IoTDS プロバイダ

IoT Data Share / IoT Data Server

Version 1.3.0

ユーザーズ ガイド

October 28, 2024

【備考】

【改版履歴】

バージョン	日付	内容
1.0.0	2017-10-31	初版.
1.1.0	2018-01-29	ローカル接続時の問題を修正.
1.2.0	2019-09-24	プロジェクトファイルデータベース化対応 $TREE_LIST$のプロジェクト検索順について追記 バグ修正
1.2.1	2020-11-01	$TREE_LIST$処理でプロジェクト名に英数字以外の文字が含まれていると正しく取得できなくなってしまうのを修正 メモリ関連の処理修正 全体的に処理を修正
1.2.2	2021-03-26	読み取り専用アイテムに対して書き込みを行った場合にエラーとして出力されるように修正
1.2.3	2021-04-01	ローカル接続失敗時のエラーメッセージの綴りを修正
1.2.4	2021-08-19	起動, 終了時処理の修正 再接続時処理の修正 メモリ関連処理の修正 排他処理の修正 メッセージ処理の非同期対応
	2021-12-20	誤字修正 OpenSSL バージョンアップ.
	2023-10-09	IoTDS の設定方法を追記. CaoVariable::get_ID および CaoVariable::put_ID の説明を追記. AddController 時のデフォルトポート番号を追記. サンプルプログラム追加.
1.3.0	2024-10-28	UseCB オプション、CBPort オプション追加

【動作確認機器】

機種	バージョン	注意事項

目次

1. はじめに

IoTDS プロバイダは，IoT Data Share，IoT Data Server に対しデータの書き込み/読出しを行う ORiN2 CAO プロバイダです.

本ドキュメントでは，IoTDS プロバイダの概要と，実装されている CAO インタフェース（関数仕様）について説明しています.

2. プロバイダの概要

2.1. 概要

IoTDS プロバイダは， IoT Data Share，または IoT Data Server に 対してデータの書き込み/読み出しを行う CAO プロバイダです．

接続先には，ローカルマシン上の IoT Data Share，またはネットワーク上にある IoT Data Share/IoT Data Server を選択することが可能です．

ネットワーク上の IoT Data Share/IoT Data Server に接続する場合には，b-CAP を使用し，IoTDS プロバイダは b-CAP クライアントとして動作します．このため，接続先には b-CAP サーバが起動していなければなりません．

ローカルマシン上の IoT Data Share に接続する場合には，直接接続を行います．

ファイル形式は DLL(Dynamic Link Library)であり，CAO エンジンから使用時に動的にロードされます．IoTDS プロバイダを使用するにあたっては ORiN2SDK をインストールするか，下表を参照して手作業でレジストリ登録を行う必要があります．

表 2-1 IoTDS プロバイダ

ファイル名	CaoProvDENSOIoTDS.dll
ProgID	CaoProv.DENSO.IoTDS
レジストリ登録	regsvr32 CaoProvDENSOIoTDS.dll
レジストリ登録の抹消	regsvr32 /u CaoProvDENSOIoTDS.dll

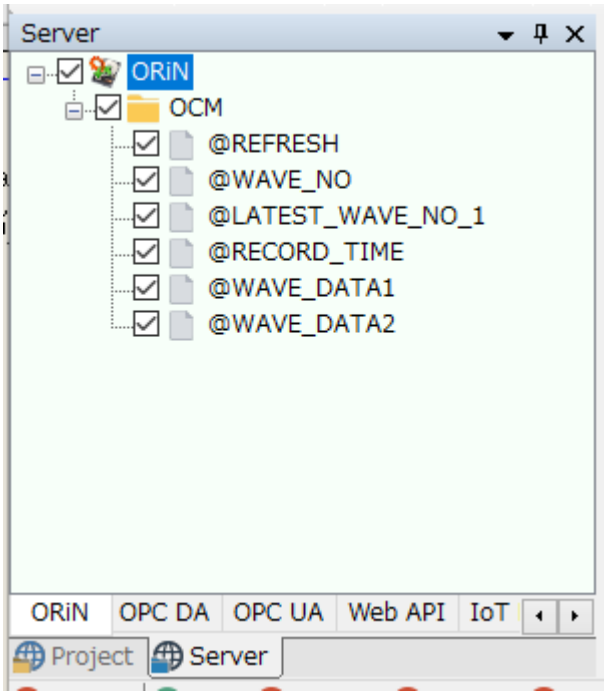
IoTDS プロバイダは，接続先となる IoT Data Share または IoT Data Server のプロジェクトで設定されているコントローラに対し以下のような機能を提供します．

- コントローラに対するコマンド実行
- アイテムに対する情報の取得
- アイテムに対する値の取得/設定

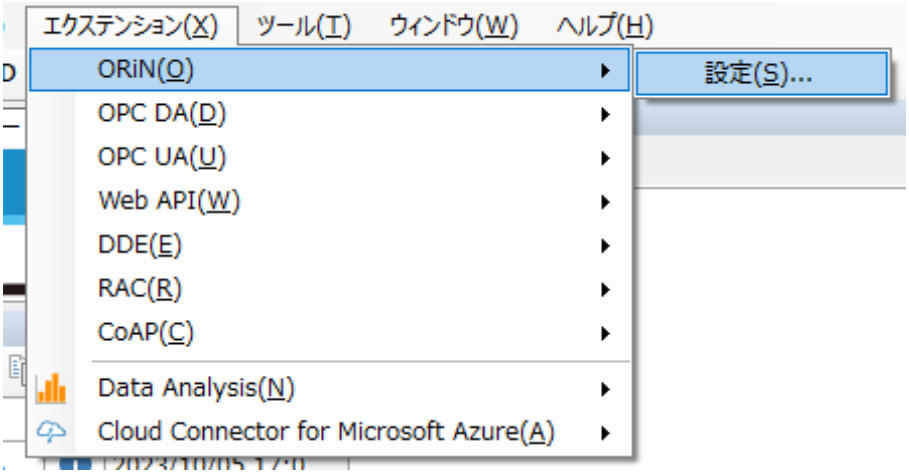
2.2. IoT Data Share/IoT Data Server での公開設定

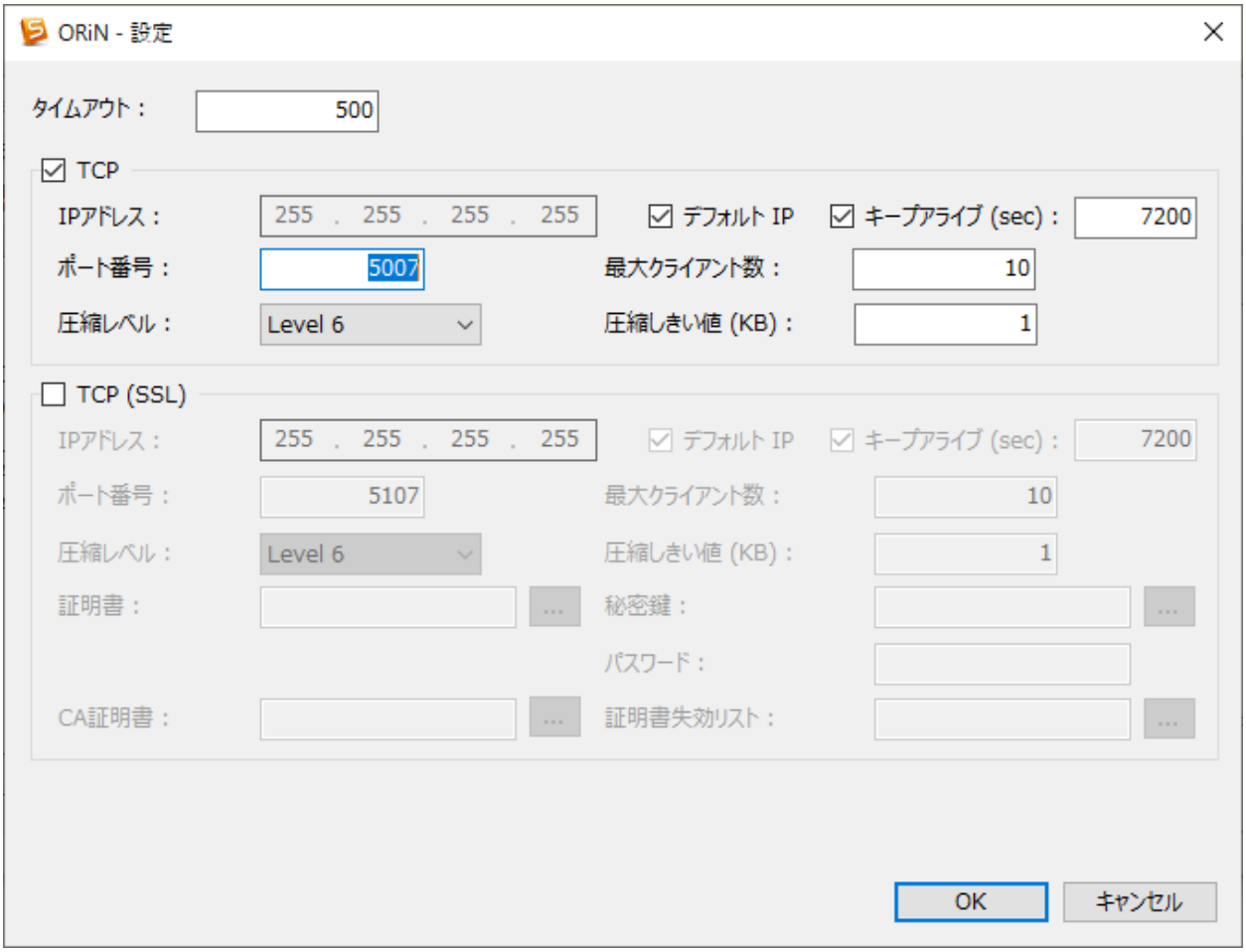
IoTDS プロバイダで， IoT Data Share，または IoT Data Server に 対してデータの書き込み/読み出しを行うためには IoT Data Share，または IoT Data Server での公開設定が必要です．

IoT Data Share/IoT Data Server のツリービューの Server/ORiN タブを選択し，データの書き込み/読み出しを行うコントローラおよびアイテムにチェックを入れることで IoTDS プロバイダからのデータの書き込み/読み出しが可能になります．



また，IoT Data Share/IoT Data Server が接続を待機するポート番号はメニューのエクステンション-ORiN-設定を選択すると表示される設定ダイアログで変更可能です．




ORiN - 設定
タイムアウト：
500
TCP
IPアドレス：
255 . 255 . 255 . 255
デフォルト IP
キープアライブ (sec)：
7200
ポート番号：
5007
最大クライアント数：
10
圧縮レベル：
Level 6
圧縮しきい値 (KB)：
1
TCP (SSL)
IPアドレス：
255 . 255 . 255 . 255
デフォルト IP
キープアライブ (sec)：
7200
ポート番号：
5107
最大クライアント数：
10
圧縮レベル：
Level 6
圧縮しきい値 (KB)：
1
証明書：
秘密鍵：
パスワード：
CA証明書：
証明書失効リスト：
OK
キャンセル

2.3. メソッド・プロパティ

2.3.1. CaoWorkspace::AddController メソッド

IoTDS プロバイダは Controller オブジェクト生成時に IoT Data Share 及び IoT Data Server に接続します. また, 生成された Controller オブジェクトは, プロジェクトのコントローラに対応されます.

書式 AddController(<bstrCtrlName:BSTR>,<bstrProvName:BSTR>,
<bstrPCName:BSTR>,<bstrOption:BSTR>))

bstrCtrlName : [in] コントローラ名
bstrProvName : [in] プロバイダ名. 固定値 = ” CaoProv.DENSO.IoTDS”
bstrPcName : [in] プロバイダの実行マシン名
bstrOption : [in] オプション文字列

以下にオプション文字列に指定するリストを示します.

表 2-2 CaoWorkspace::AddController のオプション文字列

<table>
<tr><th>オプション ([1])</th><th>説明</th></tr>
<tr><td>Controller
[=<プロジェクトのコントローラ名>]</td><td>接続先 IoT Data Share 及び IoT Data Server のプロジェクトに登録されているコントローラ名
（デフォルト: AddController の第一引数で指定したコントローラ名）
以下の予約名もコントローラ名として使用することができます.

<table>
<tr><th>コントローラ名</th><th>説明</th></tr>
<tr><td>$ENGINE$</td><td>CaoController::get_VariableNames で, IoT Data Share 及び IoT Data Server に登録されているコントローラ名の一覧を取得することができます.
このコントローラ名で CaoController オブジェクトを作成した場合, get_VariableNames, Execute 以外は実行できません.</td></tr>
<tr><td>$TREE_LIST$</td><td>CaoController::get_VariableNames で, IoT Data Share 及び IoT Data Server に登録されているコントローラ, アイテムの構造情報を取得することができます.
このコントローラ名で CaoController オブジェクトを作成した場合, get_VariableNames, Execute 以外は実行できません.</td></tr>
</table>
</td></tr>
</table>

[1] 角括弧(“[ ]”)内は省略可能を示します. また, 各パラメータの解説中の下線部はオプションを指定しなかったときのデフォルト値になります.

Server[=<IP アドレス> [:<ポート番号>]]	リモート接続する IoT Data Share または IoT Data Server の IP アドレスとポート番号を指定します. ポート番号は省略可能で, 省略した場合は, ポート番号として 5007 が使用されます. このオプションを省略した場合は, ローカル上の IoT Data Share に対して接続を行います.
MyIP[=<ローカルマシン IP アドレス>]	ローカルマシン上の IP アドレス. (複数 NIC 用途) (デフォルト:指定なし)
Timeout[=<タイムアウト時間>]	送受信時のタイムアウト時間. (ミリ秒) (デフォルト:5000)
UseCB[=TRUE/FALSE]	イベント受信用コールバックポートの使用の有無を指定します。 使用する場合のポート番号は CBPort オプションで指定します。 TRUE:コールバックポートを使用する。 FALSE:コールバックポートを使用しない (デフォルト:TRUE)
CBPort[=<コールバックポート番号>]	イベント受信用のゴールバックで使用するポート番号を指定します。 0 を指定した場合は動的にポート番号を割り当てます。 このオプションは UseCB オプションで TRUE を指定した場合に使用します。 (デフォルト:0)

```
AddController
(
    "<コントローラ名>",              // プロバイダのコントローラ名.
    " CaoProv.DENSO.IoTDS",         // プロバイダ名. (固定)
    "",
    "[[Controller=<コントローラ名>][,Server=<接続先 IoT Data Share の IP アドレス]"
       // 登録されているコントローラ名, 接続先 IoT Data Share の IP アドレス.
)
```

(例 1)

```
AddController
(
     "IoTDSCtrl1",                  // プロバイダのコントローラ名に IoTDSCtrl1 を使用
     "CaoProv.DENSO.IoTDS",         // プロバイダ名. (固定)
     "",
     "Controller=RC1"
       // 登録されているコントローラ RC1 を使用, ローカルマシンの IoT Data Share に接続
)
```

(例 2)

```
AddController
(
     "IoTDSCtrl1",                  // プロバイダのコントローラ名に IoTDSCtrl1 を使用
     "CaoProv.DENSO.IoTDS",         // プロバイダ名. (固定)
     "",
```

```
        "Controller=RC1,Server=192.168.0.1"
          // 登録されているコントローラ RC1 を使用，ネットワーク上(192.168.0.1)の IoT Data Share
に接続
    )
```

2.3.2. CaoController::AddVariable メソッド

IoTDS プロバイダでは Variable オブジェクトの生成時に IoT Data Share 及び IoT Data Server のコントローラに登録されているアイテムに対応されます．このため，AddVariable で指定する変数名は，指定コントローラのアイテム名を入力します．

```
AddVariable
(
    "<コントローラのアイテム名>"   // プロバイダの変数名 ＝ 指定コントローラのアイテム名
)
```

2.3.3. CaoController::Execute

このメソッドで実行される内容は，AddController 時に指定したコントローラ名によって異なります．各コントローラ名の毎の詳細な内容を以下に示します．

(1) $ENGINE$の場合

以下のコマンドを実行することができます．

コマンド名	引数	戻り値	概要
$CSQP_GET_COUNT$	None	プロジェクトに登録されているコントローラ数	IoT Data Share 及び IoT Data Sever のプロジェクトに登録されているコントローラ数を取得します

(2) $ENGINE$以外の場合

以下のコマンドを実行することができます．

コマンド名	引数	戻り値	概要
$CSQP_GET_COUNT$	なし	コントローラに登録されているアイテム数	IoT Data Share 及び IoT Data Sever の指定したコントローラに登録されているアイテム数を取得します
$CSQP_GET_STATE$	なし	コントローラのステータス状態 0:Uninit 1:Active 2:Inactive 3:Error 4:Terminated	IoT Data Share 及び IoT Data Sever の指定したコントローラのステータス状態を取得します

2.3.4. CaoController::get_Help プロパティ

コントローラの Description プロパティを取得します.

2.3.5. CaoController::get_ID プロパティ

コントローラの Index プロパティを取得します.

2.3.6. CaoController::get_Tag プロパティ

コントローラの Tag プロパティを取得します.

2.3.7. CaoController::put_Tag プロパティ

コントローラの Tag プロパティを設定します.

2.3.8. CaoController::get_VariableNames プロパティ

コントローラ名が“$TREE_LIST$”の場合は, IoT Data Share 及び IoT Data Server に登録されているコントローラ, アイテムの構造情報を取得することができます. このとき, 取得対象となる IoT Data Share プロジェクトは以下の順番で検出します.

- 実行中のプロジェクト
- IoT Data Server 又は IoT Data Share Manager によって登録されているデフォルトプロジェクト
- 最後に実行したプロジェクト

コントローラ名が“$ENGINE$”の場合は, IoT Data Share 及び IoT Data Server に登録されているコントローラの一覧を取得します.

それ以外のコントローラ名の場合は, AddController で指定したコントローラに登録されているアイテムの一覧を取得します.

2.3.9. CaoVariable::get_Attribute プロパティ

アイテムの属性値を取得します. 取得する属性値は以下の通りです.

表 2-3 取得可能な属性値

属性値	意味
0	Read/Write
1	ReadOnly
2	WriteOnly

2.3.10. CaoVariable::get_DateTime プロパティ

アイテムの更新日時を取得します.

2.3.11. CaoVariable::get_Help プロパティ

アイテムの Description プロパティを取得します.

2.3.12. CaoVariable::get_ID プロパティ

アイテムの ID プロパティの値を取得します.

アイテムの ID プロパティはアイテムが保持する CAO の Variable の ID プロパティ(サブアイテムの場合は親アイテムの保持する CAO の Variable の ID プロパティ)にアクセスします.

CAO の Variable の ID プロパティが実装されていないプロバイダの場合, ID プロパティにアクセスを行うと未実装エラー(0x80004001)が発生します.

CAO の Variable を持たないアイテム(仮想アイテム, エイリアスアイテム, 配列型アイテム)の ID プロパティにアクセスを行った場合は 0x80004005 のエラーが発生します.

2.3.13. CaoVariable::put_ID プロパティ

アイテムの ID プロパティの値を設定します.

2.3.14. CaoVariable::get_Microsecond プロパティ

アイテムの更新日時のマイクロ秒を取得します.

2.3.15. CaoVariable::get_Tag プロパティ

アイテムの Tag プロパティを取得します.

2.3.16. CaoVariable::put_Tag プロパティ

アイテムの Tag プロパティを設定します.

2.3.17. CaoVariable::get_Value プロパティ

アイテムの値を取得します.

2.3.18. CaoVariable::put_Value プロパティ

アイテムに値を設定します.

2.4. エラーコード

IoTDS プロバイダでは, 以下の固有エラーコードが定義されています. ORiN2 共通エラーについては,「ORiN2 プログラミングガイド」のエラーコードの章を参照してください.

表 2-4 独自エラーコード一覧

エラー名	エラー番号	説明
E_CAOP_NO_CAOSQL	0x80100000	プロジェクトが起動されていません.

3. サンプルプログラム

以下にローカルホストで動作している IoT Data Share の SampleController コントローラ下の SampleItem アイテムから値を取得するサンプルを示します.

List 3-1 Form1.vb

```
Imports ORiN2.ManagedCAO

Public Class Form1
    Private eng As CCaoEngine = Nothing
    Private ctrl As CCaoController = Nothing
    Private variable As CCaoVariable = Nothing

    Private Sub DisposeEngine()
        If Not IsNothing(eng) Then
            eng.Dispose()
            eng = Nothing
        End If
    End Sub

    Private Sub BtnConnect_Click(sender As Object, e As EventArgs) Handles BtnConnect.Click
        If Not IsNothing(eng) Then Return
        Try
            eng = New CCaoEngine()
            ctrl = eng.Workspaces(0).AddController( _
                "SampleController", _
                "CaoProv.DENSO.IoTDS", _
                "", _
                "Server=127.0.0.1")
            variable = ctrl.AddVariable("SampleItem", "")
        Catch ex As Exception
            MessageBox.Show(Me, ex.ToString(), Text, MessageBoxButtons.OK, MessageBoxIcon.Error)
            DisposeEngine()
        End Try
    End Sub

    Private Sub BtnGetValue_Click(sender As Object, e As EventArgs) Handles BtnGetValue.Click
```

```
        If IsNothing(variable) Then Return
        Dim value As Object = variable.Value
        MessageBox.Show(Me, String.Format("{0}({1})", value, _
                                If(IsNothing(value), "Nothing", value.GetType().Name)))
    End Sub

    Private Sub Form1_FormClosed(sender As Object, e As FormClosedEventArgs) _
        Handles MyBase.FormClosed
        DisposeEngine()
    End Sub
End Class
```