プロバイダは CoAP プロトコルの送受信を行って，リモートマシンと通信するプロバイダです．詳しい CoAP プロバイダの使用方法については「CoAP プロバイダガイド」を参照して下さい．

3.8. ヒストリ機能(データ ロギング)

ヒストリ機能は，アイテムの値が変化したとき，もしくはSnapshotメソッドを明示的にコールしたとき[9]に，アイテムの値をデータベースに記録するデータロギング機能です．

ヒストリのデータベースの指定はCaoSQLConfigツールで設定します(詳細については，4.2.1.1 を参照して

[9] (注) Snapshot メソッドを使う場合は，自動的な記録と混在すると使いにくいので，[Auto History]の機能は OFF にしたほうがよいでしょう．これで Snapshot メソッドをコールしたときだけデータベースに記録されます．

下さい). ヒストリの最大レコード数を設定すると, 最大レコード数を超えた場合にヒストリへの記録を停止します[10].

対象とするデータベースは選択できますが, CaoSQL はデータベース, 及びテーブルを新規に作成しませんので, 必要に応じて新規にデータベース及びテーブルを作成する必要があります. データベースの構造を表 3-5 に示します. 具体的には次のSQLを実行することで作成できます.

```
CREATE TABLE caosql_history
(controller_name varchar(32) NOT NULL,
item_name varchar(32) NOT NULL,
item_value varchar(256),
item_vartype short NOT NULL,
item_time datetime NOT NULL,
item_time_ms long NOT NULL,
flags short NOT NULL)
```

このデータログ機能は, コントローラ毎またはアイテム毎に OFF することができます. CaoSQL はコントローラ単位で独立のスレッドがサンプリングを行っていますが, ログ出力は一つのデータベースに出力するのでその際にシリアライズされて, その結果サンプリングのパフォーマンスが落ちることがあります. また, そもそもログ出力はオーバーヘッドの大きな処理ですので, 不必要なコントローラまたはアイテムのログ出力をしないように設定して下さい.

表 3-5 ヒストリテーブルの構造

フィールド名	データ型	長さ(Byte)	備考
controller_name	varchar NOT NULL	32	対象アイテムの属するコントローラ名 32byte 以上のデータの場合エラーが発生します.
item_name	varchar NOT NULL	32	対象アイテム名 32byte 以上のデータの場合エラーが発生します.
item_value	varchar	256	変化後の値を文字列化して記録します. (256byte 以上のデータは 256byte まで記録されます.)
	(注)記録するすべてのデータの型を一つに限定できる場合などはデータ型をその型にすることができます. その場合は, CaoSQL によって文字列変換されないように「文字列変換しない」オプションを有効にしてください.		
item_vartype	short NOT NULL	2	変化後の値の型を数値[11]で記録します.

[10] ログ出力のタイミングで毎回検査されるのでオーバーヘッドが多くなります. 通常はレコード数の制限は OFF にし, データベースエンジンの方で容量制限などを設定して下さい.

[11] 各型の数値は VT_EMPTY=0, VT_I2 = 2, VT_I4 = 3, VT_R4 = 4, VT_R8 = 5, VT_CY = 6, VT_DATE = 7, VT_BSTR = 8, VT_DISPATCH = 9, VT_ERROR = 10, VT_BOOL = 11, VT_UNKNOWN = 13, VT_I1 = 16, VT_UI1 = 17, VT_UI2 = 18, VT_UI4 = 19 と

item_time	datetime NOT NULL	8	値が変化した時間を記録します.
item_time_ms	long NOT NULL	4	値が変化した時間(ms)を記録します.
flags	short NOT NULL	2	レコードの種類を記録します. 0:履歴データ, 1:最新データ[12]

CaoSQLは, ADO(ActiveX Data Object)を用いてデータベースの処理を実行しています. したがって, ADOのプロバイダ(OLE-DBプロバイダ)があるデータベースシステムであれば, 基本的にはCaoSQLで使えるはずです. データベースの動作確認は, 以下を行っています。

SQLServer, Oracle, Access, Excel, CSVファイル, MySQL, PostgreSQL

表 3-6 に, 主なデータベースシステムの接続文字列を示します.

表 3-6 データベースの設定内容一覧

データベース	テーブル名	指定オプション	入力内容
SQL Server	テーブル名	Provider	“SQLOLEDB.1”
		Data Source	SQL サーバ名
		Initial Catalog	データベース名
		User ID	データベースのユーザ ID
		Password	ユーザのパスワード
Oracle	テーブル名	Provider	“MSDATAORA”
		Data Source	Oracle サーバ名
		Initial Catalog	データベース名
		User ID	データベースのユーザ ID
		Password	ユーザのパスワード
Acess97	テーブル名	Provider	“Microsoft.Jet.OLEDB.3.51”
		Data Source	ファイル名(*.mdb)
Acess2000	テーブル名	Provider	“Microsoft.Jet.OLEDB.4.0”
		Data Source	ファイル名(*.mdb)
Excel2000[13]	[<シート名>$] 又は	Provider	“Microsoft.Jet.OLEDB.4.0”

なります.

[12] CaoSQLConfig の“Table Type”の設定で, “History”を選択すると履歴データのみが記録され, “Current Value”を選択すると最新データのみが記録され, “History & Current Value”を選択すると履歴データと最新データの両方が記録されます.

[13] シートには, 表 3-5 のフィールド名が入力されたタイトル行が必要です.

	[<シート名>$<開始セル>:<終了セル>][14] 例:[Sheet1$A1:G31]	Data Source	ファイル名(*.xls)
		Extended Properties	“Excel 8.0”[15]
CSVファイル[16]	CSVファイル名[17]	Provider	“Microsoft.Jet.OLEDB.4.0”
		Data Source	CSV ファイルの存在するフォルダパス
		Extended Properties	“Text”
MySQL	テーブル名	Provider	MSDASQL
		Misc.	"Driver={PostgreSQL Unicode}; Server=サーバ名; Port=ポート番号; Database=スキーマ名; UID=データベースのユーザ ID; PWD=ユーザのパスワード"
PostgreSQL	テーブル名	Provider	MSDASQL
		Misc.	"Driver={MySQL ODBC 8.0 Unicode Driver}; Server=サーバ名; Port=ポート番号; Database=データベース名; UID=データベースのユーザ ID; PWD=ユーザのパスワード"

3.8.1. システム情報記録

ヒストリ機能では、サンプリングの情報をデータベースに記録することが出来ます。

ヒストリデータの<controller_name>は、サンプリングを行っているコントローラ名、< item_time >, < item_time_ms>は、記録した時間が保存されます。< flags >は、アイテム値のレコードと同じです。

レコードに記録される種別及び上記以外のフィールド値については、表 3-7 を参照してください。

表 3-7 システム情報一覧

item_name	item_value	item_vartype	説明

[14] 指定したシート名，及びセル範囲指定を大括弧“[ ]”で囲む必要があります.
[15] “Excel”と“8.0”の間に空白が入ります.
[16] CSV ファイルの場合，“UPDATE”と“DELETE”は実行できません. また，CSV ファイルの 1 行目には列名をカンマ区切りで記述して下さい.
[17] CSV ファイルパスではなく，CSV ファイル名のみを設定します.

$$START$$	-	0 (VT_EMPTY)	サンプリングの開始
$$STOP$$	-	0 (VT_EMPTY)	サンプリングの停止
$$BEGIN$$	起動してからの更新回数	3 (VT_I4)	値の更新開始
$$END$$	起動してからの更新回数	3 (VT_I4)	値の更新終了
$$OPEN$$	-	0 (VT_EMPTY)	データベースのオープン
$$CLOSE$$	-	0 (VT_EMPTY)	データベースのクローズ
$$SWITCH$$	-	0 (VT_EMPTY)	データベースのプリマリからセカンダリへの切り替え

3.8.2. アイテム値の文字列変換

アイテム値は、文字列に変換されてレコードに記録されます。

このとき、配列データは、カンマ(“,”)で区切られた文字列に変換されます。

記録するデータの型が 1 種類のみである等の理由により、<item_value>フィールドの型を限定する場合は、この機能を無効にしてください。

3.8.3. バイナリデータのファイル保存

ヒストリ機能では、バイナリデータをファイルに保存することが出来ます。

ファイル保存を行うと、レコードに記録されるアイテム値には保存ファイル名が記録されます。

保存ファイル名は、レコードのフィールドを使って以下のように命名されます。

履歴データ　　<controller_name>_<item_name>_<item_time>_<item_time_ms>.dat

最新データ　　<controller_name>_<item_name>_Current.dat

<item_time>の値には、デリミタは付加しません。(例:2015/07/01 10:11:12 → 20150701101112)

ファイルの保存先となるフォルダは、CaoSQL の設定ファイルと同じ場所に作成されます。保存フォルダ名は、設定ファイルから拡張子を取り除いてものが使用されます。

この機能では、ファイルの最大数またはファイルサイズによる制限はありません。

このため、ファイルの数やディスクの使用量に制約を付ける場合は、以下の方法を検討してください。

- ヒストリ機能の最大レコード数
- データベースによるサイズ制限
- ディスククォータによるドライブの使用量

3.9. セカンダリDB切替え機能

データベースはプライマリとセカンダリの 2 つのデータベースを設定することができます. ヒストリ機能を使っ

てアイテムの値をデータベースに記録している最中にデータベースサーバの異常で記録できない場合に，自動的にセカンダリデータベースに切替えて記録を継続することができます．これにより，たった一つのデータも取りこぼすことなく記録できます．

典型的な使い方は，プライマリにネットワーク上のデータベースサーバ(例えば，SQL-Server)，セカンダリにローカルのデータベースサーバ(例えば，Access)という組合せでしょう．こうしておけば，何かネットワークトラブルが発生してもデータを取りこぼすことがなくなります．

データベースの切替えは，プライマリからセカンダリへの一方向のみで一度セカンダリに切替えられたらプライマリに戻ることはできません．あくまでもセカンダリは非常用であって，そのような事態が発生しないようにデータベースサーバをメンテナンスするのが基本です．

また，CaoSQLHistory クラスの Execute メソッド(任意の SQL 文を実行する関数)を使って，クライアントプログラムから任意の SQL 文を実行して仮にエラーが発生しても自動的に切替えることはしません．これは，システムではデータベースサーバの異常なのか SQL 文の問題なのかの判別が難しい為です．

3.10. 配列要素抽出機能

アイテムの値が配列の場合に，必要な要素だけを抽出することがでます．抽出したい要素のインデックス番号をカンマ区切りで()で括ってアイテム名に付加します．3 次元まで指定可能です．

(例) Item1(2,3)

抽出できる型は，VT_BOOL，VT_I1，VT_UI1，VT_I2，VT_UI2，VT_I4，VT_UI4，VT_R4，VT_R8，VT_CY，VT_DATE，VT_BSTR，VT_VARIANT の 13 のデータ型です．

(注)この機能を使うと配列要素毎にマスク処理や BCD 変換などのデータ加工処理が行え，同じく要素毎にリンク先を分けるなどの処理が可能になります．しかし逆に，そのような機能は不要で単純に配列の一要素だけを取得したいときは，そのまま配列として取得してクライアントアプリケーション側で要素を抽出した方が効率的な場合があります．

3.11. データのマスキング機能

アイテムの値をマスキングします．必要なビットだけを抽出したい場合にマスクを設定すると自動的にマスキングされます．また，読込み・書き込み共にマスキングされます．BCD変換機能，二項演算機能と同時に使った場合，マスキング処理→BCD変換→二項演算の順で処理されます．

この機能が有効なのは配列ではない 1 バイト，2 バイト，4 バイトの符号付き・符号なし整数のみです．具体的には，VT_I1，VT_UI1，VT_I2，VT_UI2，VT_I4，VT_UI4 の 6 つのデータ型です．

3.12. BCD変換機能

アイテムの値をBCD(Binary Code Decimal)変換します．また，読込み・書き込み共に変換されます．マスキング機能，二項演算機能と同時に使った場合，マスキング処理→BCD変換→二項演算の順で処理されます．

この機能が有効なのは配列でない 1 バイト，2 バイト，4 バイトの符号付き・符号なし整数のみです．具体的には，VT_I1，VT_UI1，VT_I2，VT_UI2，VT_I4，VT_UI4 の 6 つのデータ型です．

3.13. 二項演算機能

二項演算機能を使うと自アイテムの取得した値と別のアイテムの値との間で二項演算することができ，演算した結果が自アイテムの値としてキャッシュされます．つまり，

<自アイテムの値> ← <自アイテムの値> <演算子> <(別)アイテム>

という処理が行われます．[18]この変換処理は読込み時のみ実行されます．ここで使える<演算子>には，加算(+)，減算(-)，掛算(*)，割算(/)，割算の剰余(MOD)，論理積(AND)，論理和(OR)，排他的論理和(XOR)の 8 つの演算子があります．マスキング機能，BCD変換機能と同時に使った場合，マスキング処理→BCD変換→二項演算の順で処理されるので注意して下さい．

また<(別)アイテム>の書式は，

“[<Controller Name>¥]<Item Name>” または “#<定数>”

となっています．コントローラ名が省略された場合は，その自アイテムと同じコントローラと判断されます．また，定数は倍精度実数として扱われます．“#100”(10 進)，“#&H10000”(16 進)などの表記が可能です．

例えば，「ある品番で且つあるIOがオンになったときに．．．」というような条件では，品番アイテムにIOアイテムとの AND 演算を設定しておくと上位クライアントは一つのアイテムだけで条件を判定することができます．

この二項演算機能の実装にはoleaut32.libの表 3-8 の関数を用いているので，異なる型同士の演算結果など詳細についてはMicrosoftのドキュメントを参照して下さい．

一つのアイテムに対しては二項演算しかできませんが，複雑な演算も全て二項演算に分解できるように，リンク機能と併用すれば複雑な演算も可能になります．詳しくは3.17 を参照して下さい．

表 3-8 二項演算機能の内部関数

	演算	表記	戻り値	内部処理関数
1	加算	+	(演算結果)	VarAdd()
2	減算	-	(演算結果)	VarSub()
3	掛算	*	(演算結果)	VarMul()
4	割算	/	(演算結果)	VarDiv()
5	割算の剰余	MOD	(演算結果)	VarMod()
6	論理積	AND	(演算結果)	VarAnd()
7	論理和	OR	(演算結果)	VarOr()
8	排他的論理和	XOR	(演算結果)	VarXor()
9	ビット左シフト	<<	(演算結果)	<<
10	ビット右シフト	>>	(演算結果)	>>
11	等しい	=	True / False	VarCmp()
12	等しくない	<>	True / False	VarCmp()

[18]設定を間違えると演算処理が無限に行われる可能性がありますので，設定は慎重に行って下さい．

13	以下	<=	True / False	VarCmp()
14	より小さい	<	True / False	VarCmp()
15	以上	>=	True / False	VarCmp()
16	より大きい	>	True / False	VarCmp()
17	文字列連結	CAT	(演算結果)	VarCat()
18	整数化割算	IDIV	(演算結果)	VarIdiv()
19	論理包含演算	IMP	(演算結果)	VarImp()
20	累乗	POW	(演算結果)	VarPow()
21	絶対値	ABS	(演算結果)	VarAbs()
22	整数部分	FIX	(演算結果)	VarFix()
23	整数部分	INT	(演算結果)	VarInt()
24	数値否定(符号反転)	NEG	(演算結果)	VarNeg()
25	ビット反転	NOT	(演算結果)	VarNot()
26	単位変換(ラジアン→度)	RAD2DEG	(演算結果)	-
27	単位変換(度→ラジアン)	DEG2RAD	(演算結果)	-
28	文字列長	LEN	(演算結果)	-
29	部分文字列(LEFT)	LEFT$	(演算結果)	-
30	部分文字列(RIGHT)	RIGHT$	(演算結果)	-

3.13.1. 第二演算機能[19]

Ver. 1.8.0 以降では, 第二演算を実行することができます. 第二演算の書式や演算処理については上記と全く同じです. 第二演算は第一演算の結果に対して実行されます. この第二演算機能を用いるとこで, 下記のような変換も用意に表現できるようになります.

X ← AX + B ‘ 第一演算子＝’*’, 第二演算子＝”+”

X ← $(AX)^B$ ‘ 第一演算子＝’*’, 第二演算子＝”POW”

3.14. データ型変換機能

アイテムのデータ型を指定した型に変換する機能です. 多次元配列にも対応しています. データ型変換は読込み時／書込み時で別々に設定することができます.

型変換可能な型は, VT_BOOL, VT_I1, VT_UI1, VT_I2, VT_UI2, VT_I4, VT_UI4, VT_R4, VT_R8, VT_CY, VT_DATE, VT_BSTR, VT_VARIANT の 13 のデータ型です.

3.15. 不感帯設定機能

値の変化と見なさない範囲を最小値・最大値で指定します. この範囲内の値の変化は値の変化と見なされませんので, 値変化に同期した機能が実行されません. 値変化に同期した機能とは次の 4 つの機能で

[19] 本機能は Ver. 1.8.0 以降で利用可能です.

す.

- イベント通知機能(OnChangeItem イベント)
- アイテムリンク機能
- トリガ機能
- ヒストリ(データロギング)機能

この機能が有効なのは配列ではない 1 バイト, 2 バイト, 4 バイトの符号付き・符号なし整数のみです. 具体的には, VT_I1, VT_UI1, VT_I2, VT_UI2, VT_I4, VT_UI4, VT_R4, VT_R8 の 8 つのデータ型です. データ型が変化した場合は, 無条件に値が変化したと判定されます.

3.16. 感帯設定機能

値の変化と見なす範囲を最小値・最大値で指定します. この範囲外の値の変化は値の変化と見なされませんので, 値変化に同期した機能が実行されません. 値変化に同期した機能とは次の 4 つの機能です.

- イベント通知機能(OnChangeItem イベント)
- アイテムリンク機能
- トリガ機能
- ヒストリ(データロギング)機能

この機能が有効なのは配列ではない 1 バイト, 2 バイト, 4 バイトの符号付き・符号なし整数のみです. 具体的には, VT_I1, VT_UI1, VT_I2, VT_UI2, VT_I4, VT_UI4, VT_R4, VT_R8 の 8 つのデータ型です. データ型が変化した場合は, 無条件に値が変化したと判定されます.

3.17. アイテムリンク機能

アイテムリンク機能とは, アイテムの値が変化したとき, そのアイテムから別のアイテムにリンクが設定してあると, そのリンク先のアイテムに値を伝播させる機能です. これは, あたかも IO の結線をソフト的に実施するような機能でもあります.

例えば, RC1 のアイテム I1 が RC2 のアイテム I1 にリンクが張られている場合, RC1 の I1 が 0 から 1 に変化すると, CaoSQL はそれを検知し, RC2 のアイテム I1 に 1 を書き込みます.

また CaoSQLConfig で設定した Link 先情報は CaoSQLItem のインタフェースで取得できます. 書式は

[<Controller Name>¥]<Item Name>

となっています. コントローラ名が省略された場合は, その自アイテムと同じコントローラと判断されます. アイテムを複数指定する場合はカンマ(“**,**”)で区切ります.

デフォルトでは全てのリンク先アイテムに同じ値を伝播させます. 配列要素分配リンク機能(Extract Linking オプション)が有効で, 且つそのアイテムが配列データであった場合は配列要素を抽出してそれぞれの値をリンク先のアイテムに順に伝播させます.

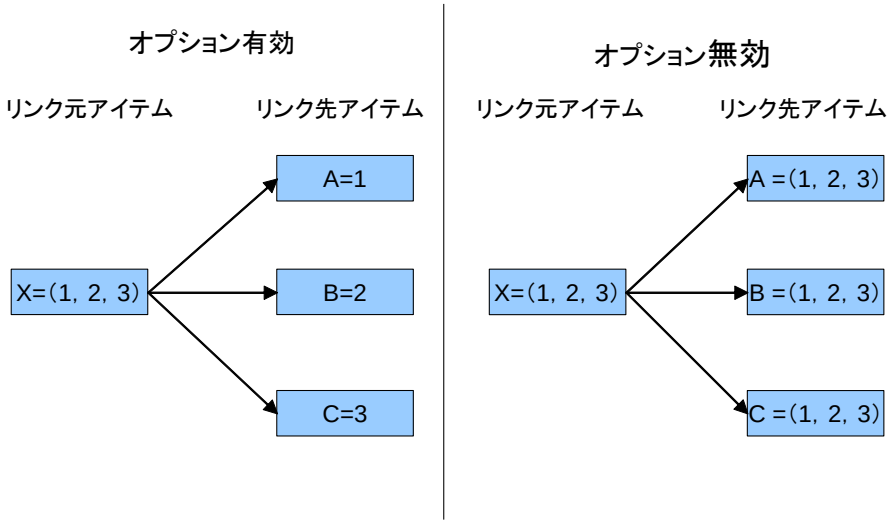


図 3-12 配列要素分配リンク機能

このリンク機能は特別なプログラミングを必要としないで値を伝播させることができるという意味で非常に便利な機能ですが，リンク設定を間違えると無限伝播が発生する可能性がありますので，設定は慎重に行う必要があります．例えば，図 3-13 に示すような設定でリソースαとリソースβが同時に異なる値になった場合，相互リンクにより無限に処理がループしてしまいます．これはCaoSQLのインタフェースを使ってプログラミングする場合も同様です[20]．

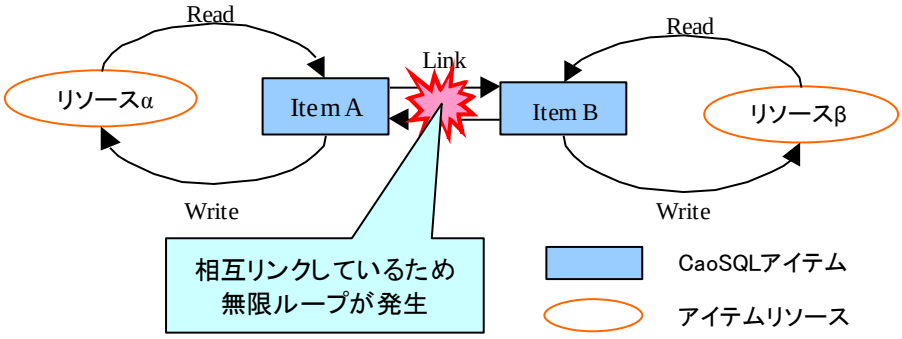


図 3-13 相互リンクによる無限ループ

3.17.1. リンク先プロパティ選択

デフォルト設定では，リンク先アイテムの Value プロパティにリンクされます．アイテムの[詳細設定]で，このプロパティを選択することができます．選択できるプロパティは，Value (default)，ID，Activation，LinkActivation の 4 つです．

例えば，ある変数オブジェクト A（CaoVariable）が ID プロパティで指定された配列要素を取得できるような

[20] 物理的な IO 結線でも無限ループになるような結線は異常であるように，CaoSQL のソフト結線でも同じく異常です．

プロバイダの場合，この機能を活用すると，別の変数オブジェクト B で A の ID を指定することができるので，B の値と連動して A の要素番号を切り替えることができます．

3.18. 不活性化機能

アイテムの不活性化機能を有効にした場合，コントローラのサンプリング周期時にアイテムの読込み，または書込みを行わないようになります．CaoSQL 実行時に不活性化を有効にしたい場合は，Activation プロパティを有効にします．この機能のデフォルト値は無効です．

3.19. 配列型アイテム機能

配列型アイテム機能は，アイテムの読込みを行った時に，リンク設定をしているアイテムへ値を配列の要素として伝播します．書き込みを行った場合は，同じくリンク設定をしているアイテムの値を配列の要素として，配列を形成して値が書込まれます．

ただし，リンク設定しているアイテムの順序は，意図する配列の順序にならないことがあります．この機能を使用する場合，ソート機能を使って昇順でリンク設定をおこなってください．

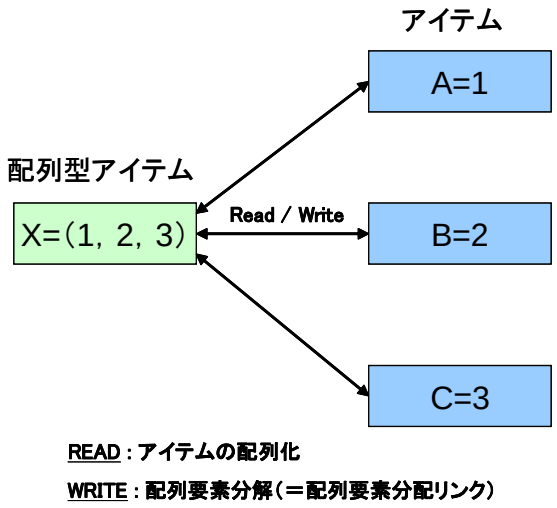


図 3-14 配列型アイテム機能

3.20. エイリアス機能

2.2 オブジェクトモデルでも述べているように，コントローラ，アイテムのそれぞれにエイリアス機能を設定することができます．エイリアス機能とは，登録してあるアイテムを別名で登録を行い，値のみ伝播する機能です．エイリアスに設定する伝播先情報はCaoSQLItemのインタフェースで取得できます．書式は

[<Controller Name>¥]<Item Name>

となっています．リンクの設定ができない配列型アイテムの値を伝播する場合は，このエイリアス機能を使用する必要があります．

3.21. RACサーバ機能[21]

RAC サーバ機能とは，RAC プロバイダを利用しネットワークを介して CaoSQL とデータのやり取りを行う機能です．RAC サーバを複数設定することも可能です．

RACサーバ機能の設定は，CaoSQLConfig.exeのメニューから**[アクション]→[オプション]**を選択し，[**RAC**]タブで行います．(4.2.2.3 参照)設定内容はRACプロバイダのAddControllerのオプション文字列を指定します．現在，サーバ機能のみが実装されていますので，サーバモードの設定を入力して下さい．[22] 詳しいRACプロバイダの使用方法については「RAC プロバイダガイド」を参照して下さい．

以下に RAC サーバ機能の設定例の画面を示します．

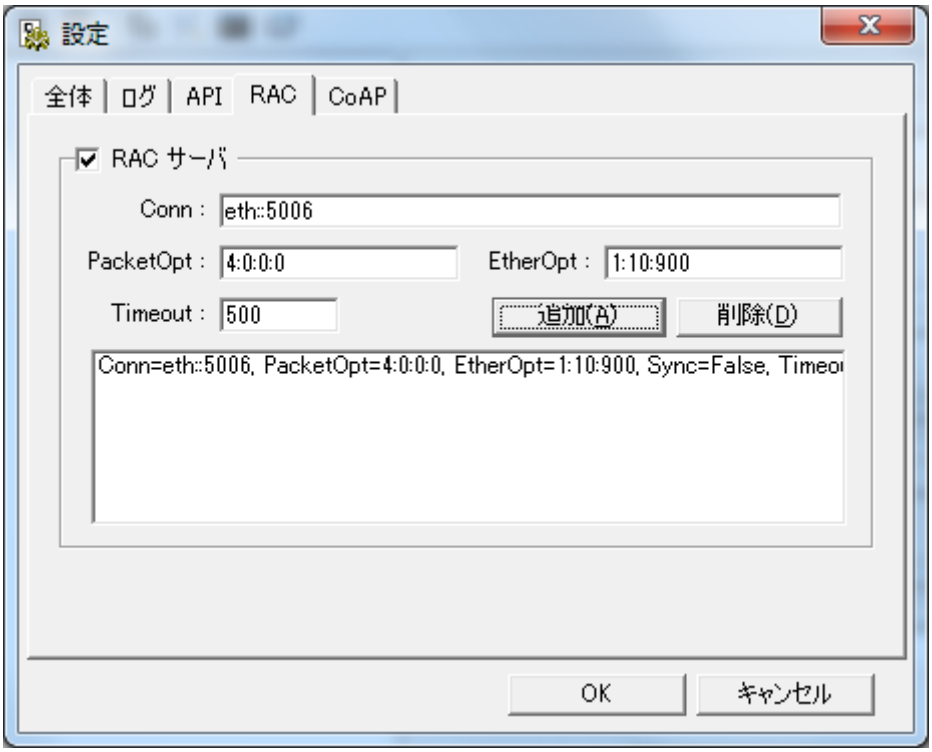


図 3-15 RAC サーバの設定例

次に使用できる RAC コマンドについて説明します．

3.21.1. GETコマンド

コマンドの書式は以下の通りです．[23]

GET:[＜コントローラ名＞]:[<アイテム ID>]:[＜アイテム名＞][！＜プロパティ名＞]:

指定アイテムのプロパティ値を取得します．<アイテム ID>が指定されている場合は取得の直前に ID を設

[21] 本機能は，(財)機械振興協会 技術研究所が競輪の補助により実施した研究成果の一部を使用しています．
[22] 具体的には，接続パラメータを **Ethernet** に設定，Ether パラメータを**サーバ**に設定，パケットパラメータの **unicode 変換を ON** に設定します．サーバモードは自動的に非同期モードになりますので，Sync オプションは関係ありません．
[23] (注)コントローラ名とアイテム名の間には“:”が 2 つ,アイテム名の後に“:”が 1 つ必要です．

定(put_ID)します. [24]

コントローラ名を省略した場合, デフォルトコントローラ名が使用されます.

使用可能なエンジンおよびコントローラのプロパティー覧は表 3-9 を参照してください.

表 3-9 RAC サーバ機能 エンジン／コントローラプロパティー覧

プロパティ名	説明	GET	PUT
ControllerNames	コントローラ名一覧を取得します. コントローラ名に"$ENGINE$"を指定した場合のみ有効です.	○	×
ItemNames	指定したコントローラのアイテム名一覧を取得します.	○	×
Description	指定したコントローラの Description プロパティの値を取得します.	○	×
State	指定したコントローラの State プロパティの値を取得します.	○	×

使用可能なアイテムのプロパティー覧は表 3-10 を参照して下さい. 省略した場合はValueを使用します.

表 3-10 RAC サーバ機能 アイテムプロパティー覧

プロパティ名	説明	GET	PUT
Value	アイテム値	○	○
Attribute	属性値	○	×
Datetime	タイムスタンプ(日時)	○	×
Description	説明	○	×
Microsecond	タイムスタンプ(マイクロ秒)	○	×
State	状態	○	×

```
例 1) GET:Ctrl1::Item1:              // コントローラ(Ctrl1)のアイテム(Item1)の値を取得
例 2) GET:::Item2!Value:             // デフォルトコントローラのアイテム(Item2)の値を取得
例 3) GET:Ctrl2::Item1!Attribute:  // コントローラ(Ctrl2)のアイテム(Item1)の属性値を取得
例 4) GET:$ENGINE$::!ControllerNames:        // コントローラ名の一覧を取得
例 5) GET:Ctrl1::!ItemNames:         // コントローラ(Ctrl1)のアイテム名の一覧を取得
```

3.21.2. PUTコマンド

コマンドの書式は以下の通りです. [25] [26]

PUT:[＜コントローラ名＞]:[<アイテム ID>]:＜アイテム名＞[！＜プロパティ名＞]:＜型, 値＞

[24] 自動サンプリングがアクティブな状態では, ID 設定しても値が更新されていない可能性もありますので注意してください.

[25] (注)コントローラ名とアイテム名の間には:が 2 つ必要です.

[26] 「型, 値」の記述方式の詳細は,『ORiN2 プログラミングガイド』の「2.2.5.データの記述方式」を参照して下さい.

指定アイテムのプロパティに値を設定します. <アイテム ID>が指定されている場合は設定の直前に ID を設定(put_ID)します. [24]

コントローラ名を省略した場合, デフォルトコントローラ名が使用されます.

使用可能なプロパティ名は表 3-10 を参照して下さい. 省略した場合はValueを使用します.

例) PUT:Ctrl1::Item1:8,testok // コントローラ(Ctrl1)のアイテム(Item1)に BSTR 型で“testok”を設定

3.22. CoAP サーバ機能

CoAP サーバ機能とは, CoAP プロバイダを利用しネットワークを介して CaoSQL とデータのやり取りを行う機能です. CoAP サーバを複数設定することも可能です.

CoAPサーバ機能の設定は, CaoSQLConfig.exeのメニューから**[アクション]→[オプション]**を選択し, [CoAP]タブで行います. (4.2.2.3 参照)設定内容はCoAPプロバイダのAddControllerのオプション文字列を指定します. 詳しいCoAPプロバイダの使用方法については「CoAPプロバイダガイド」を参照して下さい.

以下に CoAP サーバ機能の設定例の画面を示します.

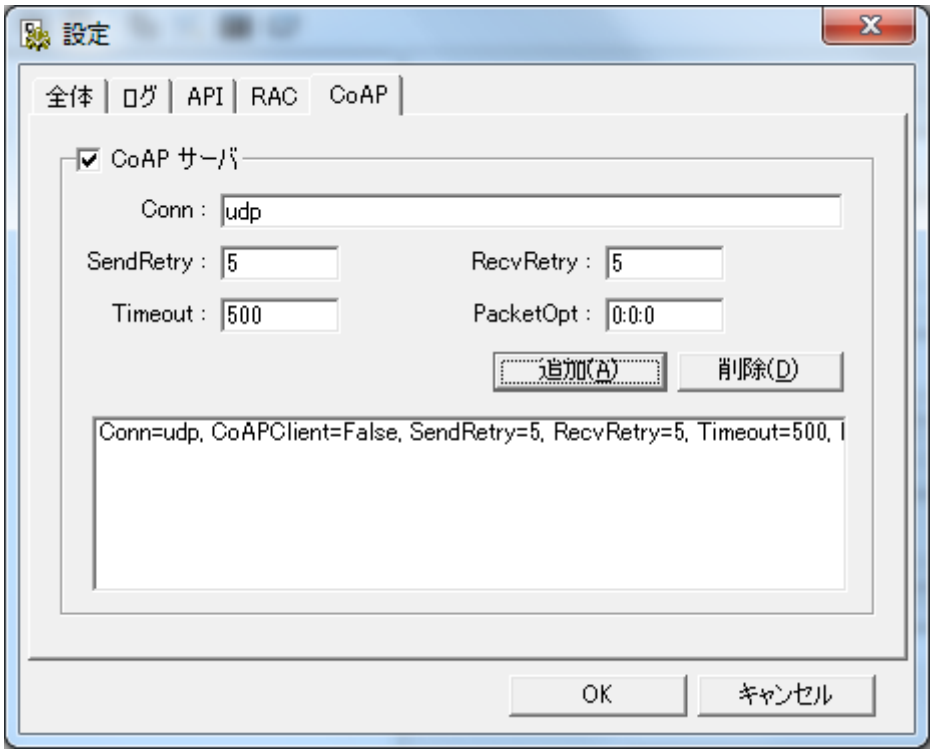


図 3-16 CoAP サーバの設定例

次に使用できる CoAP コマンドについて説明します.

3.22.1. GETコマンド

CoAPプロトコルでCaoSQLの指定アイテムのプロパティ値を取得するには以下のパケットを送信します.

[Code] **Get (1)**
[Option] **値:URI パス (11), データ:[<コントローラ名>/]<アイテム名>**

コントローラ名を省略した場合, デフォルトコントローラ名が使用されます.

3.22.2. PUTコマンド

CoAPプロトコルでCaoSQLの指定アイテムのプロパティに値を設定するには以下のパケットを送信します.

[Code] **Put (2)**
[Option] **値:URI パス (11), データ:[<コントローラ名>/]<アイテム名>**
[Payload] **VT_VARIANT**

コントローラ名を省略した場合, デフォルトコントローラ名が使用されます.

3.23. アイテム動的登録/削除機能

アイテム動的登録/削除機能とは, CaoSQL 起動後にアイテムを登録/削除できる機能です. 但し, 動的削除は動的登録したアイテムにのみ, 可能です.

アイテム動的登録は, コントローラの AddItem メソッドに各データを格納した VARIANT 配列を渡すことにより実行されます. 配列の各要素には格納するデータが以下のように決まっています. (配列の最小添え字が 0 であるとします.)

表 3-11 AddItem メソッドへの引数

配列要素番号	データ型	データ項目名
0	VT_BSTR	アイテム名
1	VT_BOOL	CaoVariable 有無設定
2	VT_BSTR	CaoVariable 名
3	VT_BSTR	CaoVariable の作成時オプション

4	VT_I4	CaoVariable の親クラス種別 0 Controller クラス 1 Task クラス 2 Robot クラス 3 Extension クラス 4 File クラス
5	VT_BSTR	CaoVariable の親オブジェクト名
6	VT_BSTR	詳細情報
7	VT_I4	読み書き属性 0 読込み / 書き込み 1 読込み 2 書き込み
8	VT_BSTR	リンクアイテム設定
9	VT_BOOL	初期化有無設定
10	VT_BSTR	初期化値
11	VT_I4	初期化型 0 VT_BOOL 1 VT_I1 2 VT_UI1 3 VT_I2 4 VT_UI2 5 VT_I4 6 VT_UI4 7 VT_R4 8 VT_R8 9 VT_BSTR 10 VT_DATE 11 VT_CY
12	VT_BOOL	VT_EMPTY 使用有無設定
13	VT_BOOL	History 使用有無設定
14	VT_BOOL	OnChange イベント有無設定
15	VT_I4	値強制認識有無設定 0 なし 1 読み込み時 2 書き込み時
16	VT_BOOL	不感帯有無設定
17	VT_R4	不感帯最小値
18	VT_R4	不感帯最大値
19	VT_BOOL	感帯有無設定
20	VT_R4	感帯最小値
21	VT_R4	感帯最大値
22	VT_BOOL	チャタリング有無設定

23	VT_I4	チャタリング値
24	VT_BOOL	マスキング有無設定
25	VT_I4	マスク値
26	VT_BOOL	BCD 変換有無設定
27	VT_I4	二項演算処理演算子(第一演算) 0 (NOP) 1 + 2 - 3 * 4 / 5 MOD 6 AND 7 OR 8 XOR 9 << 10 >> 11 = 12 <> 13 <= 14 < 15 >= 16 > 17 CAT 18 IDIV 19 IMP 20 POW 21 ABS 22 FIX 23 INT 24 NEG 25 NOT 26 RAD2DEG 27 DEG2RAD
28	VT_BSTR	二項演算処理演算値設定(第一演算)
29	VT_BOOL	データ型変換処理有無設定(読込み時)
30	VT_I4	データ型変換処理のデータ型設定(読込み時) 0 VT_BOOL 1 VT_I1 2 VT_UI1 3 VT_I2 4 VT_UI2 5 VT_I4 6 VT_UI4 7 VT_R4 8 VT_R8 9 VT_BSTR 10 VT_DATE 11 VT_CY 12 VT_VARIANT
31	VT_BOOL	データ型変換処理有無設定(書込み時)

32	VT_I4	データ型変換処理のデータ型設定(書込み時) 0 VT_BOOL 1 VT_I1 2 VT_UI1 3 VT_I2 4 VT_UI2 5 VT_I4 6 VT_UI4 7 VT_R4 8 VT_R8 9 VT_BSTR 10 VT_DATE 11 VT_CY 12 VT_VARIANT
33	VT_BOOL	配列内データ比較有無設定
34	VT_BOOL	配列要素分配リンク設定
35	VT_BOOL	不活性化
36	VT_I4	アイテムタイプ 0 通常アイテム 1 エイリアスアイテム 2 配列型アイテム
37	VT_I4	リンク先プロパティ 0 Value 1 ID 2 Activation 3 LinkActivation
38	VT_I4	二項演算処理演算子(第二演算)
39	VT_BSTR	二項演算処理演算値設定(第二演算)
40	VT_BSTR	CaoVariable の親オブジェクトのオプション文字列

3.24. スクリプト実行機能[27]

スクリプト実行機能とは，VB スクリプトを実行する機能で，スクリプトはコントローラ単位で設定・実行することができます．起動時に CaoSQLConfig(図 3-17)で設定された VB スクリプトを読み込み，サンプリング毎にスクリプト内のサブルーチン Main を実行します．

[27] 本機能は Microsoft Windows 7 (64bit 版)では動作しません．

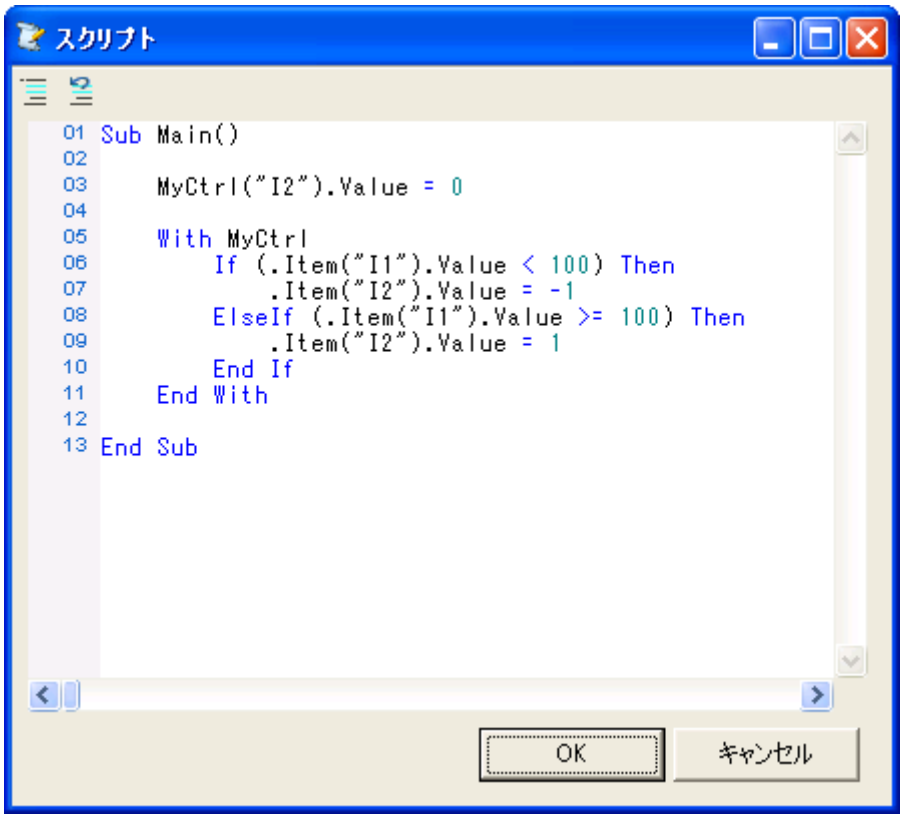


図 3-17 スクリプト画面

VB スクリプトは，Microsoft VBScript 仕様に完全準拠していますが，MsgBox，InputBox 等の UI 系関数と，CreateObject，GetObject 等のオブジェクト取得系関数は，CaoSQL のミドルウェアという性質と，スクリプトの安全性の理由で制限されています．

逆に，”MyCtrl”という組込みオブジェクトが追加されており，それはスクリプトを設定した CaoSQLController オブジェクト自身です．したがって，下記のように使えば演算結果をアイテムの値に容易に設定することができます．

例） 自コントローラのアイテム“I2”の値の 3 倍を”I1”にコピーする

MyCtrl.Item("I1").Value = MyCtrl.item("I2").Value * 3

演算結果を別のコントローラのアイテムに設定する場合は，リンク機能と組み合わます．具体的にはリンク設定しているアイテムに値を設定します．ただし，リンク先に伝播されるのは次のスキャンのタイミングなので，1 スキャン分の遅延があることに注意してください．

スクリプトの実行はサンプリングの最後にサブルーチン Main を毎回呼び出すことで実行されます．もし，Main 内で無限ループを記述した場合は，サンプリング処理自体も停止状態になりますので注意してください．ただし，他のコントローラのスレッドは停止しないので，並行して処理されています．スクリプトの開始や停止を API 経由で制御したい場合は，Execute メソッドの CSQ コマンド($CSQ_SCRIPT_ENABLED$)を使います．MyCtrl オブジェクトを使ってスクリプト内から実行することもできます．

例） 自コントローラのスクリプトの実行を停止する

MyCtrl.Execute “$CSQ_SCRIPT_ENABLED$”, False

スクリプト内でエラーが発生した場合は，スクリプトの実行が停止されます．再開するには，API 経由でCSQコマンド（$CSQ_SCRIPT_ENABLED$）を実行します．スクリプトを停止させたくない場合は，スクリプトの先頭で”On Error Resume Next”ステートメントを実行し，スクリプト内のエラーを無視することもできます．

エラー情報の詳細はCSQコマンド（CSQ_SCRIPT_ERROR）で取得することができます．ただし，文法エラー等の起動時（初期化時）エラーはこのコマンドで取得することはできません．起動時エラーはログに出力されています．

【注意事項】

このスクリプト実行機能によって，BCD 変換，二項演算等，その他の機能をスクリプトでも実現できますが，スクリプト実行はオーバーヘッドが大きいので，スクリプトを使わないで実現できる場合はそれを使うことを推奨します．

3.25. CSQコマンド

CaoSQLEngine クラス, および CaoSQLController の Execute メソッドで下記のコマンドを使うことができます.

表 3-12 CaoSQLEngine クラスの CSQ コマンド

コマンド名	引数	意味
(未実装)		

表 3-13 CaoSQLController クラスの CSQ コマンド

コマンド名	引数	意味
$CSQ_RESET_SCANTIME$	なし	3 つのスキャンタイム測定値(現在値, 最小値, 最大値)をリセットします. 現在値と最大値は 0 に, 最小値は LONG_MAX に初期化されます.
$CSQ_SCAN_ONETIME$	なし	コントローラやアイテムがアクティブでない状態であっても強制的にスキャン(値取得)します. (注)このコマンドは同期スキャンするので, 全アイテムのスキャンが完了するまで制御は戻りません. したがって, 自動サンプリング中にこのコマンドを実行すると処理がブロックされ戻りが遅くなることがあります.
CSQ_SCRIPT_ERROR	なし	スクリプト実行時のエラー情報を下記の3要素の配列で取得します. (<Code>, <Line No>, <Description>) <Code> VT_I4 : エラーコード <Line No> VT_I4 : 行番号 <Description> VT_BSTR : エラー詳細
$CSQ_SCRIPT_ENABLED$	VT_BOOL	サンプリング時のスクリプトの実行を許可(True)・拒否(False)します. 初期値が許可に設定されていた場合は, スクリプトは CaoSQL 起動時にロードされます. 初期値が拒否に設定されていたときは, スクリプトは許可に変更された最初のサンプリング時にロードされます.
その他		対応するプロバイダの CaoController クラスの Execute メソッドにリダイレクトされます.

3.26. 使用可能型情報一覧

以下に CaoSQL で使用可能な型の一覧を示します.

表 3-14 使用可能型情報

機能	使用可能型情報
イベント通知 (アイテム値変化) (ステータス変化)	VT_EMPTY , VT_NULL , VT_BOOL , VT_I1 , VT_UI1 , VT_I2 , VT_UI2 , VT_I4 , VT_UI4 , VT_R4 , VT_R8 , VT_CY , VT_DATE , VT_BSTR, VT_ERROR, VT_DISPATCH, VT_UNKNOWN
配列要素抽出機能	VT_BOOL , VT_I1 , VT_UI1 , VT_I2 , VT_UI2 , VT_I4 , VT_UI4 , VT_R4, VT_R8, VT_CY, VT_DATE, VT_BSTR, VT_VARIANT
マスキング	VT_I1, VT_UI1, VT_I2, VT_UI2, VT_I4, VT_UI4
BCD変換	VT_I1, VT_UI1, VT_I2, VT_UI2, VT_I4, VT_UI4
データ型変換機能	VT_BOOL , VT_I1 , VT_UI1 , VT_I2 , VT_UI2 , VT_I4 , VT_UI4 , VT_R4, VT_R8, VT_CY, VT_DATE, VT_BSTR, VT_VARIANT
不感帯設定	VT_I1, VT_UI1, VT_I2, VT_UI2, VT_I4, VT_UI4, VT_R4, VT_R8
感帯設定	VT_I1, VT_UI1, VT_I2, VT_UI2, VT_I4, VT_UI4, VT_R4, VT_R8

4. CaoSQLConfig

4.1. 概要

CaoSQL Configuration Manager(実行ファイル名 CaoSQLConfig.exe，以後 CaoSQLConfig)は，マシン内の CaoSQL のアイテム設定を行うツールです．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，CaoSQL の起動時に読み込まれます．

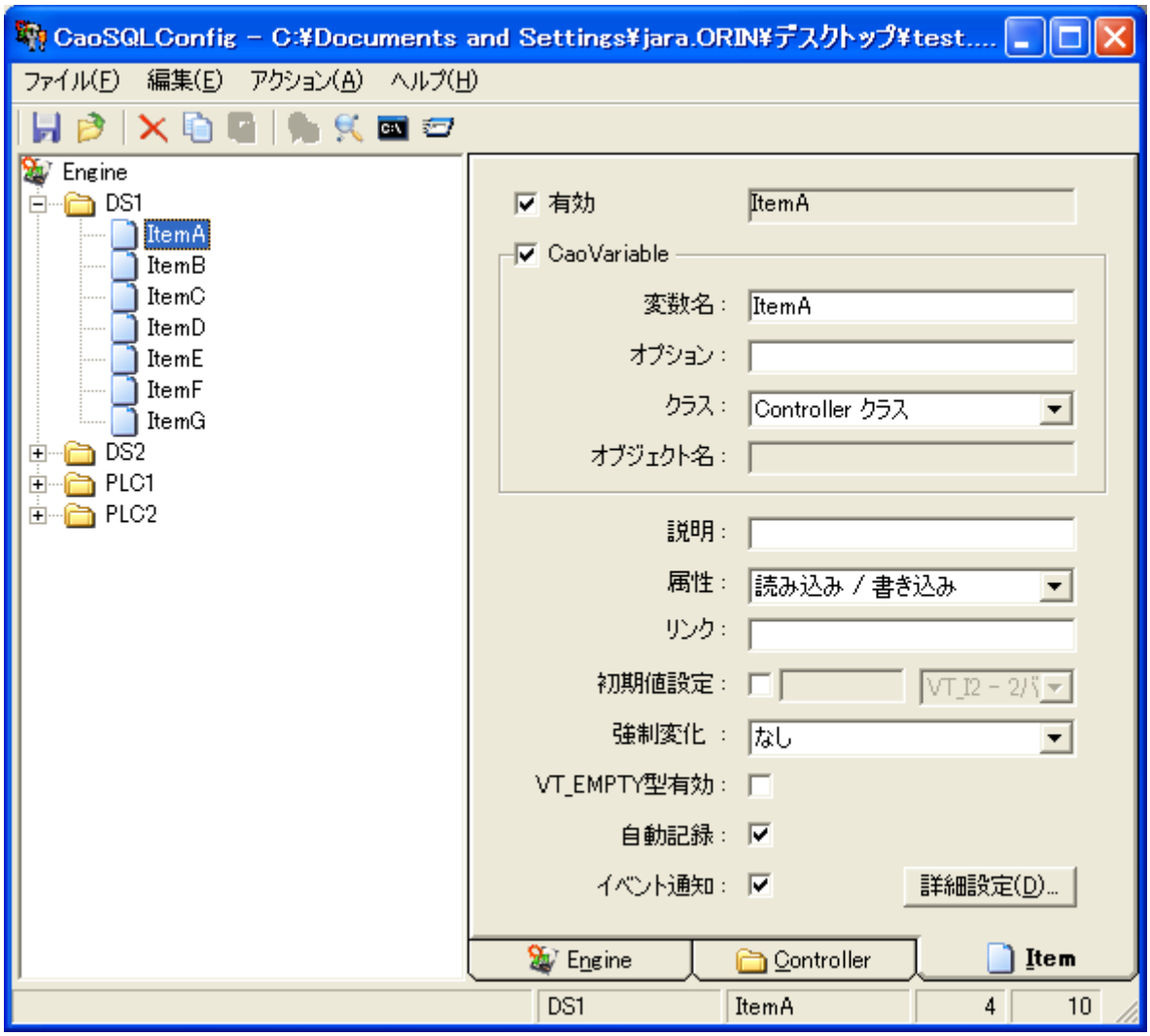


図 4-1 CaoSQLConfig 画面

4.2. 操作方法

4.2.1. タブ入力

画面右のタブは，ツリービューのノードを選択することによって，選択されたノードに対応したオブジェクトの情報を表示します．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 CaoSQL を起動した時にエラーが発生し，CaoSQL のリストから外れてしまう可能性がありますので CaoSQLConfig でデータを入力する処理で充分に確認をすることが必要です．

4.2.1.1. エンジンタブ

エンジンタブでは図 4-2 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．（CaoSQLエンジン名は自動的にデフォルトの名前が適応されています）

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．

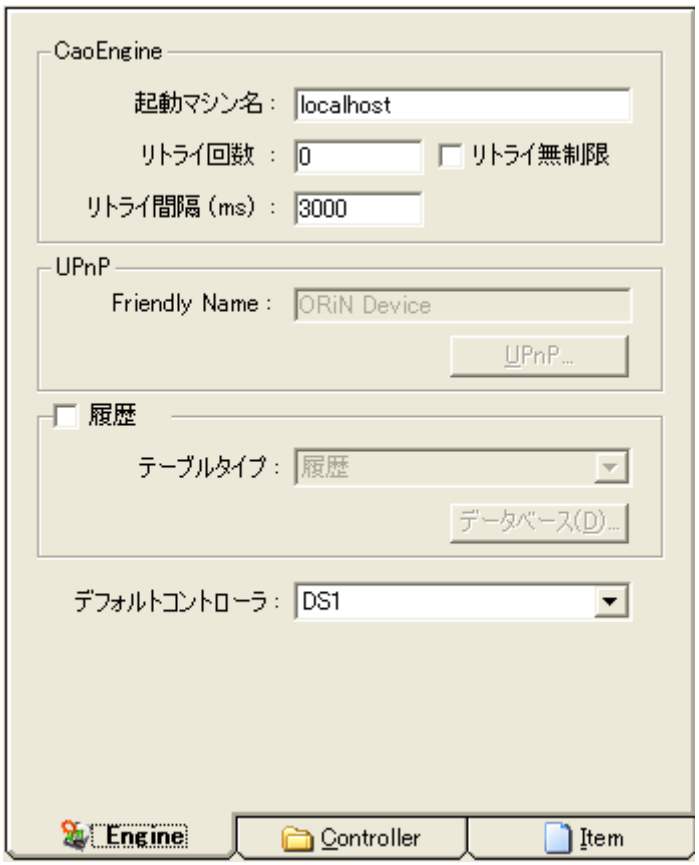


図 4-2 エンジンタブ

ここで設定可能な項目は以下の通りです.

[起動マシン名] – Location

CAO.exe の起動コンピュータ名，又は IP アドレスを入力します. 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] – RetryCount, Unlimited

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] – Retry Interval(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.

[ヒストリ機能] – History

このチェックボックスが入っていると，ヒストリ機能が有効になります.

[テーブルタイプ] – TableType

テーブルにレコードを書き込む方法を設定します.

履歴データと最新データの判別は，レコードの flags という項目で判別することができます.

- History
  履歴データをデータベースに書き込みます. アイテムの値が変化すると，レコードをデータベースに追加します.

- Current Value
  最新データをデータベースに書き込みます. アイテムの値が変化すると，最新データのレコードを上書きします.

- History & Current Value
  上記 2 つを同時に行います.

[デフォルトコントローラ] – Default Controller

コントローラを指定する処理でコントローラ名を指定しなかった場合，このコントローラ名が使用されます.

[DataBase の設定] – DataBase

ヒストリ機能(データロギング)の設定画面が立ち上がります.

4.2.1.1.1. ヒストリ機能設定画面

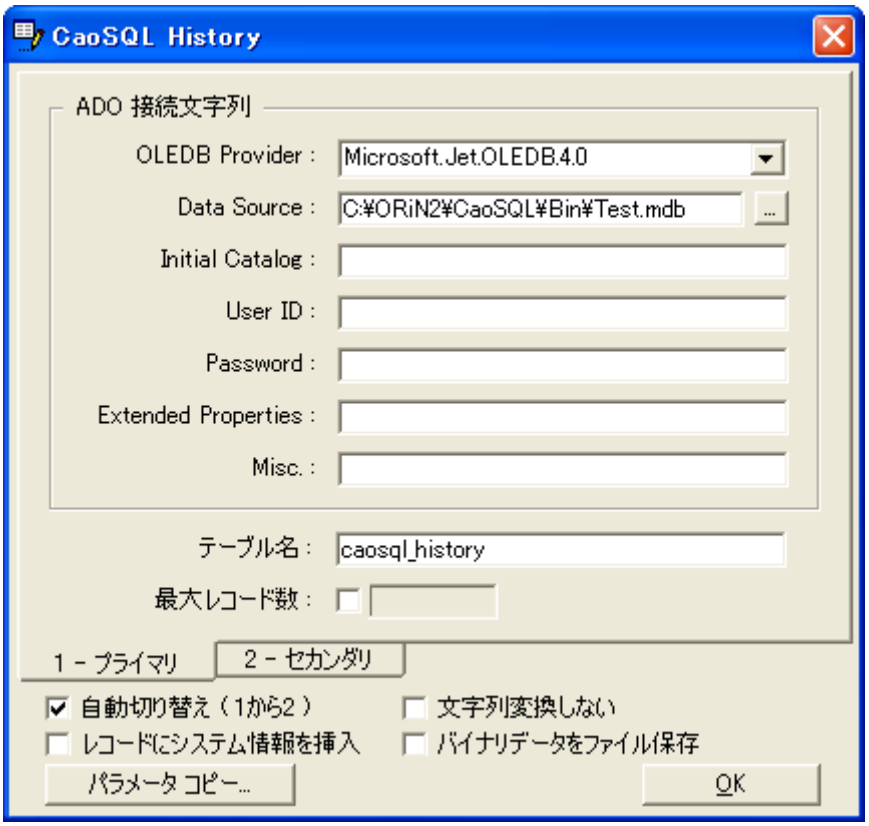


図 4-3 ヒストリ設定ダイアログ

[ADO 接続文字列] – ADO Connection String

ADO コネクションを開くときに使用されるオプションを指定します。

その他のADO接続文字列のオプションを指定したい場合は[**Misc.**]で

<設定項目名>=<設定内容>[;<設定項目>=<設定内容>][;<設定項目>=<設定内容>]…

の形式で入力します. 詳細は ADO リファレンスを参照して下さい.

[テーブル名] – Table Name

ヒストリ機能に使用するテーブルの名前を入力します.

[最大レコード数] – Table Name

ログの最大レコード数制限を設定します. チェックボックスを入れ, 制限数を入力します. チェックボックスを外す, 又は 0 を入力すると最大レコード数をチェックしません. このチェックはとてもオーバーヘッドが高いので, 特に必要がない限りオフにして下さい.

[自動切り替え] – Auto Switch

セカンダリデータベースへの切り替え機能の有効・無効を設定します.

セカンダリデータベースの設定は、ヒストリ機能設定画面のタブで切り替えることができます.

[文字列変換をしない] – Auto Switch

レコード記録時にアイテム値を文字列に変換するかどうかを設定します。

[バイナリデータをファイルに保存] – BLOB To File

アイテム値がバイナリデータであった場合にファイルに保存するかどうかを設定します。

[パラメータ コピー] – Copy Parameters

パラメータをコピーするプルダウンメニューが表示されます. (図 4-4)

ADO 接続文字列, テーブル名, 最大レコード数の設定をプライマリからセカンダリ, またはセカンダリからプライマリへパラメータをコピーすることができます.

また, ADO 接続文字列を DataBase プロバイダの AddController 引数のオプション文字列にしてクリップボードへコピーすることもできます.

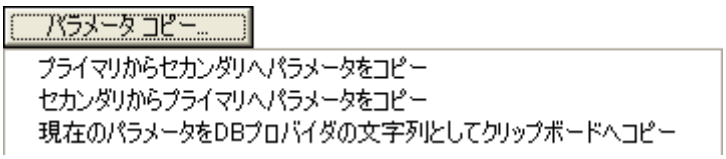


図 4-4 パラメータコピー メニュー

4.2.1.2. コントローラタブ

コントローラタブでは図 4-5 に示した形式で情報を設定・表示します.

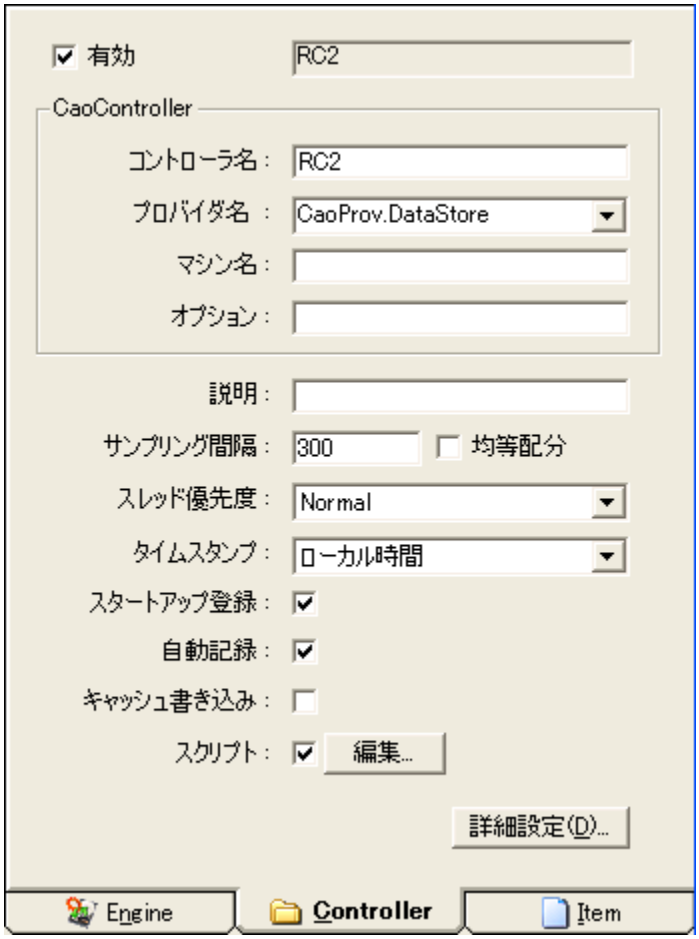


図 4-5 コントローラタブ

コントローラの情報は自由に設定できますが,「CaoSQL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, メニューバーの[編集]→[名前の変更]でのみ変更できます. 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.

[コントローラの有効/無効設定] – Enabled

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 CaoSQL 実行時に起動されません.

[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]– Name，Provider Name，Machine，Option

CaoControllerを作成する際に渡すパラメータです．入力した値がそのままAddControllerの引数に使用されます．

[サンプリング間隔] – Sampling Rate

アイテムをポーリングする周期をms単位で指定します．1周期が指定した値未満の時に指定した周期になるまでスレッドが待機します[28]．

[均等配分] – Distribute equally

均等配分のオプションにチェックがついていると，CaoSQL はサンプリングを 1 アイテムの値を取得する毎に(サンプリング間隔／アイテム数)時間だけ遅延させます．

[スレッド優先度] – Thread Priority

コントローラ毎のサンプリングスレッドの優先度を設定します．優先度の高い順に優先的にサンプリングされます[29]．通常はNORMALにして下さい．優先度に対する調整は以下の通りです．

TIME CRITICAL > HIGHEST > ABOVE NORMAL > **NORMAL** > BELOW NORMAL > LOWEST > IDLE

ここで TIME CRITICAL はプロセス優先度に関わらず最優先になり，IDLE は最低の優先度に設定されます．プロセス優先度は，[**アクション**]→[**オプション**]で変更できます．

[詳細情報] – Description

ここは任意の文字列を格納できるので，そのコントローラの説明などを記述しておくと分かりやすいでしょう．無効な場合は CAO プロバイダのコントローラオブジェクトの Help プロパティにリダイレクトされます[30]．

[タイムスタンプ] – Time Stamp

アイテムのタイムスタンプの更新方法を設定をします．

- **n/a** :タイムスタンプを更新しません．
- **Device Time** :CAOプロバイダの変数オブジェクトの日時で更新します．[31]
- **Local Time** :現在のローカル日時で更新します．
- **System Time** :世界協定時刻(UTC)で表された現在のシステム日時で更新します．

[28] (参考)経験的には，クライアントが要求する周期の 1/5 から 1/2 くらいの間に設定するといいでしょう．例えば，クライアントが 100ms の周期を期待するなら，20ms から 50ms くらいの間で設定するとよいでしょう．
[29] コントローラ毎の OnChangeItem，OnChangeState イベント処理スレッドの優先度もここで設定した値に設定されます．
[30] Ver. 1.8.0 以降の場合のみ．
[31] Device Time の機能は，現在未実装となっています．

[スタートアップ登録] – Startup

ここのチェックを入れると，CaoSQL起動後すぐにサンプリングが開始されます．逆に外すと，CaoSQL起動時にスレッドが停止状態(ステータスがディアクティブ)になりキャッシュ値は更新されません．書き込みはサンプリングが停止中でも可能です[32].

但し，スレッドが停止状態で値の読込みを実施すると，その時点でキャッシュ値を更新してから値を返します．

[History の自動記録] – Auto History

ここのチェックを外すと，そのコントローラ以下のアイテムでは個別の History の設定は保持されますが，History の記録はされません．Item タブの History の設定は変更不可となります．Snapshot 機能で明示的に History を記録する場合などは，このチェックを外してまとめて OFF にすると便利です．

[キャッシュ書込み機能] – Write to Cache

キャッシュ書込み機能のオン/オフを設定します．オンの場合，サンプリングを待たずにアイテムの値をキャッシュに書き込みます．アイテムの put_Value を行った後，すぐ get_Value を行う場合などに有効です．

[スクリプト] – Script

スクリプト実行機能のオン/オフを設定します．オンの場合，サンプリングの最後にスクリプトが実行されます．この機能を使用すると複雑な条件などを設定することも可能です．

[編集...]

スクリプト実行機能で実行するスクリプトを編集することができます．

[Details の設定] – Details

ウォッチドッグタイマ，トリガ，Pre/PostProcessing の設定ダイアログが立ち上がります．ダイアログのキャプションには設定しているコントローラ名が表示されます．

[32] (ヒント)接続先の CAO プロバイダの変数がキュー(先入れ先出し)やスタック(先入れ後出し)形式のバッファ型変数の場合に，複数の CaoSQL をカスケード接続してデータを共有するときは，このオプションを使ってデータを読込むアプリケーション，データを書込むアプリケーションを明示的に指定できます．(書込むアプリケーションは OFF にして下さい)

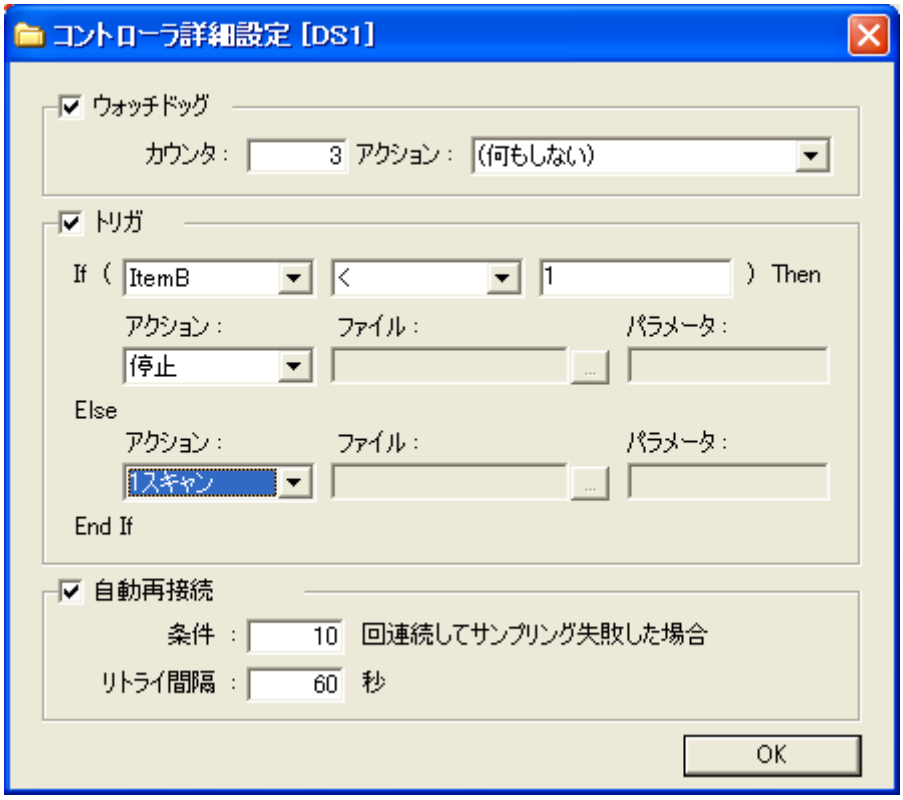


図 4-6 Controller Details の設定

[ウォッチドッグタイマ] – WatchDog

ウォッチドッグタイマの設定を行います。

有効にした場合、1 サンプリング中にサンプリング間隔時間を指定回数(**Counter**で設定した値[33])以上超えると，コントローラのステータスをエラーにし，**Action**で選択した処理を行います．**Action**でStopを選択した場合，サンプリングスレッドを停止します．

【注意事項】

サンプリングの経過時間の評価はアイテム値の更新毎に行われ、その時点でサンプリング間隔を経過していた場合に、時間超過としてカウントを 1 増加します。

このため、アイテム数が指定回数より少ない場合は、ウォッチドッグタイマは条件を満たしません。

また、アイテム値の更新がサンプリング間隔と比べて十分に遅い場合、ウォッチドッグの評価時間が遅延します。

[トリガ] – Trigger

Trigger のチェックを ON/OFF することでトリガ機能を使用するかどうかを設定できます．

- **STEP 1:**

  条件式にトリガ対象となるアイテムを選択します．ここではトリガを設定するコントローラのアイテ

[33] **Counter** 設定値の有効範囲は 1～5 です．

ムのみ選択できます.

・ **STEP 2**:
トリガを実行する条件式と条件を設定します. ここで設定された条件は文字列として保存されますが, トリガ実行時にはトリガ対象のアイテムの型に合わせて変換されます.

・ **STEP 3**:
最後にトリガ時に実行されるアクションを設定します. “Else”では逆の条件の場合を設定することが可能です.

[自動再接続] – Auto Reconnection

エラーが **Condition** に設定された回数発生した場合に, **Retry Interval** に設定した秒数遅延してから再接続を行います.

4.2.1.3. アイテムタブ

アイテムタブでは図 4-7 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.

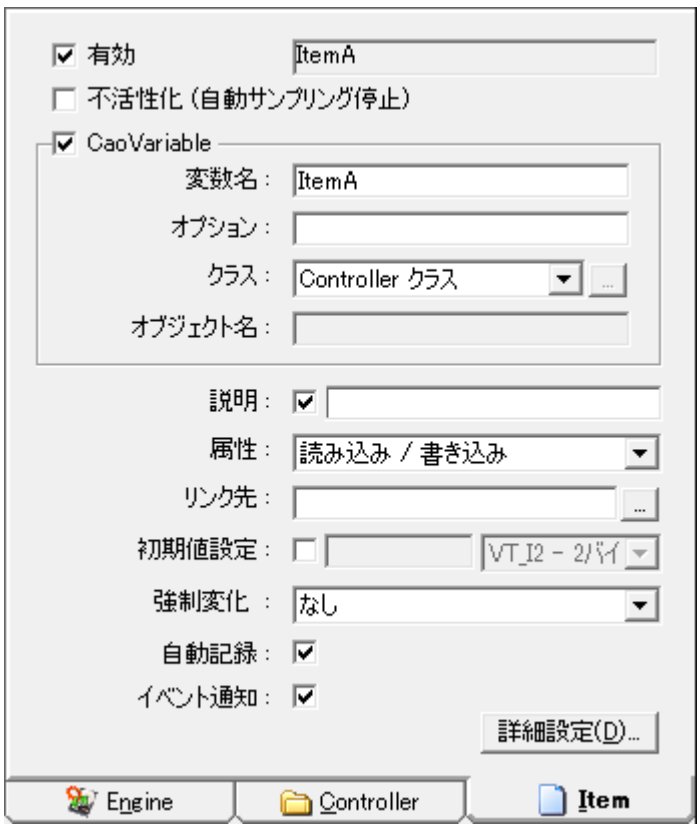


図 4-7 アイテムタブ

ここで設定できる項目は以下の通りです.

[有効] – Enabled

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.

[不活性化(自動サンプリング停止)] - Inactive

コントローラのサンプリング周期時にアイテムの読込み, または書込みを行わないようにする設定です.

[CaoVariable 変数の有効] – CaoVariable

このアイテムをCAOの変数クラスとバインドするか否かの設定です. 無効に設定された場合はバインドされないで, CaoSQL 内だけのローカル変数となります.

[変数名] – Variable Name

CaoVariable オブジェクトの作成(ICaoController::AddVariable)時に渡される引数です. 入力された値はそのまま使われます. 大文字・小文字は区別されます.

[オプション] – Option

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.

[クラス] – Class

変数クラスの種類を選びます.

[システムオブジェクト名] – Object Name

変数の親オブジェクト名を入力します. 但しコントローラの変数オブジェクトは, 親コントローラのみ可能なので入力できません.

[システムオブジェクト オプション] – Object Option

親オブジェクト生成時におけるオプション文字列を指定します. [クラス]横の[…]ボタンを押下し, 表示されたウィンドウから設定します. 但し, コントローラクラスの変数オブジェクトは, 親コントローラのみ可能なので[…]ボタンは無効となります.

[詳細情報] – Description

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 CSQ 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. 無効な場合はCAOプロバイダの変数オブジェクトのHelpプロパティにリダイレクトされます[34].

[34] Ver. 1.8.0 以降の場合のみ.

[値の初期化設定] – Initialize

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.

[リンク(値伝播)] – Link Item

リンク機能によりリンクするアイテムを設定します. リンクアイテム名は“コントローラ名¥アイテム名”の表記で指定します. アイテム名のみの場合, 同じコントローラのアイテムとして認識します. (デフォルトコントローラ名でないことに注意して下さい) リンク先のアイテムの整合性はチェックしていません.

[History の自動記録] – Auto History

読込みアイテムの時このチェックを入れると, 値変化をヒストリデータベースにログすることができます. 但し History 機能がオフ, または親コントローラで History を不使用に設定されている場合, この設定は使用できません.

[強制値変化] – Force to change

CaoSQL のデフォルト動作では, 値の変化で OnChangeItem イベントやヒストリへの記録などが行われます. このオプションを”1 - Reading”にすることで, 一つ前の値との比較がされず, 常に値が変化したように処理されます. つまりサンプリング毎に値が変化したと見なされます.

例えば, 1→1→2→2 とCaoSQLがサンプリングしたとき, デフォルト動作では初回の 1 と, 1→2 の計 2 回しかOnChangeItemイベント, OnChangeStateイベントやヒストリへの記録がされませんが, このオプションを有効にしておくと, 4 回とも全てイベントが発生し, ヒストリにも記録されます[35]. (もちろん, ヒストリ機能が有効になっている場合に限ります.)

また, ”2 - Writing” は外部アプリケーションから値の更新があった場合に強制的に値の変化と見なします. この設定は[**Write to Cache**]が有効になっているか, もしくは[**CaoVariable**]の設定が無効になっている場合に有効です.

[イベント有効/無効] – OnChange Event

CaoSQL アイテムイベント機能の有効/無効を設定します.

[35] (注)ヒストリ機能が有効で, かつサンプリング周期が短いと大量のデータが記録されるので注意して下さい.

[Details の設定] – Details

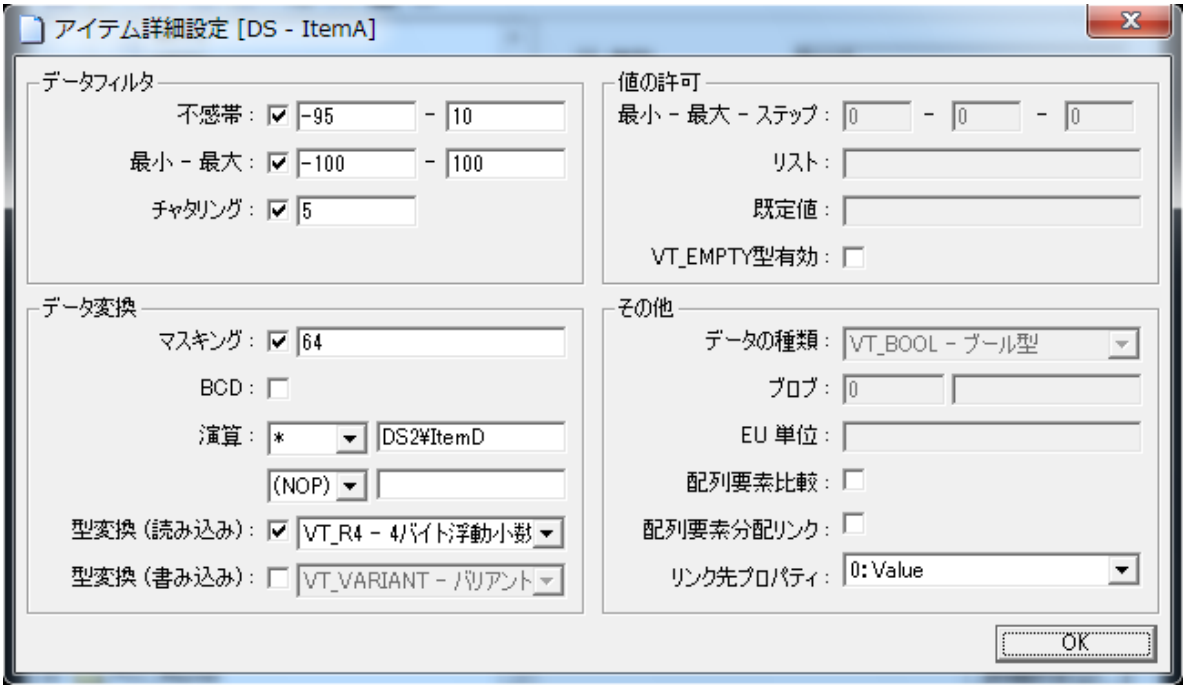


図 4-8 アイテム Details ダイアログ設定

[不感帯設定] – Dead-Band

不感帯の設定．最小値・最大値で指定された範囲内の変化は値の変化として認識されず，OnChangeItemイベント通知処理やヒストリ処理などが行われません．

[感帯設定] – Min-Max

感帯の設定．最小値・最大値で指定された範囲外の変化は値の変化として認識されず，OnChangeItemイベント通知処理やヒストリ処理などが行われません．

[チャタリング]－Chattering

指定した回数分，同じ値であった場合に真値とします．

[マスキング] – Masking

アイテムの値にマスクをかけます．マスクデータは 10 進表記です．

[BCD 変換] – BCD

アイテムの値をBCD変換します．Read／Write共に変換されます．

[二項演算] – Calculation

別のアイテムと二項演算します. Read時にのみ変換されます.

[値受取型設定] – Request Type R/W

サンプリングしたデータまたは書き込まれたデータを指定した型に変換します.

[VT_EMPTY 型有効] – Enable VT_EMPTY

有効なデータ型として VT_EMPTY を使うかどうかの設定です. ここのチェックを外すと VT_EMPTY は有効なデータとして認識されませんので, サンプリングされたデータがこの型だった場合は値の変化として認識されず, イベント通知処理やヒストリ処理などが行われません.

[配列内データ比較] – Compare Arrays

データが配列の場合, 配列の各要素を比較します. 値の変化があったとき OnChangeItem イベント通知処理やヒストリ処理を行います.

[配列要素分配リンク] – Extract Linking

配列要素分配リンク機能(Extract Linking オプション)の有効/無効を設定します.

4.2.2. メニュー

CaoSQLConfig 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.

4.2.2.1. 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.

Open,**Save**,**Save As..**で使用されるファイルの拡張子は“**csq**”です.

Import,**Export** で使用されるファイルの拡張子は“**csx**”です.

[新規作成] – New

新規に csq ファイルを作成します.

[ファイルから読込み] – Open…

指定したcsqファイルから設定情報を読込みます[36].

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] – Save

現在の設定内容を csq ファイルに上書き保存します.

[36] 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.

[名前を付けて保存] – Save As…

現在の設定内容を csq ファイルに保存します.

[インポート] – Import...

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] – Export...

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] – Import from CSV file...

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] – Export to CSV file...

選択しているノードを csv ファイルに出力します.

[終了] – Exit

CaoSQLConfig のプログラムを終了します.

4.2.2.2. 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] – Add Controller

CaoSQL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] – Add Item

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). また,通常アイテムはエイリアスコントローラへの追加もできません. 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.

[配列型アイテム追加] – Add Array Item

ツリービューで選択されたノードに配列型アイテムを追加します. アイテム追加同様, 既に登録されているアイテム名は名前として登録できません. また,配列型アイテムはエイリアスコントローラへの追加もできません.

[エイリアスコントローラ追加] – Add Alias Controller

エイリアスコントローラを追加します. 既存の通常コントローラ, エイリアス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.

[エイリアスアイテム追加] – Add Alias Item

ツリービューで選択されたコントローラにエイリアスアイテムを追加します. アイテム追加同様, 既に登録されているアイテム名は名前として登録できません.

[グループの追加] – Add Group

グループを追加します. グループを追加することで, アイテムをカテゴリに分けることができます. グループはアイテム群のラベルとして使用します.

[名前変更] – Rename

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] – Delete

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] – Copy

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] – Paste

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] – Sort

選択されているレベルのノードを昇順ソートします.

[DDE 文字列のコピー] – Copy DDE String

ツリービューで選択されたアイテムの DDE 文字列をクリップボードにコピーします.

4.2.2.3. アクションメニュー

CaoSQL.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] – Settings

CaoSQL の環境設定ダイアログが表示されます.

[全般タブ] – General

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

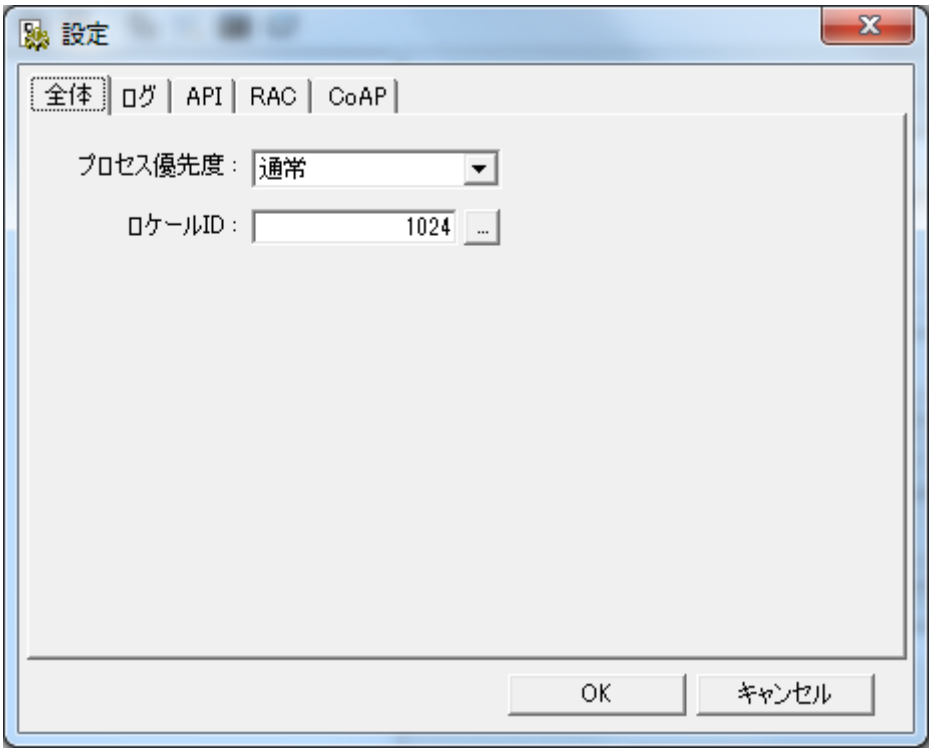


図 4-9 CaoSQL 環境設定（General）

[プロセス優先度] – Process Priority

CaoSQL 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.

REAL TIME > HIGH > **NORMAL** > IDEL

[ロケール ID] – LocaleID

使用する言語 ID を設定します.

[ログ設定タブ] – Log

CaoSQL のログ出力に関する設定を行うことができます.

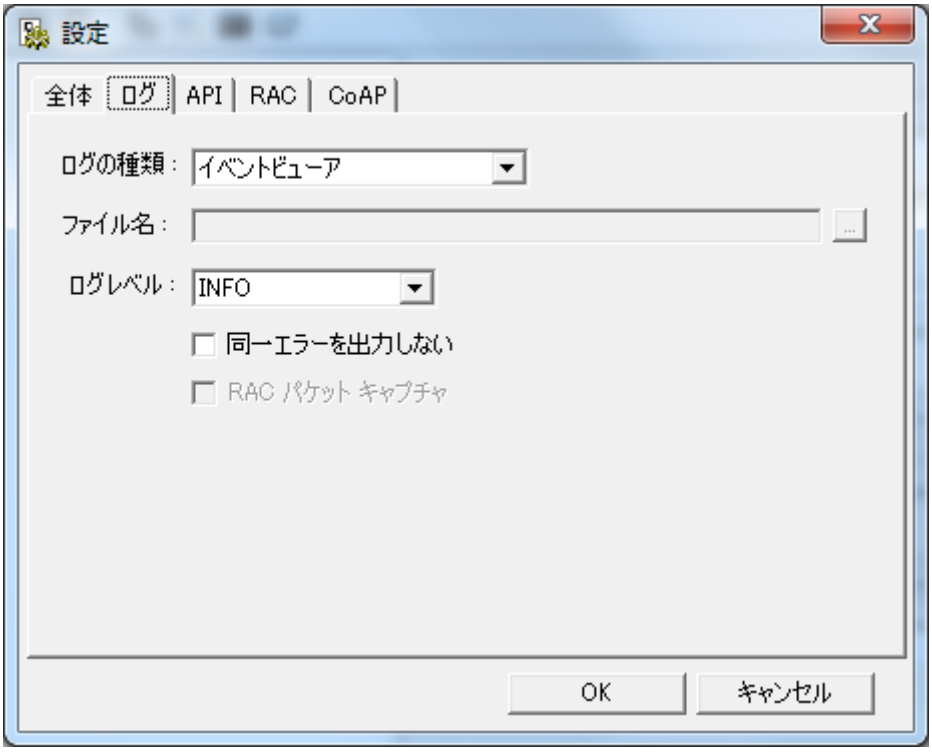


図 4-10 CaoSQL 環境設定(Log)

[ログタイプ] – Log Type

CaoSQL.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 4-1 ログタイプ

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

[ファイル名] – File Name

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] – Log Level

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] – NOT log the same error repeatedly.

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

DDE や JNI などの CaoSQL で利用する API に関する設定を行うことができます.

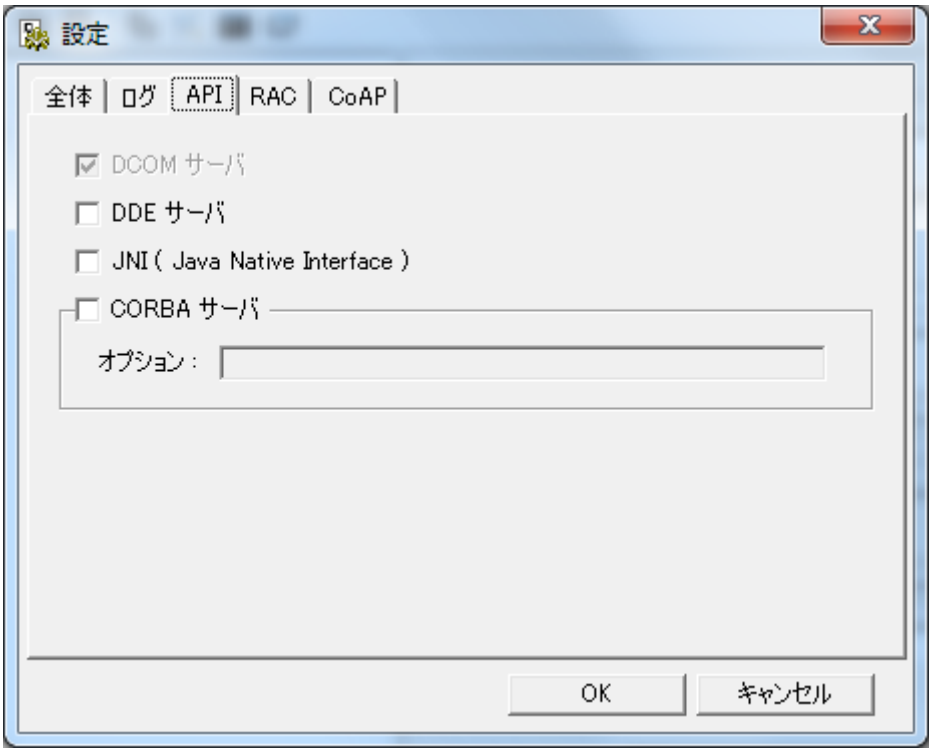


図 4-11 CaoSQL 環境設定 (API)

[DCOM サーバ機能] – DCOM Server

DCOM サーバ機能の有効/無効設定です. (常に有効)

[DDE サーバ機能] – DDE Server

DDE サーバ機能を有効/無効設定です.

[JNI機能] – JNI(Java Native Interface)[37]

JNI 経由のアクセスを有効に/無効設定です.

[CORBAサーバ機能] – CORBA Server[38]

CORBA サーバ機能を有効/無効設定です. Option に入力された文字列を引数として使用します.

[RAC 機能設定タブ] – RAC

ここでは, RACを利用してCaoSQLとデータをやり取りするための, RACサーバの設定を行います. 詳しい設定内容については,「RAC プロバイダガイド」の 2.2.1 AddControllerを参照して下さい.

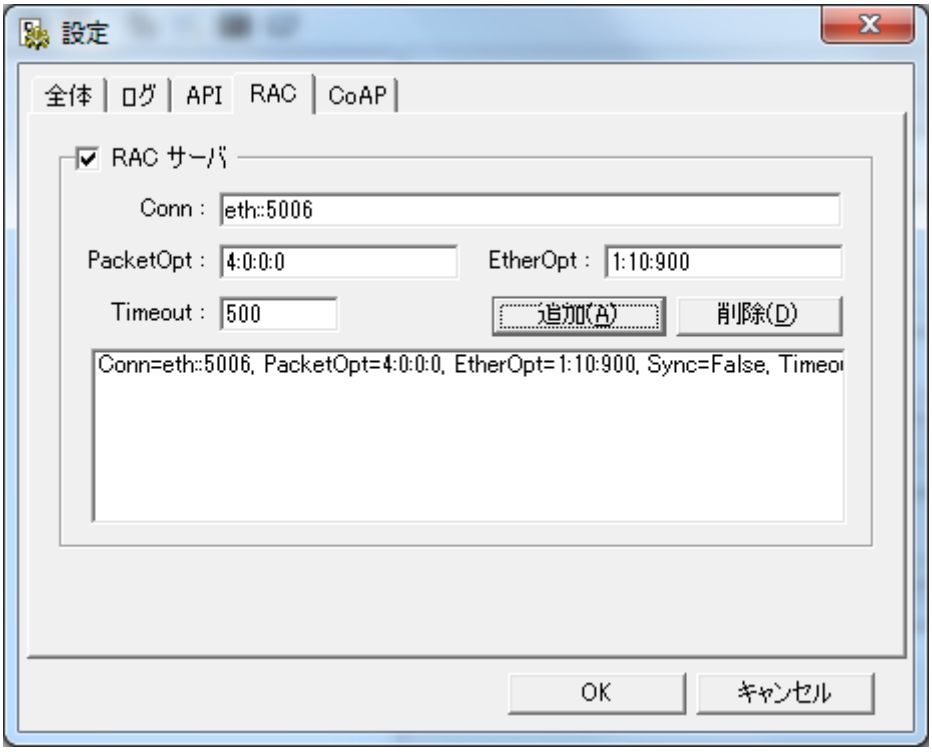


図 4-12 CaoSQL 環境設定(RAC)

[RAC サーバ機能の有効/無効] – RAC Server

RAC サーバ機能を有効にするか無効にするかの設定を行います.

[37] 現在, JNI 機能はこの設定に関係なく使用できます.
[38] 現在, この機能は未実装です.

[RAC サーバの接続パラメータ] – Conn, PacketOpt, EtherOpt, Timeout

RAC プロバイダの AddController メソッドへの引数を設定します.

[RAC サーバの追加] – Add

設定したパラメータで RAC プロバイダの AddController メソッドへのオプション引数を作成します.

[RAC サーバの削除] – Delete

現在選択されている, [RAC サーバの追加]で追加した文字列を削除します.

[CoAP 機能設定タブ] – CoAP

ここでは, CoAPを利用してCaoSQLとデータをやり取りするための, CoAPサーバの設定を行います. 詳しい設定内容については,「CoAP プロバイダガイド」の 2.2.1 AddControllerを参照して下さい.

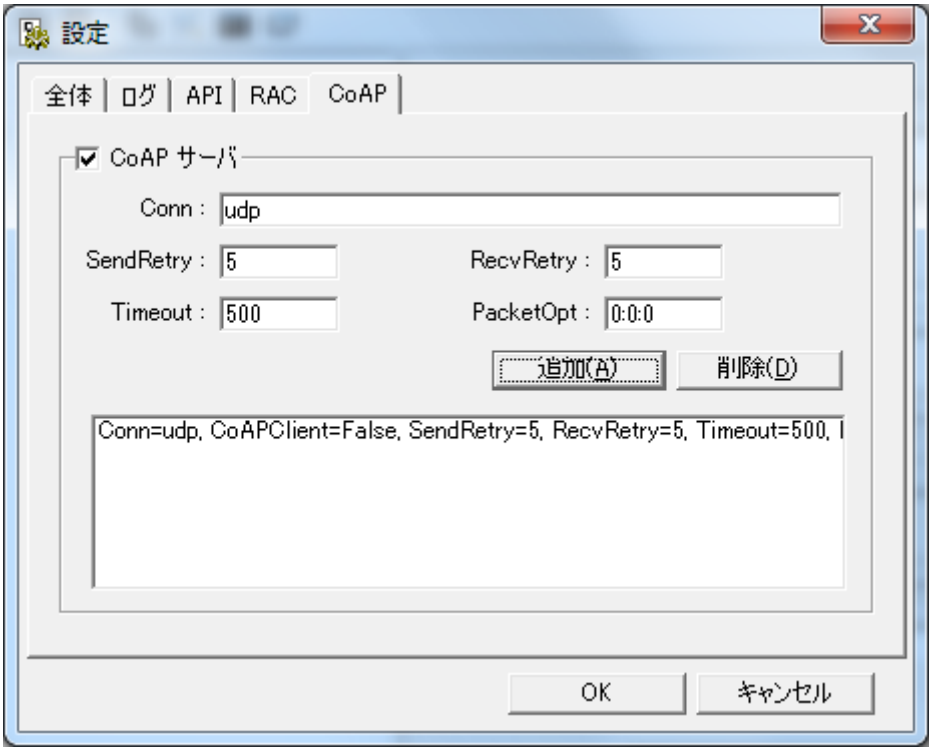


図 4-13 CaoSQL 環境設定(CoAP)

[CoAP サーバ機能の有効/無効] – CoAP Server

CoAP サーバ機能を有効にするか無効にするかの設定を行います.

[CoAP サーバの接続パラメータ] – Conn, SendRetry, RecvRetry, Timeout, PacketOpt

CoAP プロバイダの AddController メソッドへの引数を設定します.

[CoAP サーバの追加] – Add

設定したパラメータで CoAP プロバイダの AddController メソッドへのオプション引数を作成します.

[CoAP サーバの削除] – Delete

現在選択されている, [CoAP サーバの追加]で追加した文字列を削除します.

[サービスの起動] – Service Start

CaoSQL サービスを起動します. csq ファイル情報は CaoSQL の起動時に読込まれる為, csq ファイル情報を更新したい場合は CaoSQL サービスを再起動する必要があります.

CaoSQL.exe がサービスとして登録されていない場合や, 既にサービスが実行されている場合は使用できません.

[サービスの停止] – Service Stop

CaoSQL サービスを停止することができます. CaoSQL.exe がサービスとして登録されていない場合や, 既にサービスが停止されている場合は使用できません.

[サービスの再起動] – Service Restart

CaoSQL サービスを再起動することができます. CaoSQL.exe がサービスとして登録されていない場合は使用できません.

[CaoSQLTester の実行]

CaoSQLTester ツールを起動します. 内部的には Shell 関数をそのまま呼び出しています.

[CaoSQLCmd の実行]

シェルコマンドを打ち込むダイアログが表示されます. デフォルトでCaoSQLCmd.exeのパスが表示されるので, CaoSQLCmdのコマンドを追加して実行することができます. (図 4-14) 内部的にはShell関数をそのまま呼び出しています.

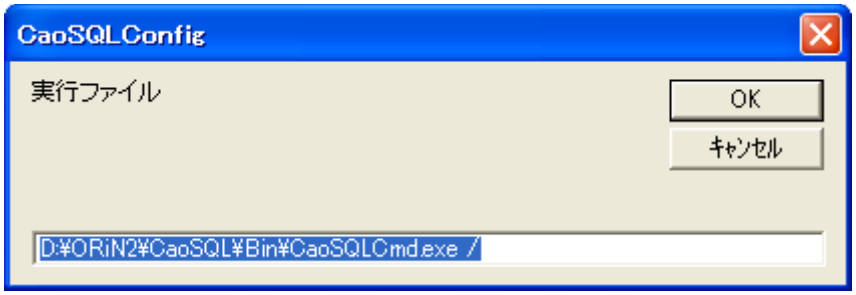


図 4-14 シェルコマンドの実行ダイアログ

[CaoSQLLauncher の実行]

CaoSQLLauncher ツールを起動します. 内部的には Shell 関数をそのまま呼び出しています.

4.2.2.4. 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.

4.3. デンソーロボットモード

CaoSQLConfig を「DENSO NetwoRC プロバイダ」または「DENSO RC8 プロバイダ」に特化したモードで起動することが可能です。

起動方法は、CaoSQLConfig の起動ファイルにオプション ”denso” を渡します。

<ORiN2 インストールディレクトリ>¥CaoSQL¥Bin¥CaoSQLConfig denso

4.3.1. コントローラの追加（デンソーロボットモード）

コントローラの追加は，通常モードと同様に行います.

デンソーロボットモードでは，コントローラの追加時にウィザード形式のダイアログが表示されます.

追加するプロバイダを選択して、コントローラ名を入力します。入力後、 [次へ]ボタンを押下して下さい.

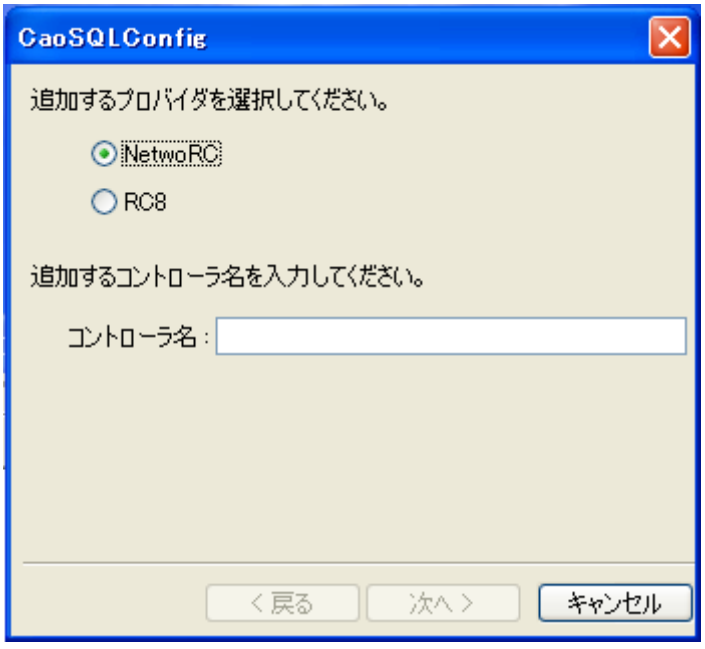


図 4-15 デンソーロボットモード時のコントローラの追加

次に，接続先のコントローラのパラメータを設定します. EtherNet なら IP を RS232C ならポート番号とボーレ

ートを入力します. (図 4-16)

RC8 プロバイダを選択した場合は, IP アドレスのみ入力します. (図 4-17)

図 4-16 デンソーロボットモード時の NetwoRC を選択した場合のコントローラの追加

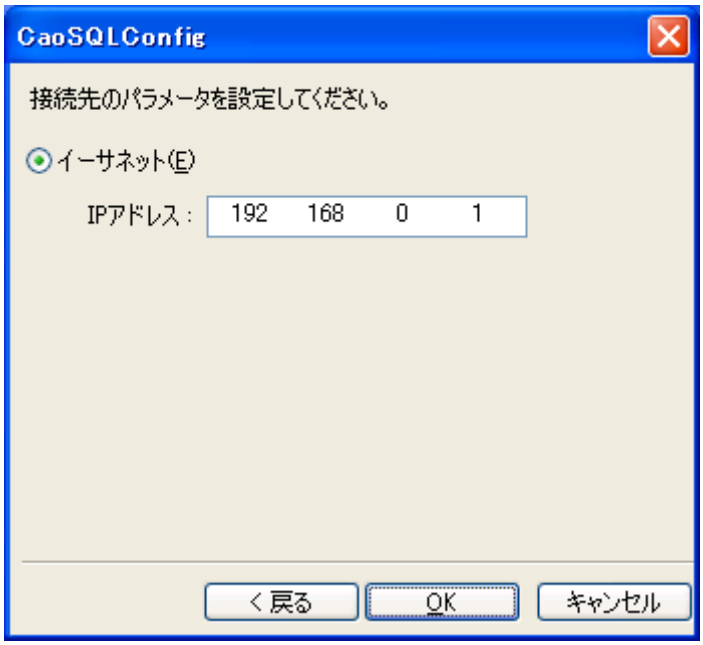


図 4-17 デンソーロボットモード時の RC8 プロバイダを選択した場合のコントローラの追加

[OK]ボタンを押すと, コントローラが追加されます. 同じ名前のコントローラが既に存在してあり, 使用できない文字が含まれている場合は, 追加失敗します. (4.2.1.2)

追加が正常に終わると，次のような画面になります． 接続先などを変更したい場合は，[編集...]ボタンを押すと，追加時に表示されたダイアログの 2 ページ目が表示されます．(図 4-18，図 4-19)

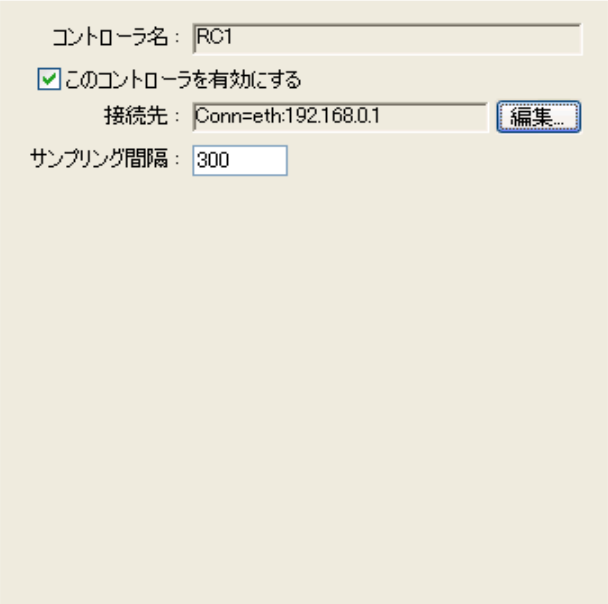


図 4-18 デンソーロボットモード時の RC8 プロバイダを選択したコントローラタブ

図 4-19 デンソーロボットモード時の NetwoRC プロバイダを選択したコントローラタブ

CaoSQL でこのコントローラにアクセスしたくない場合は，[このコントローラを有効にする]チェックを外してください．また，CaoSQL が行うこのコントローラのサンプリング周期を msec で設定することができます．

4.3.2. アイテムの追加（デンソーロボットモード）

アイテムの追加もコントローラの追加同様に，通常モードと同じように追加します．

ダイアログはコントローラと同じく，ウィザード形式になっています..

まず,最初にアイテム名を入力します．　(図 4-20)

図 4-20 デンソーロボットモード時のアイテム追加

次に変数を選択します．初めに変数のクラスタイプを選択します．「DENSO NetwoRC」「DENSO RC8」でのクラスの種類はController，Task，Robot，Extension，Fileの5つから選択します．(図 4-21，図 4-22)

図 4-21 デンソーロボットモード時の NetwoRC アイテム追加（2/3）クラス選択画面

図 4-22 デンソーロボットモード時の RC8 アイテム追加（2/3）クラス選択画面

クラスによって使用できる変数が変わるので，追加する変数が属するクラスを選択します．そのクラスの中から利用目的に応じた変数を選択します．(図は Robot クラスを選択したもの)

図 4-23 デンソーロボットモード時の NetwoRC アイテム追加（3/3）変数タイプの選択画面

図 4-24 デンソーロボットモード時の RC8 アイテム追加（3/3）変数タイプの選択画面

変数番号が必要な変数は，[変数番号]欄に数字を入力します．また，[変数情報]の欄では変数毎に何の変数かを説明しています．[OK]を押すと次の画面のようにアイテムの追加が正常に行われています．

変数の種類を変更したい時は，[編集...]ボタンを押します．追加時のダイアログが再度表示されます．

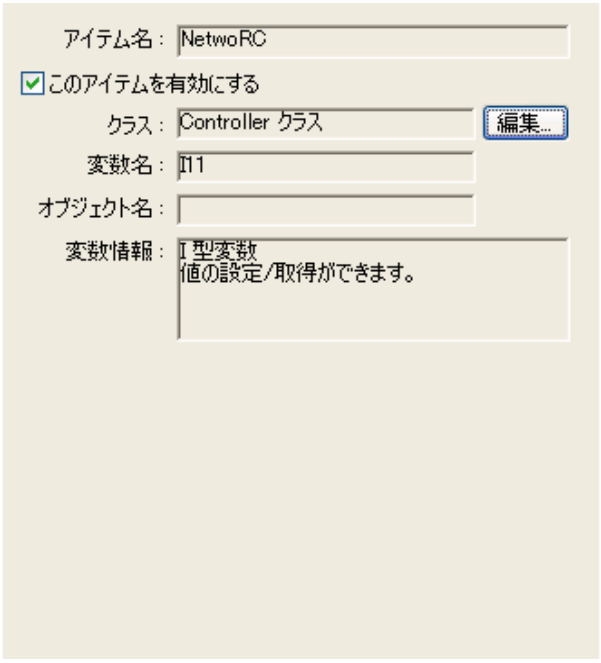


図 4-25 デンソーロボットモード時のアイテムタブ

CaoSQL でこの変数にアクセスしたくない場合は，[このアイテムを有効にする]のチェックを外してください．

4.3.3. 通常モードのファイルの読み込み

デンソーロボットモードでも，Normal モードで保存された設定ファイルを読み込むことができます．
ただし、設定ファイルに「NetwoRC プロバイダ」と「RC8 プロバイダ」以外が設定されている場合は，コントローラタブ，アイテムタブは表示されません．その場合は、下記のメッセージが表示されます．

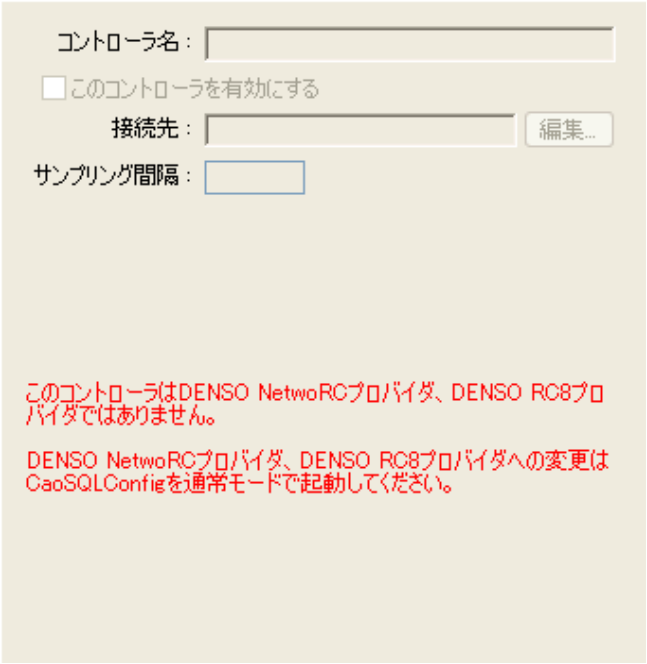


図 4-26 プロバイダの違いによる表示できなかったコントローラタブ

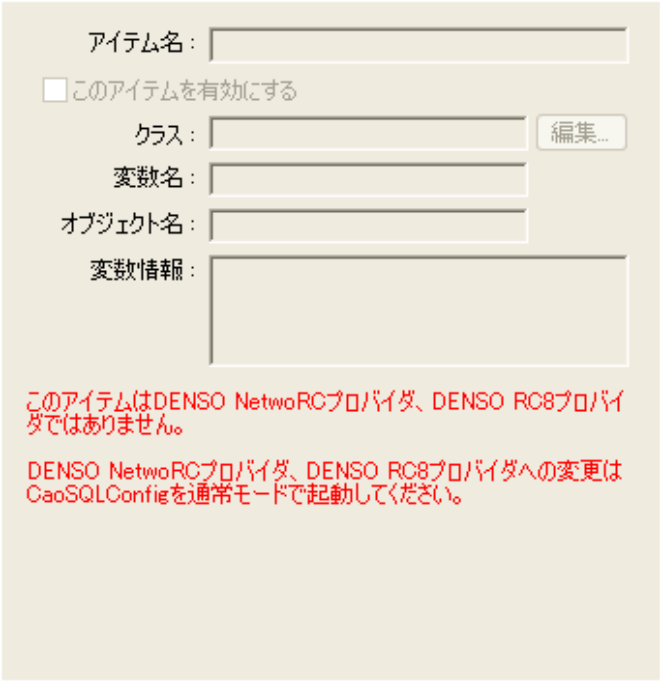


図 4-27 プロバイダの違いによる表示ができなかった変数タブ

【注意事項】

変数名が「NetwoRC プロバイダ」または「RC8 プロバイダ」で使用できない場合は CaoSQL でのアクセスに失敗します. その場合は下記の警告メッセージが表示されます.

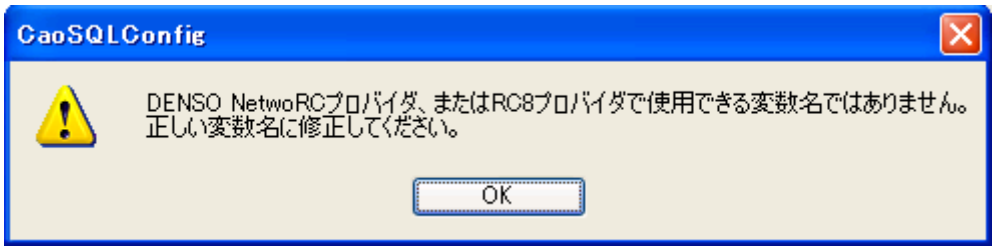


図 4-28 DENSO NetwoRC または RC8 プロバイダで使用できない変数を編集した時の警告

【注意事項】

また、変数番号が不正な場合は、下記の警告メッセージが表示されます

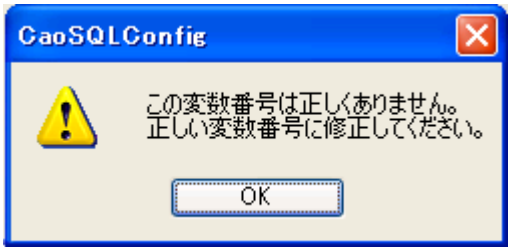


図 4-29 変数番号に文字列が含まれている場合の警告

4.4. レジストリ構成情報

CaoSQLConfig によって設定するレジストリ構成を以下に示します.

CaoSQLConfig では, 太字で示した項目を変更することができます.

登録されているレジストリのキー: HKEY_CLASSES_ROOT¥Software¥CaoSQL

[]:Key Name

<>:Entry Name

{}:Numeric Character

[CaoSQL]		
---<**App.EXEName**>	String	CaoSQLConfig ツールの EXE 名
---<**App.Path**>	String	CaoSQLConfig ツールのパス名
---<**CORBAEnabled**>	DWORD	CORBA サーバの有効/無効スイッチ
---<**CORBAOption**>	String	CORBA サーバの起動オプション
---<**DataPath**>	String	csq ファイルのパス名
---<**DCOMEnabled**>	DWORD	DCOM サーバの有効/無効スイッチ(常に TRUE)
---<**DDEEnabled**>	DWORD	DDE サーバの有効/無効スイッチ
---<**FileName**>	String	ログタイプがファイルの場合のパス名
---<**JNIEnabled**>	DWORD	JNI の有効/無効スイッチ
---<**LCID**>	DWORD	ロケール ID
---<**LogLevel**>	DWORD	ログレベル設定
---<**LogType**>	DWORD	ログタイプ設定
---<**NotRepeatError**>	DWORD	エラーログの繰り返し設定
---<**ProcessPriority**>	DWORD	CaoSQL プロセス優先度
---<**RACCapture**>	DWORD	RAC パケットキャプチャの有効/無効スイッチ
---<**RACS{連番}**>	String	RAC サーバのオプション文字列
---<**RACServerCount**>	DWORD	RAC サーバの登録数
---<**RACServerEnable**>	DWORD	RAC サーバの有効/無効スイッチ

4.5. セッティングTips

ここでは CaoSQL を使用する上で知っておくと便利な設定等を説明します.

4.5.1. 配列要素の分解

あるアイテムの値に配列が格納される場合, 配列要素抽出機能(3.10 参照)とアイテムリンク機能(3.17 参照), CaoVariableの無効を利用することで, 配列の 1 要素を 1 つのCaoSQLItemオブジェクトとして設定することができます.

この設定を利用することで以下のような利点があります.

- サンプリングが 1 度しか行われないので効率的
- 各要素の値の同時性が保障される
- 要素毎にデータ加工(マスク処理など)の設定が行える
- ヒストリ機能は配列に対応していないが, 要素単位でヒストリに記録できる

以下に DataStore プロバイダを利用した具体的な設定例を示します.

例)アイテム値(配列)の要素番号 0 と要素番号 1 の値を取得する設定

(1) 1. コントローラ名 :“CtrlA”を作成し, プロバイダ名に“CaoProv.DataStore”, Write to

Cache に Enable を設定します.

図 4-30 キャッシュ書込み機能設定画面

(2) アイテム名：“ItemA”を作成し，CaoVariable=Enable，リンクアイテムに“ItemA(0),ItemA(1)”を設定します.

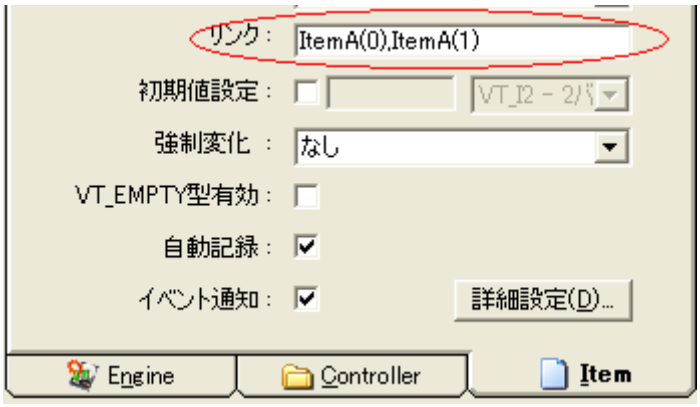


図 4-31 リンクアイテム設定画面

(3) アイテム名：“ItemA(0)”，“ItemA(1)”を作成し，共に CaoVariable を Disable にします．

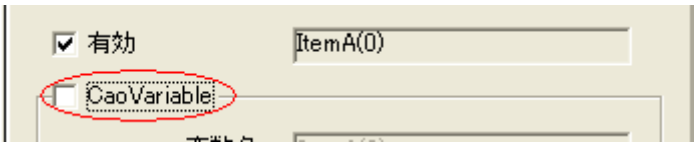


図 4-32 CaoVariable 設定画面

4.5.2. ビット要素の分解

データのマスキング機能（3.11 参照）とアイテムリンク機能（3.17 参照）を利用することで，アイテムの値の 1 ビットを 1 つのCaoSQLItemオブジェクトとして設定することができます．

この設定を利用することで以下のような利点があります．

- サンプリングが 1 度しか行われないので効率的
- 各要素の値の同時性が保障される
- 要素毎にデータ加工（マスク処理など）の設定が行える
- ビット単位でヒストリに記録できる[39]

以下に具体的な設定例を示します．

例）DataStore プロバイダのアイテム値の 1 ビット目と 2 ビット目の値を取得

(1) コントローラ名：“CtrlA”を作成し，プロバイダ名に“CaoProv.DataStore”を設定します．

(2) アイテム名：“ItemB”を作成し，CaoVariable=Enable を設定します．

(3) アイテム名：“ItemB1”を作成し，アイテムの Details ウインドウで Masking=Enable，マスク値=1 を設定する．

(4) アイテム名：“ItemB2”を作成し，アイテムの Details ウインドウで Masking=Enable，マスク値=2 を設定する．

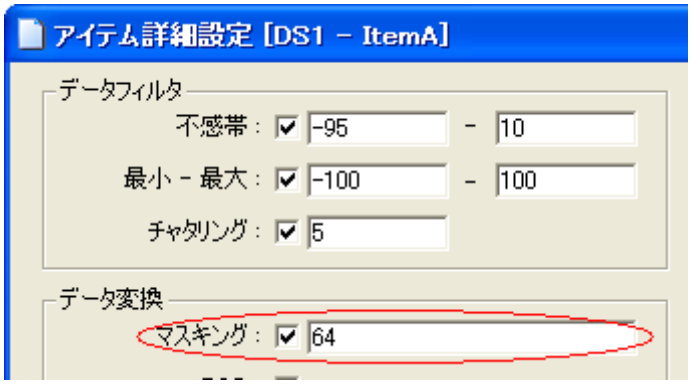


図 4-33 マスク値設定画面

[39]負荷が大きくなるため，お勧めいたしません．

4.5.3. 多項演算

二項演算機能(3.13 参照)において，アイテムリンク機能を併用すれば複雑な演算も全て二項演算に分解できます．

以下に具体例を示します．

例) A←A+B*C-Dという処理を行う場合の設定[40]

(1) アイテム C，アイテム D を作成します．

(2) テンポラリ用アイテム T1 を作成し，アイテムの Details ウインドウで Calculation に“-D”を設定します．

(3) アイテム B を作成し，リンクアイテムにテンポラリ用アイテム T1 を設定，アイテムの Details ウインドウで Calculation に“*C”を設定します．

(4) アイテム A を作成し，同じくアイテムの Details ウインドウで Calculation に“+T1”を設定します．

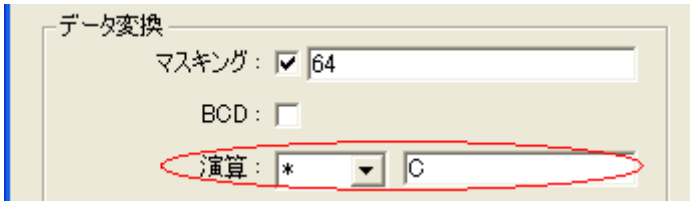


図 4-34 二項演算子設定画面

(5) CaoSQLTester を起動し，アイテム A，B，C，D に値を設定するとアイテム A の値に A+B*C-D の計算結果が設定されます．

4.5.4. 条件(トリガ)によるアイテム伝播(リンク)

コントローラのトリガ機能(3.3 参照)とアイテムの伝播機能(3.17 参照)を併用することで，あるアイテムが条件を満たすと別コントローラのアイテムにデータを伝播(リンク)させることができます．

例えば，図 4-36 の構成のようなことを実現したい場合，以下のような設定手順となります．

(6) コントローラ名：“CtrlA”，“CtrlB”を作成し，“CtrlA”はあるデバイス Provider，“CtrlB”には“CaoProv.DataStore”を設定します．

(7) コントローラ“CtrlA”にアイテム名：“ItemA”，“ItemB”を作成し，両アイテム共に「Variable Name」にデバイスの変数をアイテムとします．次に，コントローラ“CtrlB”にアイテム名：“ItemC”，「Variable Name」に“Item”を設定します．

(8) コントローラ“CtrlA”の Details ウインドウから，「Trigger」=Enable にし，上から順に条件対象のアイテム“ItemA”，条件の種類を「=」，値に「1」，処理1を「Start」，処理 2 を「Stop」に設定します．

[40] こういった用途に対しては，CAO の DataStore プロバイダ(データ共有用プロバイダ)を使うとよいでしょう．また，Ver. 1.8.0 以降では第二演算機能が追加されましたのでより簡単に複雑な演算が可能になりました．

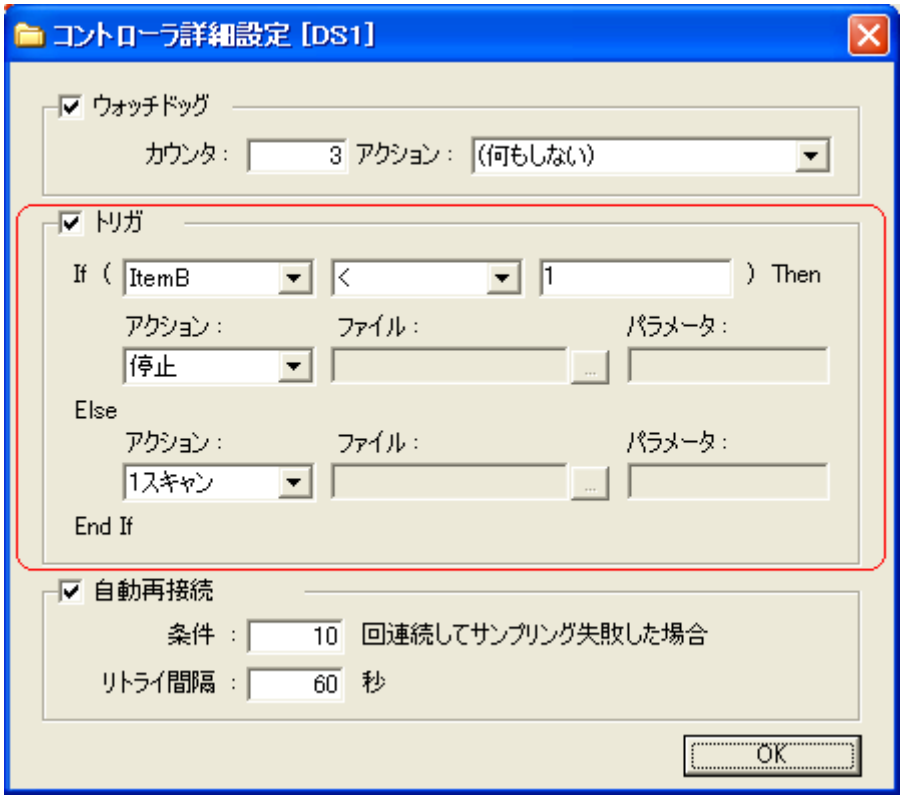


図 4-35 Trigger 設定画面

(9) コントローラ“CtrlA”のアイテム“ItemB”の「Link Item」に「CtrlB¥ItemC」に設定します.

上記, 設定を終えて実行します. 次に処理を順に説明します.

まず, コントローラ“CtrlA”のアイテム“ItemA”が外部から書き込まれた時, コントローラ“CtrlA”のトリガ条件に一致すると, トリガ設定で「Start」としたことにより, コントローラ“CtrlA”に登録されているアイテム全てがサンプリングを開始します. (トリガ動作に関しては3.3 参照) 当然コントローラ“CtrlA”内のアイテム“ItemB”のサンプリングも開始されるので, アイテム“ItemB”の値が変わった時にコントローラ“CtrlB”のアイテム“ItemC”に値の伝播が行われます.

また, トリガ条件と違う値が設定されるとコントローラ“CtrlA”のサンプリングがトリガ対象アイテム以外「Stop」するので, 再度トリガ条件が一致すると, 伝播がまた行われます.

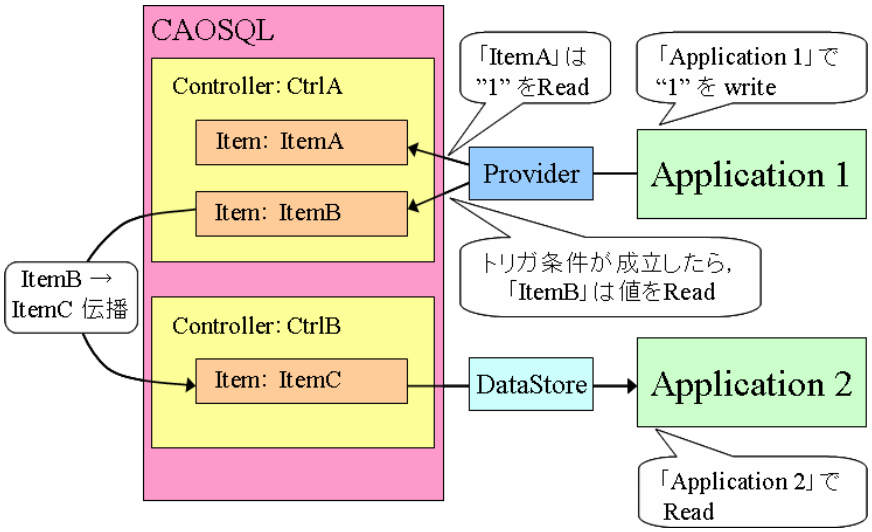


図 4-36 トリガ条件によるアイテム伝播の処理の流れ

4.5.5. VT_I4 から二つのVT_I2 へ分解

VT_I4 の Long 型を I2 の Short 型に分解する方法を紹介します.

以下に具体例を示します.

例) VT_I4 の「1048575」(&HFFFFF)を分解

(1) VT_I2 の格納するアイテムを二つ登録します(VT_2_Hi, VT_2_Lo とします).

(2) これら分解するアイテムに分解元の VT_I4 から Link するように設定します.

(3) “VT_I2_Hi”のアイテムには 16(bit)シフト演算をするような二項演算を, ”VT_I2_Lo”のアイテムは 65535(&HFFFF) と AND をとる二項演算を行うようにします.

(4) 上記の設定を行った後 CaoSQL を実行すると, それぞれ”VT_I2_Hi” には「15」,”VT_I2_Lo”には「65535」が入ります.

VT_I2_Hi

VT_I2_Lo

図 4-37 分解アイテムの二項演算設定（詳細設定）

4.6. プロバイダ・セッティングTips

ここでは CaoSQLConfig で様々なプロバイダに接続する際の設定方法を紹介します.

4.6.1. DataStoreプロバイダ

DataStore プロバイダは，内部に変数テーブルを格納し，その変数テーブルを共有することに
より ORiN2 アプリケーション間でのデータ共有することができます．詳細は DataStore プロバイダユーザーズガイドを参照してください．

CaoSQL で DataStore プロバイダを使用することで，他機器のデータを収集し加工，演算等を行った結果を格納したり，伝播させたり様々な使い方ができます．

コントローラ，アイテム設定を以下に示します．

(1) コントローラの設定

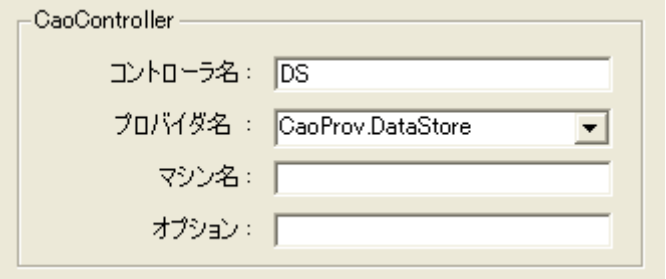


図 4-38 DataStore プロバイダのコントローラ設定画面

・[Controller Name]

任意の名前を入力します．(例 “DS”)

・[Provider Name]

“CaoProv.DataStore”を選択します．

・[Machine]

DataStore プロバイダを動作させるマシン名を指定します．

・[Option]

DataStore プロバイダでは使用しません．

(2) アイテムの設定

図 4-39 DataStore プロバイダのアイテム設定画面

・[Variable Name]
任意の名前を入力します. (例 “Item1”)

・[Option]
AddVariable 時のオプションがあれば, ここに指定します.

・[Class]
DataStore プロバイダの AddVariable は Controller クラスなので,「Controller Class」を指定します.

・[Object Name]
DataStore プロバイダでは使用しません.

4.6.1.1. DataStoreプロバイダのVarsを使用した応用例

DataStoreプロバイダにはVars機能が備わっています. あるデバイスから値を取得しVarsへLinkすることで, 他のVars連携したアプリケーションへ値を渡すことができます. また, デバイスから取得/書き込みの際に, 一旦Varsを経由することにより二項演算等で直接デバイスの値に影響を与えないようにすることも可能です. (図 4-40 参照)

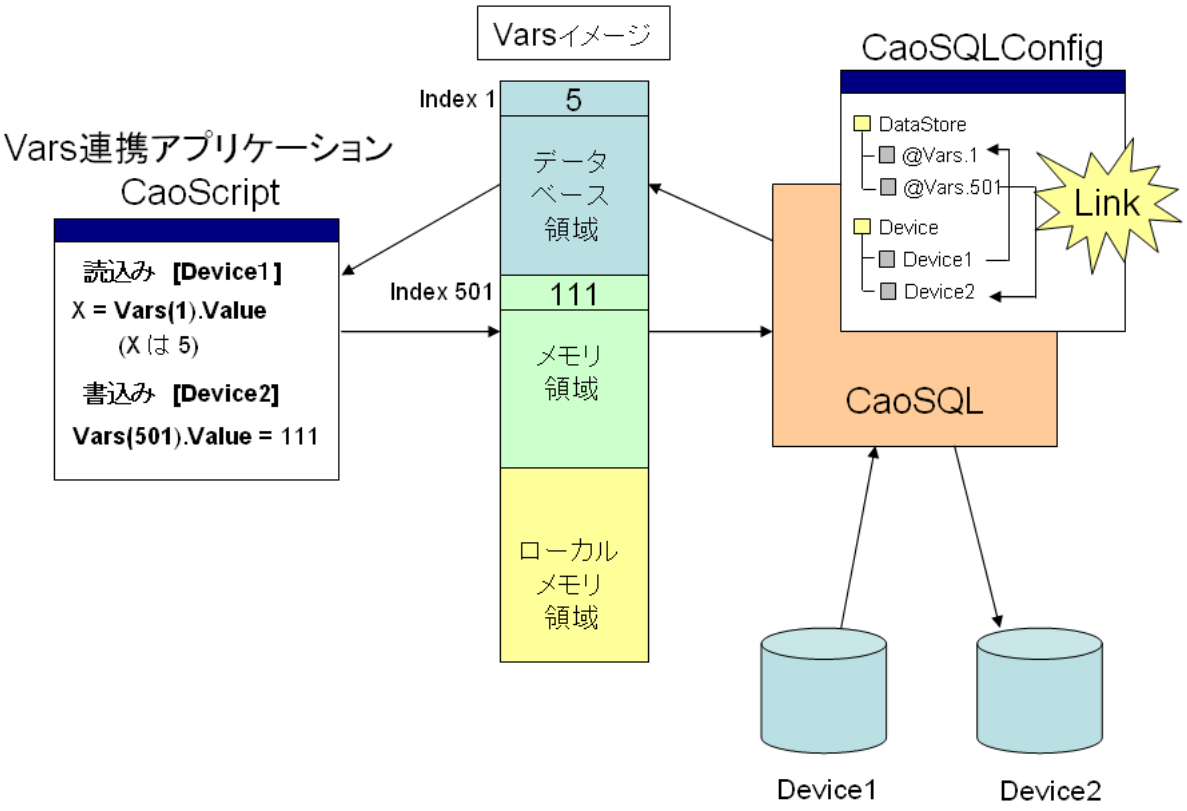


図 4-40 DataStore プロバイダの Vars 機能を使用した例

Varsを使用する場合，CaoSQLConfigではDataStoreプロバイダを指定したコントローラに対して，ItemName:"@Vars.1"と指定します．こうすることにより，Varsの 1 番目を使用するという手順になります．詳細は，DataStoreプロバイダユーザーズガイドを参照してください．

4.6.2. DataBaseプロバイダ

CaoSQL で DataBase プロバイダを使用することにより，データベースの任意のフィールドの任意のレコードへ容易にアクセスすることができます．データベースに現在値を格納するテーブルを作りたい場合に便利です．

下記の例では，DataBase プロバイダで使用するデータベースとして，“Sample.mdb” (Microsoft Access 2003) を使用しています．“Sample.mdb”のデータベース構成は，テーブル名[TestTable]，フィールド名[FieldData1(テキスト型，主キー)，FieldData2(テキスト型)]，データとして“FieldData1”フィールドの値を“Test”にあらかじめ設定しています．接続文字列は“Microsoft Access”用の設定となります．詳細は DataBase プロバイダユーザーズガイドを参照してください．

コントローラ，アイテムの設定を以下に示します．

(1) コントローラの設定

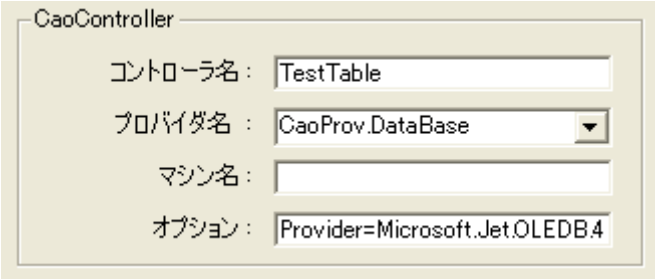


図 4-41 DataBase プロバイダのコントローラ設定画面

- [Controller Name]： アクセス先となるデータベースのテーブル名を指定します．(例 ：TestTable)
- [Provider Name] : “CaoProv.DataBase”を選択します．
- [Machine] : DataBase プロバイダを動作させるマシン名を指定します．
- [Option]： データベースに接続するADOの接続文字列を指定します．[41]

（例）

Provider=Microsoft.Jet.OLEDB.4.0,
Data Source=D:¥ORiN2¥CaoSQL¥Bin¥Sample.mdb,Key=FieldData1)

テーブル名，接続文字列等の指定が正しくなされていない場合はエラーとなります．カンマやスペースなどにも注意して入力してください．

[41] 他データベースへの接続文字列は“DataBaseプロバイダ ユーザーズガイド”を参照してください．

(2) アイテムの設定

図 4-42 DataBase プロバイダのアイテム設定画面

・[Variable Name]： 対象のフィールド名を入力します. (例 “FieldData2”)
・[Option] : AddVariable 時のオプションがあれば, ここに指定します. (例 : “FieldData2”)
・[Class] : DataBase プロバイダは「Extension Class」を指定して AddExtension を実行します.
・[Object Name]
DataBase プロバイダでは対象レコードを特定するために, コントローラ設定の接続文字列で指定した Key のフィールドの値を指定します. (例 : “Test”)

コントローラ設定の接続文字列の“Key”にフィールド名を指定し, アイテム設定の[Object Name]でキーの値を指定することで DataBase プロバイダがテーブル内のどのレコードにアクセスするかを特定することができます.

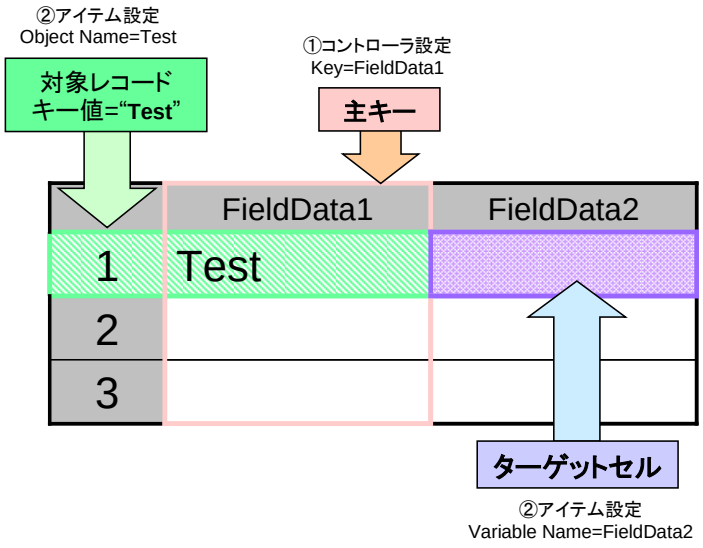


図 4-43 ターゲットセルの設定

CaoSQL で DataBase プロバイダを使用する際には, 前述したようにデータベースのデータが存在していなければ, エラーとなります. DataBase プロバイダの機能として, レコードを追加することも可能ですが, 現在 CaoSQL からはこの機能は使用できません.

5. CaoSQLチュートリアル

5.1. 概要

本章では CaoSQL を利用したシステムの構築方法について解説します. 使用するデータベースとしては, Access2000 と PostgreSQL を例に挙げて説明を行います.

5.2. Access2000 によるデータベースの設定

ORiN2 SDK では, Access2000 のデータベースファイルのスケルトンが用意されています. Jet データベースエンジンがインストールされていれば, Access2000 がインストールされていなくても, CaoSQL のヒストリ機能を利用することができます.

このデータベースファイルを利用するには, CaoSQLConfig を起動し, “History”ボタンを押下し, 以下のように項目を設定して下さい.

- OLEDB Provider : Microsoft.Jet.OLEDB.4.0
- Data Source :
  skeleton2000_en.mdb の絶対パス(ORiN2 SDK インストールディレクトリの¥CaoSQL¥Bin にあります)
- Table Name : caosql_history

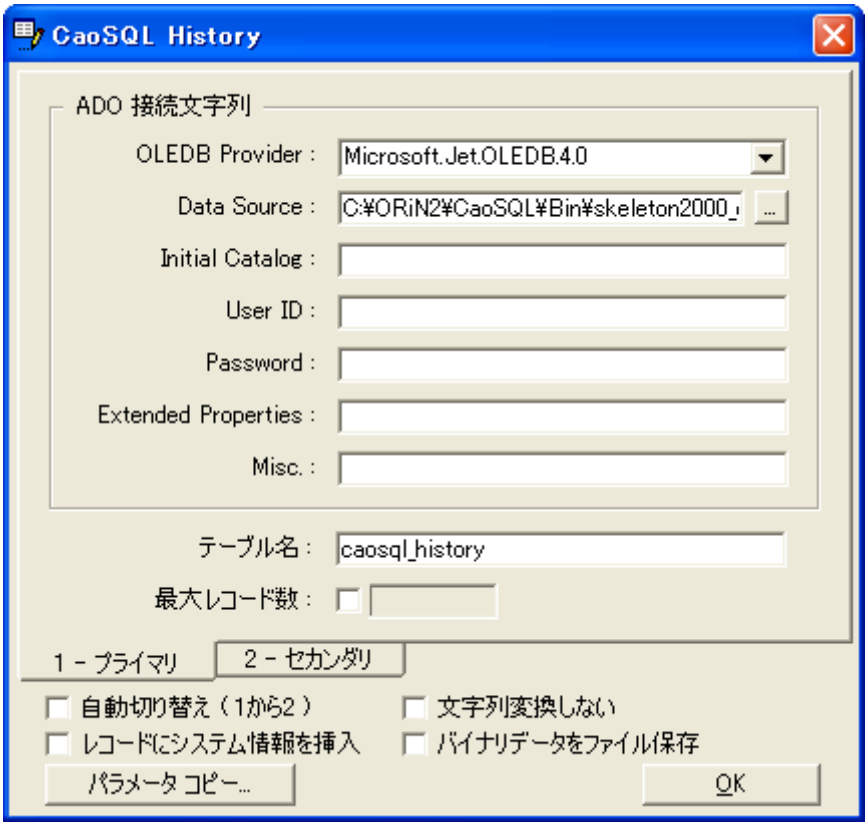


図 5-1 Access2000 のデータベース設定

5.3. CaoSQLConfigによるコントローラとアイテムの登録

ここでは，CaoSQLConfigを利用して，CaoSQLで使用するコントローラやアイテムの設定をおこないます．詳細なCaoSQLConfigの利用方法については，「CaoSQLTools ユーザーズガイド」を参照して下さい．

まず，「Engine」タブの「History」ボタンを押下して下さい．ここでは，利用するデータベースの設定をおこないます．

次に，コントローラの設定をおこないます．今回は動作確認を行うためにDataStoreプロバイダを利用します．メニューの「Edit」→「Add Controller」を選択し，図 5-2 に示すように各項目の値を設定して下さい．

次にアイテムの設定をおこないます．メニューの「Edit」→「Add Item」を選択しアイテム名を入力します．ここでは，“ItemA”，“ItemB”，“ItemC”と入力します．

本章では，ヒストリ機能の確認のみしか行いませんので，これ以外の特別な設定変更は行いません．ヒストリ機能以外の機能を利用する場合は，第3 章CaoSQLの機能を参照して下さい．

図 5-2 コントローラの追加

5.4. クライアントアプリケーションの作成

ここでは，Visual Basic6.0 を利用して，CaoSQL のクライアントアプリケーションを作成します．まずは，「新規作成」から“標準 EXE”を選択して下さい．

(1) 参照設定の追加

CaoSQLを利用するためには，参照設定を追加する必要があります．メニューから「プロジェクト」→「参照設定」を選択し，図 5-3 に示すように，以下のライブラリファイルを追加して下さい．

- CaoSQL 1.0 タイプライブラリ
- Microsoft ActiveX Data Objects 2.7 Library

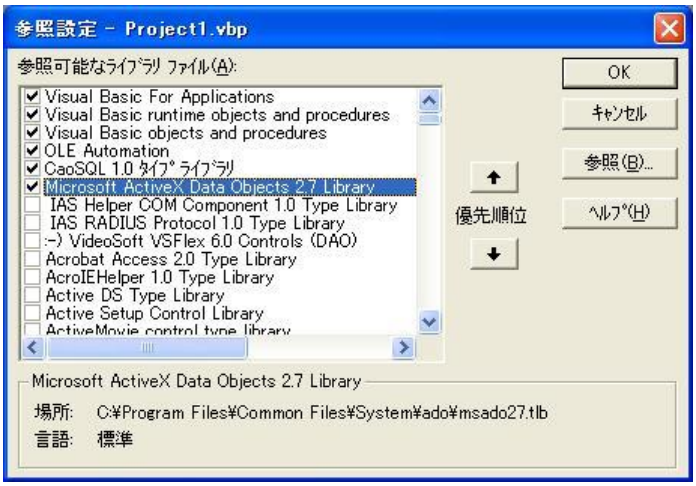


図 5-3 参照設定の追加

(2) フォームの作成

次に，以下に示すような，コンボボックス，テキストボックス，ボタンを持つフォームを作成して下さい．赤字は，それぞれのオブジェクトのオブジェクト名を示しています．

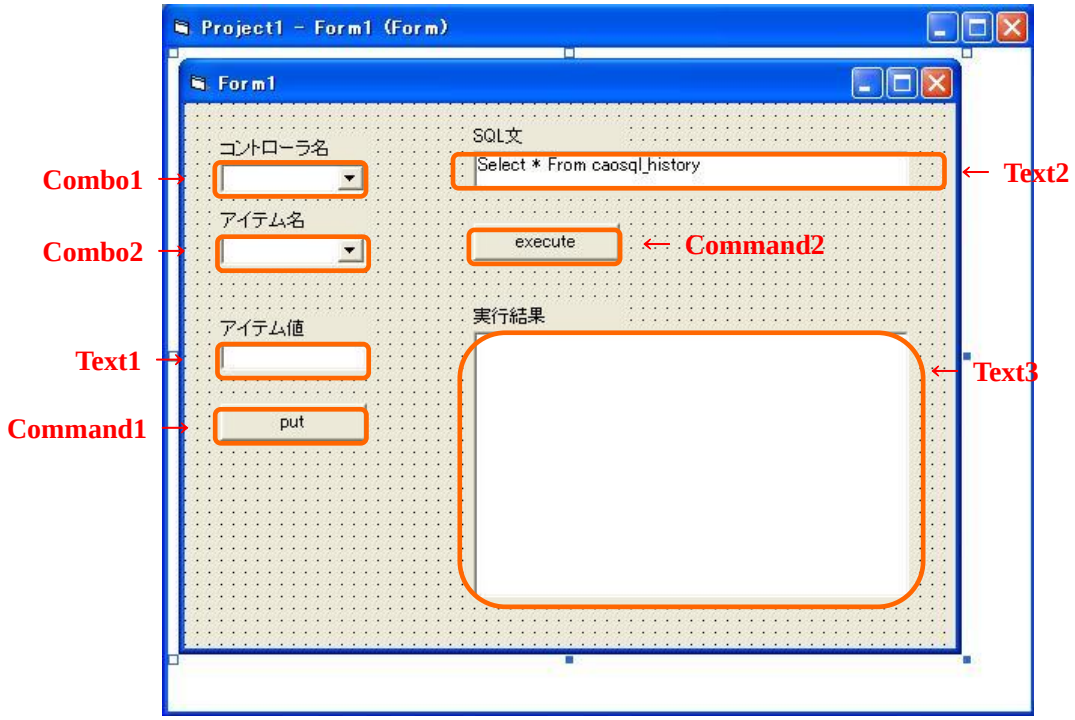


図 5-4 クライアントアプリケーションのフォーム

(3) ソースコードの記述

以下のようなソースコードを記述して下さい．

List 5-1 Form1.frm

```
Private csqlEng As CaoSQLEngine         'CaoSQL のエンジン
Private csqlHis As CaoSQLHistory        'CaoSQL のデータベース
Private csqlCtrl As CaoSQLController    'CaoSQL のコントローラ
Private csqlItem As CaoSQLItem          'CaoSQL のアイテム
Private recSet As ADODB.Recordset       'SQL 文実行結果


Private Sub Form_Load()
    Set csqlEng = New CaoSQLEngine
    Set csqlHis = csqlEng.CaoSQLHistory

    '登録されているコントローラ名を取得しコンボボックスに展開
    Dim controllerName As Variant
    For Each controllerName In csqlEng.ControllerNames
        Combo1.AddItem (controllerName)
    Next
End Sub

Private Sub Combo1_Click()
    '選択されたコントローラの取得
    Set csqlCtrl = csqlEng.Controller(Combo1.Text)
    '選択されたコントローラのアイテム一覧をコンボボックスに展開
    Dim itemName As Variant
    For Each itemName In csqlCtrl.ItemNames
        Combo2.AddItem (itemName)
    Next
End Sub

Private Sub Combo2_Click()
    '選択されたアイテムを取得
    Set csqlItem = csqlCtrl.Item(Combo2.Text)
    Command1.Enabled = True
End Sub

Private Sub Command1_Click()
    'テキストボックス 1 の値をアイテムに書き込む
    csqlItem.Value = Text1.Text
End Sub

Private Sub Command2_Click()
    'テキストボックス 2 の SQL 文を実行
    Set recSet = csqlHis.Execute(Text2.Text)
    '実行結果をテキストボックス 3 に表示
    Dim cnt As Integer
    Text3.Text = ""
    Do Until recSet.EOF
        For cnt = 0 To recSet.Fields.Count - 1
            Text3.Text = Text3.Text & " " & recSet(cnt).Value
        Next
        Text3.Text = Text3.Text & vbNewLine
        recSet.MoveNext
    Loop
End Sub
```

5.5. サンプルの実行

(1) CaoSQL の起動

CaoSQLLauncher.exe を起動し，“Start”ボタンを押下して下さい．

(2) クライアントアプリケーションの起動

5.4 で作成したクライアントアプリケーションを起動して下さい．

起動に成功すると，以下のような画面が表示され，コントローラ名のコンボボックスをクリックすると，CaoSQLConfig で設定したコントローラ名の一覧が表示されます．

図 5-5 クライアントアプリケーションの起動

(3) データの書き込み

図 5-6 に示すように，クライアントアプリケーションのコントローラ名コンボボックス，アイテム名コンボボックスから，適当なコントローラ名とアイテム名を選択し，アイテム値テキストボックスに値を入力し「put」ボタンを押下して下さい．

ここでは，コントローラのアイテム値を変更しているだけですが，ヒストリ機能により，変更した項目がデータベースへと登録されます．

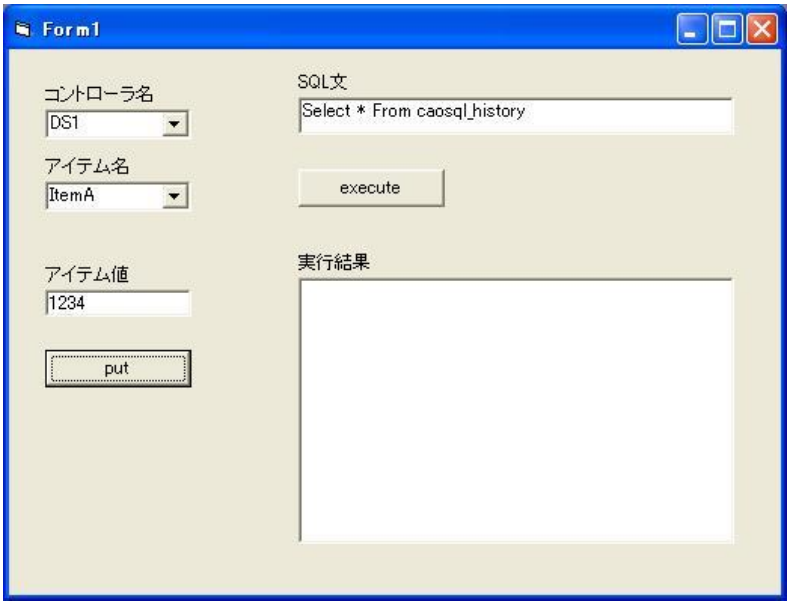


図 5-6 アイテムの書き込み

(4) SQL 文の実行

「execute」ボタンを押下すると，“Select * From caosql_history”という SQL 文が実行され，以下に示すように，先ほど書き込んだデータの一覧がテキストボックスに表示されます．

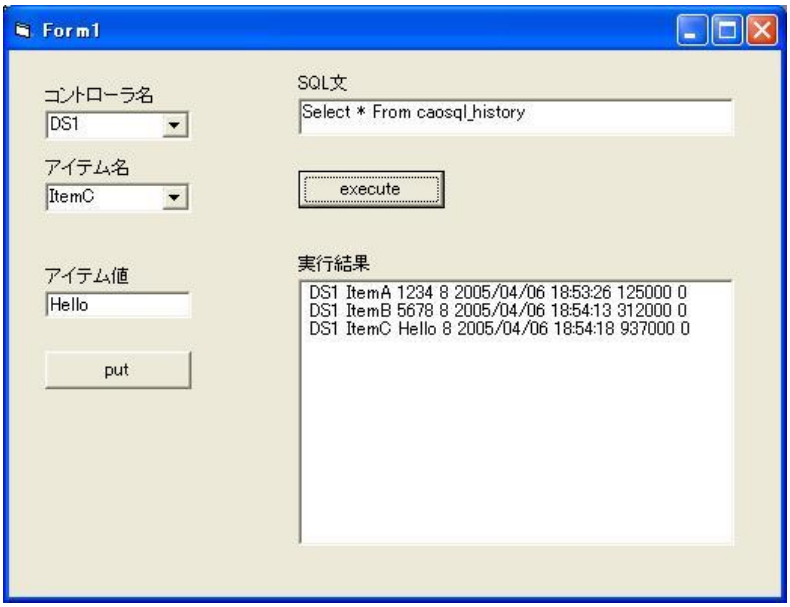


図 5-7 SQL 文の実行

付録A. 付録

付録A.1. CaoSQL API一覧

◆ CaoSQLEngine Object ― エンジン

オブジェクト	プロパティ, メソッド, イベント		説明	R/W /S	関数の引数 IN	関数の引数 OUT RETVAL	備考
CaoSQLEngine	Controller	P	コントローラオブジェクトの取得	R	コントローラ名／コントローラ番号：VARIANT	コントローラオブジェクト：ICaoSQLController	デフォルトメンバ
	Count	P	コントローラ数の取得	R		コントローラ数：Long	
	ControllerNames	P	コントローラ名リストの取得	R		コントローラ名リスト：VARIANT (VT_ARRAY|VT_BSTR)	
	CaoSQLHistory	P	ヒストリオブジェクトの取得	R		ヒストリオブジェクト：ICaoSQLHistory	
	Execute	M	拡張コマンドの実行	S	コマンド：VARIANT	実行結果：VARIANT	機能拡張用
	Snapshot	M	全コントローラの全アイテムの現在値をヒストリ DB に保存	W			
	Start	M	全てのサンプリングスレッドの開始	W			
	Stop	M	全てのサンプリングスレッドの停止	W			
記号の意味	M：メソッド P：プロパティ E：イベント			(注1)	・[]内は省略可能引数. ・BSTR 型の省略可能引数のデフォルト値は空文字. ・数値型の省略可能引数のデフォルト値はゼロ.		

(注1)記号の意味は次の通り.
R －Read : コントローラ, またはプロバイダのステータスやコンフィギュレーションを取得する.
W －Write : コントローラのステータスや, コンフィギュレーションを変化させる.
S －Setup : プロバイダのステータスやコンフィギュレーションを変化させる.

◆ CaoSQLController Object ― コントローラ

オブジェクト	プロパティ，メソッド，イベント		説明	R/W /S	関数の引数		備考
					IN	OUT RETVAL	
CaoSQLController	Count	P	アイテム数の取得	R		アイテム数:long	
	Description	P	コントローラの詳細説明	R		詳細:BSTR	
	Index	P	アイテム番号の取得	R		アイテム番号:long	
	Item	P	アイテムオブジェクトの取得	R	アイテム名／アイテム番号:VARIANT	オブジェクト:ICaoSQLItem	デフォルトメンバ
	ItemNames	P	アイテム名リストの取得	R		アイテム名リスト:VARIANT（VT_ARRAY|VT_BSTR）	
	Name	P	コントローラ名	R		コントローラ名:BSTR	
	ScanTimes	P	スキャンタイムの取得	R	スキャンタイム種別番号:long	スキャンタイム:long	スキャンタイム種別番号については enmCaoSQLCtrlScantime を参照
	SettingData	P	各設定データの取得	R	データ番号/データ名:VARIANT	データ:VARIANT	データ名前／データ番号については get_SettingData 引数一覧を参照. 書き込みは実装されていません.
	State	P	コントローラの状態取得	R		ステータス:long	enmCaoSQLCtrlState を参照
	Tag	P	タグ	R/W	タグ:BSTR	タグ:BSTR	多目的タグ
	AddItem	M	アイテムの動的追加	W	アイテムデータ:VARIANT		
	RemoveItem	M	アイテムの削除	W	アイテム名／アイテム番号:VARIANT		AddItem により動的に追加されたアイテムしか削除できません.
	Snapshot	M	全アイテムの現在値をヒストリ DB に保存	W			
	Start	M	サンプリングスレッドの開始	W			
	Stop	M	サンプリングスレッドの停止	W			

	Execute	M	コマンド文字列の実行	S	コマンド:BSTR パラメータ:VARIANT		CaoController の Execute にコマンド文字列を渡し，実行します.
	OnChangeItem	E	値変更イベント	−		CaoSQLItem インターフェースポインタ:ICaoSQLItem*	
	OnChangeState	E	ステータス変更イベント	−		ステータス値:long	ステータス値は enmCaoSQLCtrlState を参照
記号の意味	M:メソッド P:プロパティ E:イベント			(注1)	・[]内は省略可能引数. ・BSTR 型の省略可能引数のデフォルト値は空文字. ・数値型の省略可能引数のデフォルト値はゼロ.		

◆ enmCaoSQLCtrlState

Enum	備考
0: csCtrlThreadUninit	スレッドは未初期化です
1: csCtrlThreadActive	スレッドは稼動中です
2: csCtrlThreadDeactive	スレッドは停止中です
3: csCtrlThreadError	スレッドでエラーが発生しました
4: csCtrlThreadTerminated	スレッドは終了しました

◆ enmCaoSQLCtrlScantime

Enum	備考
0: csCtrlScantimeLast	最新スキャンタイム
1: csCtrlScantimeMin	最小スキャンタイム
2: csCtrlScantimeMax	最大スキャンタイム

◆ get_SettingData 引数一覧

プロパティ番号	プロパティ名	意味
0	Controller Name	コントローラ名
1	Provider Name	プロバイダ名
2	Machine	コンピュータ名
3	Option	オプション
4	Use History	History の自動記録有効/無効
5	Write to Cache	キャッシュ書込み機能有効/無効
6	Time Stamp	タイムスタンプ
7	Sampling Rate	サンプリング間隔
8	Distribute Equally	均等配分有効/無効
9	Startup	スタートアップ登録有効/無効
10	Priority	スレッド優先度
11	（名前指定なし）	ウォッチドッグタイマ有効/無効
12	（名前指定なし）	ウォッチドッグタイマカウンタ
13	（名前指定なし）	ウォッチドッグタイマアクション
14	（名前指定なし）	自動再接続有効/無効
15	（名前指定なし）	自動再接続条件
16	（名前指定なし）	自動再接続リトライ間隔
17	（名前指定なし）	コントローラの種類
18	（名前指定なし）	スクリプト 有効/無効
19	（名前指定なし）	スクリプト ソース
20	（名前指定なし）	（非公開）

◆ CaoSQLItem Object ― アイテム

オブジェクト	プロパティ，メソッド，イベント		説明	R/W /S	関数の引数		備考
					IN	OUT RETVAL	
CaoSQLItem	Activation	P	アイテム値変化イベント ON/OFF の取得/設定	R/W	設定：VARIANT_BOOL	設定：VARIANT_BOOL	
	Attribute	P	アイテム属性の取得	R		アイテム属性：long	enmCaoSQLItemAttribute を参照
	DateTime	P	タイムスタンプ（日時）の取得	R		タイムスタンプ：VARIANT	
	Description	P	コントローラの詳細説明	R		詳細：BSTR	
	Element	P	配列要素の値取得	R	1 次元目の要素：long 2 次元目の要素：long 3 次元目の要素:：long	値：VARIANT	書き込みは実装されていません.
	ID	P	アイテム ID	R/W	アイテム ID：VARIANT	アイテム ID：VARIANT	
	Index	P	アイテム番号	R		アイテム番号：long	
	LinkActivation	P	リンク先アイテムの値変化イベント ON/OFF の取得/設定	R/W	設定：VARIANT_BOOL	設定：VARIANT_BOOL	
	LinkItemNames	P	リンク先アイテム名	R		リンク先アイテム名： VARIANT(VT_ARRAY|VT_BSTR)	フォーマットは ”[<LinkControllerName>:]<LinkItemName>” LinkControllerName を省略した場合，所属するコントローラから Item を探します.
	Microsecond	P	タイムスタンプ（マイクロ秒）の取得	R		タイムスタンプ：long	
	Name	P	アイテム名の取得	R		アイテム名：BSTR	

	SettingData	P	各設定データの取得	R	データ番号/データ名 : VARIANT	データ:VARIANT	データ番号/データ名については get_SettingData 引数一覧を参照. 書き込みは実装されていません.
	State	P	アイテムの状態取得	R		ステータス:long	enmCaoSQLItemState を参照
	Tag	P	タグ	R/W	タグ:VARIANT	タグ:VARIANT	多目的タグ
	Value	P	アイテムの値取得/設定	R/W	値:VARIANT	値:VARIANT	デフォルトメンバ
	Snapshot	M	現在値をヒストリ DB に保存	W			
記号の意味	M:メソッド P:プロパティ E:イベント			(注1)	・[]内は省略可能引数. ・BSTR 型の省略可能引数のデフォルト値は空文字. ・数値型の省略可能引数のデフォルト値はゼロ.		

◆ enmCaoSQLItemState

Enum	備考
0: csItemSucceeded	アイテムの値設定/取得は成功しました
1: csItemFailed	アイテムの値設定/取得は失敗しました
2: csitemUndefined	アイテムの値は不定状態です

◆ enmCaoVariableClass

Enum	備考
0: csVarClassController	コントローラクラスの変数
1: csVarClassTask	タスククラスの変数
2: csVarClassRobot	ロボットクラスの変数
3: csVarClassExtension	拡張クラスの変数

◆ enmCaoSQLItemAttribute

Enum	備考
0: csItemReadWrite	アイテムは読み取り/書き込み属性
1: csItemReadOnly	アイテムは読み取り専用属性
2: csItemWriteOnly	アイテムは書き込み専用属性

◆ get_SettingData 引数一覧

プロパティ番号[42]	プロパティ名	意味
0	Variable Name	CaoVariable 名
1	Option	CaoVariable の作成時オプション
2	Class	CaoVariable の親クラス種別 0 Controller クラス 1 Task クラス 2 Robot クラス 3 Extension クラス 4 File クラス
3	Object Name	CaoVariable の親オブジェクト名
4	Initialized	初期化有無設定
5	Initial Data	初期化値
6	Initial Data Type	初期化型 （VARTYPE 型） 2 VT_I2 3 VT_I4 4 VT_R4 5 VT_R8 6 VT_CY 7 VT_DATE 8 VT_BSTR 11 VT_BOOL 12 VT_VARIANT 16 VT_I1 17 VT_UI1 18 VT_UI2
7	Use Empty	VT_EMPTY 使用有無設定
8	History	History 使用有無設定

[42] 「表 3-11 AddItem メソッドへの引数」の配列要素番号とプロパティ番号は同じ値ではありません. 配列要素番号が 3 以下の場合は(プロパティ番号+2), 3 より大きい場合は(プロパティ番号+5)が配列要素番号になります.

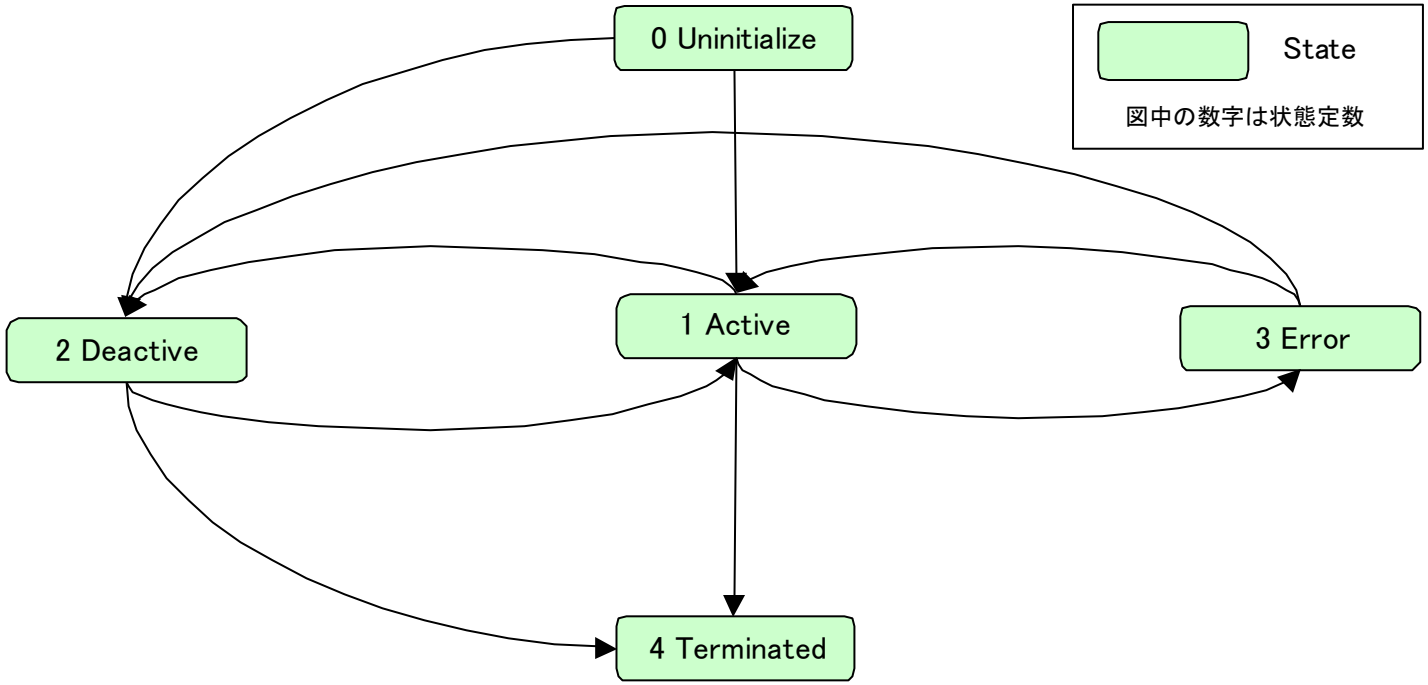
9	OnChange Event	OnChange イベント有無設定
10	Force to Change	値強制認識有無設定 0 なし 1 読み込み時 2 書き込み時
11	(名前指定なし)	不感帯有無設定
12	(名前指定なし)	不感帯最小値
13	(名前指定なし)	不感帯最大値
14	(名前指定なし)	感帯有無設定
15	(名前指定なし)	感帯最小値
16	(名前指定なし)	感帯最大値
17	(名前指定なし)	チャタリング有無設定
18	(名前指定なし)	チャタリング値
19	(名前指定なし)	マスキング有無設定
20	(名前指定なし)	マスク値
21	(名前指定なし)	BCD 変換有無設定
22	(名前指定なし)	二項演算処理演算子(第一演算) 0 (NOP) 1 + 2 − 3 * 4 / 5 MOD 6 AND 7 OR 8 XOR 9 < 10 >> 11 = 12 <> 13 <= 14 < 15 >= 16 > 17 CAT 18 IDIV 19 IMP 20 POW 21 ABS 22 FIX 23 INT 24 NEG 25 NOT 26 RAD2DEG 27 DEG2RAD
23	(名前指定なし)	二項演算処理演算値設定(第一演算)
24	(名前指定なし)	データ型変換処理有無設定(読込み時)

25	(名前指定なし)	データ型変換処理のデータ型設定(読込み時, VARTYPE 型) 2 VT_I2 3 VT_I4 4 VT_R4 5 VT_R8 6 VT_CY 7 VT_DATE 8 VT_BSTR 11 VT_BOOL 12 VT_VARIANT 16 VT_I1 17 VT_UI1 18 VT_UI2 19 VT_UI4
26	(名前指定なし)	データ型変換処理有無設定(書込み時)
27	(名前指定なし)	データ型変換処理のデータ型設定(書込み時, VARTYPE 型) 2 VT_I2 3 VT_I4 4 VT_R4 5 VT_R8 6 VT_CY 7 VT_DATE 8 VT_BSTR 11 VT_BOOL 12 VT_VARIANT 16 VT_I1 17 VT_UI1 18 VT_UI2 19 VT_UI4
28	(名前指定なし)	配列内データ比較有無設定
29	(名前指定なし)	配列要素分配リンク設定
30	(名前指定なし)	不活性化
31	(名前指定なし)	アイテムタイプ 0 通常アイテム 1 エイリアスアイテム 2 配列型アイテム
32	(名前指定なし)	リンク先プロパティ 0 Value 1 ID 2 Activation 3 LinkActivation
33	(名前指定なし)	二項演算処理演算子(第一演算)
34	(名前指定なし)	二項演算処理演算値設定(第二演算)

◆ CaoSQLHistoryObject ― ヒストリ

オブジェクト	プロパティ, メソッド, イベント		説明	R/W /S	関数の引数		備考
					IN	OUT RETVAL	
CaoSQLHistory	ConnectionString	P	History の ADO 接続文字列	R	プライマリ/セカンダリ指定:long	接続文字列:BSTR	プライマリは 1, セカンダリは 2 を指定します.
	Enable	P	ヒストリ有効/無効	R		ヒストリ有効/無効設定:VARIANT_BOOL	
	MaxRecord	P	レコード制限数	R	プライマリ/セカンダリ指定:long	最大レコード数:long	プライマリは 1, セカンダリは 2 を指定します. 出力が 0 の場合, レコード数制限は“なし”を意味します.
	TableName	P	テーブル名	R	プライマリ/セカンダリ指定:long	テーブル名:BSTR	プライマリは 1, セカンダリは 2 を指定します.
	Execute	M	SQL 文の実行	-	SQL 文:BSTR	ADO::Recordset:VARIANT	入力された SQL 文をそのまま実行します.
記号の意味	M:メソッド P:プロパティ E:イベント			(注1)	・[]内は省略可能引数. ・BSTR 型の省略可能引数のデフォルト値は空文字. ・数値型の省略可能引数のデフォルト値はゼロ.		

付録A.3. CaoSQLController状態遷移図



付録A.4. CaoSQLItem状態遷移図