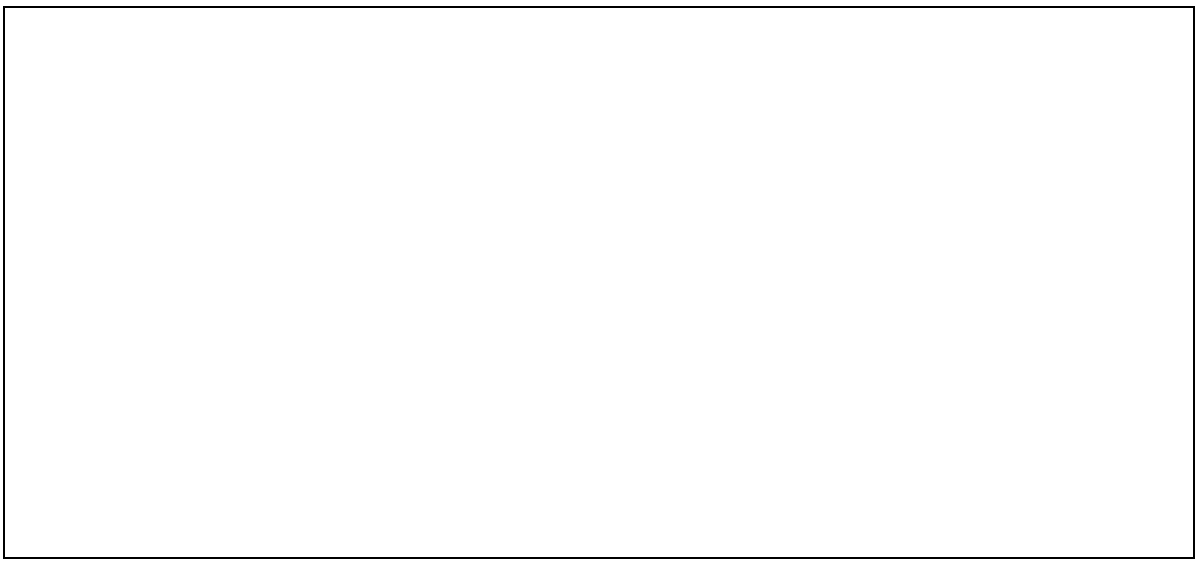
デンソー
RFID VOCS Scanner UR30 プロバイダ

Version 1.0.2

ユーザーズ ガイド

May 17, 2022



【改版履歴】

バージョン	日付	内容
1.0.0	2019-03-25	初版.
1.0.1	2019-10-23	ソースコードのリファクタリング
1.0.2	2022-05-17	StartInventory コマンドの引数として Empty が入った場合，省略扱いとするように修正

【動作確認機種】

機種	バージョン	注意事項
URCT-M31	Ver. 1.04	

目次

1. はじめに

本書は，RFID VOCS Scanner の UR30 シリーズから RFID タグの取得をするプロバイダのユーザーズガイドです．図 1-1 に本プロバイダとデバイスの全体構成図になります．以降本プロバイダを UR30 プロバイダ，RFID VOCS Scanner をスキャナと呼称します．UR30 プロバイダはスキャナとシリアル通信を行います．

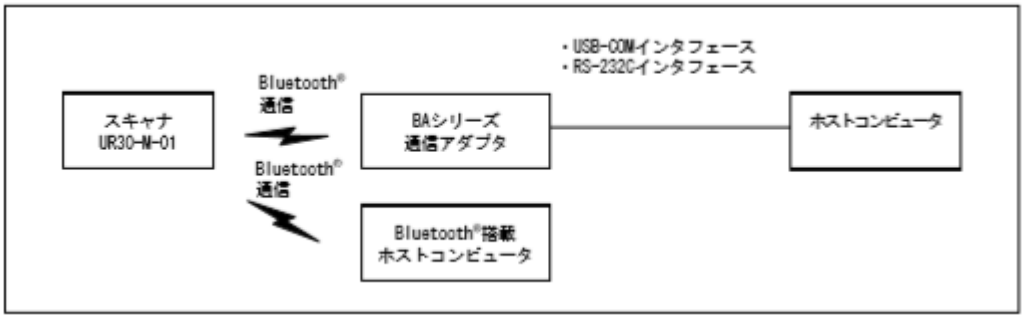


図 1-1 構成図

また，本プロバイダ及びデバイスそれぞれの対応を図 1-2に表します．
(※一例です．全てを表しているわけではありません．)

図 1-2 プロバイダの構成とデバイス情報との対応図

1.1. 本書が想定している環境とバージョン

クライアント PC が Windows 上で動作し，対象とするデバイスがシリアル接続可能なデバイス種類である環境を想定しています．PC の開発環境は，Component Object Model(COM，コンポーネント・オブジェクト・モデル)をサポートするプログラミング環境であれば開発が可能です．

1.2. 参考となる情報源

本書のプログラミング事例は，すべて Visual Basic for Applications で記載していますが，C++，Java，.NET などさまざまなプログラム言語で開発が可能です．使用方法に関しては，「ORiN2 プログラミングガイド」を参照していください．

「ORiN 2 プログラミングガイド」は ORiN2 SDK インストールフォルダの以下のファイルに該当します．

- ORiN2¥CAO¥Doc¥ORiN2_ProgrammersGuide_<*lang*>.pdf

※<lang>の部分は環境毎の言語文字列に置き換えてお読みください.

プロバイダを使ったアプリケーションを開発する上で必要となる ORiN2, COM/DCOM の基礎知識や技術に関して例を交えながら解説されています.

2. アプリケーション開発のための環境セットアップ

2.1. スキャナとクライアント PC との接続

スキャナと接続する際は，スキャナの電源を入っている状態でクライアント PC との接続を開始します．BA シリーズの通信アダプタを使用する場合は，お使いの BA シリーズの通信機器の取り扱い説明書をご参照してください．

2.2. PC 開発環境のセットアップ

2.2.1. UR30 プロバイダの自動インストール

ORiN2 SDK がインストールされている環境であれば，UR30 に接続するための動作環境（ラインタイム）の準備は完了です．

開発環境のセットアップは別途，Microsoft Visual Studio 6.0，2003/2005/2008/2010，LabVIEW など Component Object Model（COM，コンポーネント・オブジェクト・モデル）をサポートする，プログラミング環境をご準備してください．

2.2.2. UR30 プロバイダの手動インストール

UR30 プロバイダを手動でインストールする場合は下記レジストリ登録を行う必要があります．レジストリ登録を行う場合は，管理者権限でコマンドプロンプトを起動し，regsvr32 コマンドを実行してください．

また，CAO エンジンが動作するには予め，PC 毎に正規の ORiN2 SDK ライセンスが 1 つ登録されていなくてはなりません．ORiN2 SDK ユーザーズガイド内にある「ライセンスの追加と削除」の節を参照してください．

表 2-1 UR30 プロバイダ

ファイル名	CaoProvDENSOUR30.dll
ProgID	CaoProv.DENSO.UR30
レジストリ登録	regsvr32 CaoProvDENSOUR30.dll
レジストリ登録の抹消	regsvr32 /u CaoProvDENSOUR30.dll

3. コマンドリファレンス

3.1. メソッド/プロパティ一覧

表 3-1 メソッド/プロパティ一覧

カテゴリ	メソッド/プロパティ[1]		機能	参照
CaoWorkspace				
	AddController	M	コントローラに接続	P.9
CaoController				
	VariableNames	P	接続可能な変数名リストの取得	P.12
	Variables	P	コントローラが保持する変数コレクションの取得	P.12
	AddVariable	M	変数オブジェクトの追加	P.12
	Execute	M	拡張コマンドの実行	P.12
	OnMessage	E	メッセージ受信イベント	P.13
CaoVariable				
	Value	P	値の取得/設定	P.13

3.2. メソッド・プロパティ

3.2.1. CaoWorkspace クラス

3.2.1.1. AddController メソッド

CaoWorkspace に，コントローラオブジェクトを追加します．UR30 プロバイダでは，AddController メソッド実行時に渡されたパラメータを参照し，該当するスキャナと接続を行います．以下に，AddController メソッドの仕様を示します．

書式

```
CaoController AddController
(
        "<コントローラ名>",             // コントローラ名(任意)
        "CaoProv.DENSO.UR30",           // プロバイダ名（固定）
        "<マシン名>",                   // プロバイダ実行マシン名
        "<オプション>"                  // オプション文字列
)
```

オプション

[1] M:メソッド，P:プロパティ，E:イベントをそれぞれ示します．

以下にオプション文字列に指定するオプションを示します．オプション文字列は下記に示す各オプションをカンマ(,)でつなげた文字列となります．

オプション	必須	説明	値範囲
CONN=<通信パラメータ>	○	接続先情報を指定します．	COM
TriggerMode=	--	接続先のトリガモードを指定します． 0:オートオフ 1:モメンタリ 2:オルタネート 3:連続モード 1 4:連続モード 2	0-4 デフォルト値(3)
ResponseFormat=	--	接続先のレスポンスフォーマットを指定します． ResponseFormat=(a,b,c,d) a= PC 転送 (0:禁止,1:許可) b= RSSI 転送 (0:禁止,1:許可) c= 取得アンテナ情報転送 (0:禁止,1:許可) d= 偏波情報転送 (0:禁止,1:許可)	(0,0,0,0) -(1,1,1,1) デフォルト値(0,0,0,0)
Timaout=	--	通信タイムアウトを ms 単位で指定します．	0- デフォルト値(1000)

使用例

```
Dim engine As CaoEngine              ' Engineオブジェクト
Dim workspace As CaoWorkspace        ' Workspaceオブジェクト
Dim controller As CaoController      ' Controllerオブジェクト

Set engine = New CaoEngine
Set workspace = engine.Workspaces.Item(0)
Set controller = workspace.AddController("SampleController",_
                                  "CaoProv.DENSO.UR30",_
                                  "",_
                                  "conn=COM:1,TriggerMode=4,ResponseFormat=(0,1,1,0)")
```

3.2.1.1.1. CONN オプション

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

RS232C

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

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

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

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

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

<StopBits>　：ストップビット数. '1'-1bit, '2'-2bit

<Flow>　：フロー制御. 'O'-None, '1'-Xon/Xoff, '2'-ハードウェア制御 OR をとって指定できます.

3.2.1.1.2. TriggerMode オプション

以下にトリガモードの各モードについて示します.

オートオフ

トリガ押下してから最大で 5 秒間 RF タグ通信状態になります.
通信状態で 5 秒経過する前に RF タグ制御が完了することで通信待機状態になります.
また, 通信状態中にトリガを離すことでも通信待機状態となります.

モメンタリ

トリガを押下している間, RF タグ通信状態になります.
本モードは StartWrite, StartLock, StartKill が使用できません.

オルタネート

トリガを押下する毎に RF タグ通信状態と通信待機状態を交互に切り替えます.
本モードは StartWrite, StartLock, StartKill が使用できません.

連続モード 1

タグ機能制御開始コマンドの送信から, RF タグ制御停止コマンドを送られるまでの間, RF タグ通信状態になります.
本モードは StartWrite, StartLock, StartKill が使用できません.

連続モード 2

タグ機能制御開始コマンドの送信から, RF タグ制御が完了するまでの間, RF タグ通信状態になります.
RF タグ制御が完了すると自動的に通信待機状態となり, 再度通信状態にするには Execute コマンドの Stop(3.3.2.6)を送信後, タグ機能制御開始コマンドを送信してください.

3.2.2. CaoController クラス

3.2.2.1. VariableNames プロパティ

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

使用例

```
    ' ファイル名リスト取得
    Dim variables as Variant
    variables = controller.VariableNames
```

3.2.2.2. Variables プロパティ

コントローラが保持する，変数コレクションを取得します．

使用例

```
    ' 変数コレクション取得
    Dim variables as CaoVariables
    Set variables = controller.Variables

    ' 変数取得
    Dim variable as CaoVariable
    Set variable = variables.Item(0)
```

3.2.2.3. AddVariable メソッド

CaoController に変数オブジェクトを追加します．変数名には 3.4.1 に示すもののみ使用できます．
以下に，AddVariable の仕様を示します．

書式

```
CaoVariable AddVariable
(
        "<変数名>",             // 変数名
        "<オプション>"          // オプション文字列(省略可能)
)
```

3.2.2.4. Execute メソッド

ConController の拡張コマンドを実行します．以下に，Execute の仕様を示します．

書式

```
Variant Execute
(
        "<拡張コマンド名>",             // 拡張コマンド名
        "<オプション文字列>"            // オプション文字列(省略可能)
```

)

Execute で指定できる拡張コマンド一覧の詳細は 3.3 コマンド一覧を参照してください. 使用例は拡張コマンドの詳細で記述しています.

3.2.3. CaoVariable クラス

3.2.3.1. Value プロパティ

接続したスキャナからデータを取得/設定します. 変数名によって動作が異なります. 詳細は, 3.4. 変数一覧を参照してください.

3.2.4. OnMessage クラス

スキャナが RF タグ制御中に RF タグを検出した際に通知するデータをメッセージイベントとして発行します.

イベントの詳細は, 3.5 イベント一覧を参照してください.

またスキャナの RF タグ制御状態に関しては 3.3.2 を参照してください.

3.3. コマンド一覧

コマンドには,パラメータの設定取得などのパラメータ設定取得コマンドとスキャナを RF タグと通信可能状態にする RF タグ制御コマンドがあります.

コマンド	説明	参照
パラメータコマンド		
GetVersion	ファームウェアバージョンを取得します. “Ver. n.nn” n.nn=ファームウェアバージョン	P.15
GetSerial	シリアル番号を取得します. “ID. nnnnnn” nnnnnn=シリアル番号	P.16
GetSerialAndPartno	品番とシリアル番号を取得します. “ID. XXXXXXXXXnnnnnn” XXXXXXXXXX=品番 nnnnnn=シリアル番号	P.16
GetBatteryLevel	電池残量レベルを取得します.	P.16
SetTriggerMode	トリガモードを設定します.	P.17
SetReadTypeOutput	RF タグ読み取り時の出力を設定します.	P.17
GetReadTypeOutput	RF タグ読み取り時の出力を取得します.	P.18
SetWriteTypeOutput	RF タグ書き込み時の出力を設定します.	P.18
GetWriteTypeOutput	RF タグ書き込み時の出力を取得します.	P.18

コマンド	説明	参照
パラメータコマンド		
SetFrequency	中心周波数を設定します. 取得値はビットフィールドで表現され，複数のチャンネルを設定します.	P.19
GetFrequency	中心周波数を取得します. 設定値はビットフィールドで取得でき，複数のチャンネルを同時に設定できます.	P.19
SetQValue	RF タグとの通信を行った際に，RF タグからの応答が衝突する確率を調整するために使用する Q 値を設定します.	P.20
GetQValue	RF タグとの通信を行った際に，RF タグからの応答が衝突する確率を調整するために使用する Q 値を取得します.	P.20
SetSession	RF タグに対して再応答の条件を決定するセッションフラグを設定します.	P.20
GetSession	RF タグに対して再応答の条件を決定するセッションフラグを取得します.	P.21
SetSessionInit	セッションフラグの初期化有無を設定します.	P.21
GetSessionInit	セッションフラグの初期化有無を取得します.	P.22
SetRereadPrevention	RF タグに対する 2 度読み防止設定を設定します.	P.22
GetRereadPrevention	RF タグに対する 2 度読み防止設定を取得します.	P.22
SetVerify	RF タグに対して「Write」を行った後，自動的に「Read」を行い，RF タグへの書き込みが正しくできたかを確認する Verify を設定します.	P.23
GetVerify	RF タグに対して「Write」を行った後，自動的に「Read」を行い，RF タグへの書き込みが正しくできたかを確認する Verify を取得します.	P.23
SetPoewerSaving	RF 読み取り処理の省電力設定を設定します.	P.23
GetPoewerSaving	RF 読み取り処理の省電力設定を取得します.	P.24
SetPolarization	アンテナ偏波の垂直か水平のどちらかを設定します.	P.24
GetPolarization	アンテナ偏波の垂直か水平のどちらが設定されているかを取得します.	P.25
SaveParameter	設定したパラメータを UR30 のフラッシュメモリ	P.25

コマンド	説明	参照
パラメータコマンド		
	に保存します. ※UR30 の電源 on/off でも保存した内容を記憶します.	
SetResponseFormat	タグ制御開始コマンドで取得するデータの PC 転送，RSSI 転送，取得アンテナ情報転送，偏波情報転送を取得するか変更します.	P.25
RF タグ制御コマンド		
StartInventory	RF タグ制御開始コマンド. スキャナが検出した RF タグの UII データを読み込み通知します.	P.26
StartRead	RF タグ制御開始コマンド. スキャナが検出した RF タグの指定したメモリデータを読み出し通知します.	P.27
StartWrite	RF タグ制御開始系コマンド. スキャナが RF タグを検出した際に指定したメモリデータに書き込みを行います．書き込みを行った RF タグの情報を返却します.	P.28
StartLock	RF タグ制御開始コマンド. スキャナが RF タグを検出した際に指定した領域の Lock 状態を変更し．Lock 状態を変更したタグを通知します.	P.29
StartKill	RF タグ制御開始コマンド. スキャナが RF タグを検出した際に検出したタグをキル状態にします. キルしたタグの情報を通知します. キルが成功したタグは使用ができなくなるので実行する際は注意してください.	P.30
Stop	スキャナが RF タグ制御を実行中の場合，RF タグ制御を停止します.	P.30
ClearBuffer	2 度読み防止用のバッファをクリアします.	P.31

3.3.1. パラメータコマンド

パラメータコマンドは，スキャナ本体に関するパラメータの設定取得をします．コマンド実行後に AddController 時設定したタイムアウト時間までにスキャナから結果が返ってこなかった場合は，タイムアウトエラーが発生します.

3.3.1.1. GetVersion

UR30 のファームウェアバージョンを取得します.

項目	型説明

項目	型説明	
引数	なし	
戻り値	VT_BSTR	ファームウェアバージョンを"Ver. n.nn"形式で取得. n.nn = バージョン番号

使用例

```
' ファームウェアバージョンを取得
Dim version As String
version = controller.Execute("GetVersion")
```

3.3.1.2. GetSerial

UR30 のシリアル番号を取得します.

項目	型説明	
引数	なし	
戻り値	VT_BSTR	シリアル番号を"ID. nnnnnn"形式で取得. nnnnnn = バージョン番号

使用例

```
' シリアル番号を取得
Dim serial As String
serial = controller.Execute("GetSerial")
```

3.3.1.3. GetSerialAndPartno

UR30 の品番とシリアル番号を取得します.

項目	型説明	
引数	なし	
戻り値	VT_BSTR	品番とシリアル番号を"ID. XXXXXXXXXXnnnnnn"形式で取得 XXXXXXXXXX = 品番 nnnnnn = バージョン番号

使用例

```
' 品番とシリアル番号を取得
Dim serialAndPart As String
serialAndPart = controller.Execute("GetSerialAndPartno")
```

3.3.1.4. GetBatteryLevel

電池残量のレベルを取得します.

項目	型説明
引数	なし

項目	型説明	
戻り値	VT_I4	バッテリー残量を取得. 1: 残量 10%未満 2: 残量 10%以上～40%未満 3: 残量 40%以上

使用例

```
' バッテリー残量レベルを取得
Dim batteryLevel As Long
batteryLevel = controller.Execute("GetBatteryLevel")
```

3.3.1.5. SetTriggerMode

RF タグ通信のトリガモードを設定します.

RF タグ制御開始コマンド実施前に実施してください.RF タグ制御開始コマンド実施中に本コマンドを実行した場合エラーが返却されます.

デフォルトは連続モード 1 になっています.

項目	型説明	
引数	VT_I4	トリガモードを指定します. 0: オートオフ 1: モメンタリ 2: オルタネート 3: 連続モード 1 4: 連続モード 2 各モードの詳細は（P.11）を参照してください.
戻り値	なし	

使用例

```
' トリガモードを設定
Call controller.Execute("SetTriggerMode", 1)
```

3.3.1.6. SetReadTypeOutput

タグ読み取り時の出力を dBm×10 の単位で設定します. 設定した値は, 内部フラッシュメモリに保存（3.3.1.26 参照）しないと端末電源 OFF した際に初期化されます.

本コマンドは, タグ制御開始コマンド実施前に実施してください. タグ制御開始コマンド実施中に本コマンドを実行した場合エラーが返却されます.

項目	型説明	
引数	VT_I4	読み取り時の出力(dBm×10)を指定します. 値範囲：40-240
戻り値	なし	

使用例

```
' タグ読み取り時出力を設定
Call controller.Execute("SetReadTypeOutput", 100)
```

3.3.1.7. GetReadTypeOutput

タグ読み取り時の出力を dBm×10 の単位で取得します.

項目	型説明	
引数	なし	
戻り値	VT_I4	タグ読み取り時の出力を取得

使用例

```
' タグ読み取り時出力を取得
Dim readTypeOutput As Long
readTypeOutput = controller.Execute("GetReadTypeOutput")
```

3.3.1.8. SetWriteTypeOutput

タグ書き込み時の出力を dBm×10 の単位で設定します. 設定した値は, 内部フラッシュメモリに保存（3.3.1.26 参照）しないと端末電源 OFF した際に初期化されます.

本コマンドは, タグ制御開始コマンド実施前に実施してください. タグ制御開始コマンド実施中に本コマンドを実行した場合エラーが返却されます.

項目	型説明	
引数	VT_I4	読み取り時の出力(dBm×10)を指定します. 値範囲：40-240
戻り値	なし	

使用例

```
' タグ書き込み時出力を設定
Call controller.Execute("SetWriteTypeOutput", 100)
```

3.3.1.9. GetWriteTypeOutput

タグ書き込み時の出力を dBm×10 の単位で取得します.

項目	型説明	
引数	なし	
戻り値	VT_I4	タグ読み取り時の出力を取得

使用例

```
' タグ読み取り時出力を取得
Dim writeTypeOutput As Long
writeTypeOutput = controller.Execute("GetWriteTypeOutput")
```

3.3.1.10. SetFrequency

中心周波数(単位チャンネル番号)をビットフラグで設定します．設定した値は，内部フラッシュメモリに保存（3.3.1.26 参照）しないと端末電源 OFF した際に初期化されます．

本コマンドは，タグ制御開始コマンド実施前に実施してください．タグ制御開始コマンド実施中に本コマンドを実行した場合エラーが返却されます．

項目	型説明	
引数	VT_I4	許可する中心周波数を指定 値範囲：0x00000001 - 0x0007FFFF 0x00000001 : Ch5, 0x00000002 : Ch11, 0x00000004 : Ch17, 0x00000008 : Ch23, 0x00000010 : Ch24, 0x00000020 : Ch25, 0x00000040 : Ch26, 0x00000080 : Ch27, 0x00000100 : Ch28, 0x00000200 : Ch29, 0x00000400 : Ch30, 0x00000800 : Ch31, 0x00001000 : Ch32, 0x00002000 : Ch33, 0x00004000 : Ch34, 0x00008000 : Ch35, 0x00010000 : Ch36, 0x00020000 : Ch37, 0x00040000 : Ch38
戻り値	なし	

使用例

```
' タグ書き込み時出力を設定
' Ch5,11,24を指定する場合
Dim frequency As Long
frequency = &H1 + &H2 + &H10
Call controller.Execute("SetFrequency", frequency)
```

3.3.1.11. GetFrequency

中心周波数(単位チャンネル番号)をビットフラグで取得します．

項目	型説明	
引数	なし	
戻り値	VT_I4	中心周波数を取得

使用例

```
' 中心周波数を取得
Dim frequency As Long
frequency = controller.Execute("GetFrequency")
```

3.3.1.12. SetQValue

RF タグからの応答が衝突する確率を調整するために使用する Q 値を設定します．値が小さいほど衝突確率が増加し，値が大きいほど衝突確率が低減します．また値が大きいほど応答速度が劣化する可能性があります．設定した値は，内部フラッシュメモリに保存（3.3.1.26 参照）しないと端末電源 OFF した際に初期化されます．

本コマンドは，タグ制御開始コマンド実施前に実施してください．タグ制御開始コマンド実施中に本コマンドを実行した場合エラーが返却されます．

項目	型説明	
引数	VT_I4	Q 値を指定します． 値範囲：0-7
戻り値	なし	

使用例

```
' Q値の設定
Call controller.Execute("SetQValue", 4)
```

3.3.1.13. GetQValue

RF タグからの応答が衝突する確率を調整するために使用する Q 値を取得します．

項目	型説明	
引数	なし	
戻り値	VT_I4	中心周波数を取得

使用例

```
' Q値の取得
Dim qValue As Long
qValue = controller.Execute("GetQValue")
```

3.3.1.14. SetSession

RF タグに対して再応答の条件を決定するセッションフラグを設定します．設定した値は，内部フラッシュメモリに保存（3.3.1.26 参照）しないと端末電源 OFF した際に初期化されます．

本コマンドは，タグ制御開始コマンド実施前に実施してください．タグ制御開始コマンド実施中に本コマンドを実行した場合エラーが返却されます．

項目	型説明

項目	型説明	
引数	VT_I4	セッションフラグを指定します. 値範囲：0-3 0:RF タグ通信待機状態後，再度取得可能. 1:RF タグ検出後，規定時間以上[2]経過で再度同じタグの UII 取得可能 2:RF タグ通信待機状態後，規定時間以上[2]経過で再取得可能 3:2 と同じ
戻り値	なし	

使用例

```
' セッションフラグの設定
Call controller.Execute("SetSession", 1)
```

3.3.1.15. GetSession

RF タグに対して再応答の条件を決定するセッションフラグを取得します.

項目	型説明	
引数	なし	
戻り値	VT_I4	セッションフラグを取得します.

使用例

```
' セッションフラグを取得
Dim session As Long
session = controller.Execute("GetSession")
```

3.3.1.16. SetSessionInit

セッションフラグの初期化有無を設定します. RF タグ検出時に毎回セッションフラグの初期化を行うことで各セッションフラグの特徴にかかわらず，RF タグから UII を連続して読み取ることが可能になります. 設定した値は，内部フラッシュメモリに保存（3.3.1.26 参照）しないと端末電源 OFF した際に初期化されます.

本コマンドは，タグ制御開始コマンド実施前に実施してください. タグ制御開始コマンド実施中に本コマンドを実行した場合エラーが返却されます.

項目	型説明	
引数	VT_I4	セッションフラグの初期化有無を指定します. 値範囲：0-1 0: 初期化なし. 1: 初期化あり.
戻り値	なし	

[2] 規定時間は RF タグにより規定されています. ご使用になる RF タグの仕様をご確認ください.

使用例

```
' Sessionの設定
Call controller.Execute("SetSessionInit", 1)
```

3.3.1.17. GetSessionInit

セッションフラグの初期化有無を取得します.

項目	型説明	
引数	なし	
戻り値	VT_I4	セッションフラグの初期化有無を取得します.

使用例

```
' セッションフラグ初期化有無を取得
Dim sessionInit As Long
sessionInit = controller.Execute("GetSessionInit")
```

3.3.1.18. SetRereadPrevention

RF タグに対する 2 度読み防止設定を変更します. 設定した値は, 内部フラッシュメモリに保存（3.3.1.26 参照）しないと端末電源 OFF した際に初期化されます.

本コマンドは, タグ制御開始コマンド実施前に実施してください. タグ制御開始コマンド実施中に本コマンドを実行した場合エラーが返却されます.

項目	型説明	
引数	VT_I4	2 度読み防止の設定を指定します. 値範囲：0-2 0:2 度読み防止設定なし 1:RF タグ読み取り動作中の間, 2 度読み防止を継続 2:RF タグ通信モードの間, 2 度読み防止を継続
戻り値	なし	

使用例

```
'2度読み防止の設定
Call controller.Execute("SetRereadPrevention", 2)
```

3.3.1.19. GetRereadPrevention

RF タグに対する 2 度読み防止設定値を取得します.

項目	型説明	
引数	なし	
戻り値	VT_I4	2 度読み防止の設定を取得します.

使用例

```
'2度読み防止設定値を取得
```

```
Dim rereadPrevention As Integer
rereadPrevention = controller.Execute("GetRereadPrevention")
```

3.3.1.20. SetVerify

Write 処理時でのベリファイ有無を設定します.RF タグに対して「Write」を行った後,自動的に「Read」を行い, RF タグへの書き込みが正しくできたかの確認を行います. ベリファイを行う場合は, 行わない場合と比べ処理時間が長くなります. 設定した値は, 内部フラッシュメモリに保存（3.3.1.26 参照）しないと端末電源 OFF した際に初期化されます.

本コマンドは, タグ制御開始コマンド実施前に実施してください. タグ制御開始コマンド実施中に本コマンドを実行した場合エラーが返却されます.

項目	型説明	
引数	VT_I4	ベリファイ有無の設定を指定します. 値範囲：0-1 0:ベリファイなし 1:ベリファイあり
戻り値	なし	

使用例

```
'ベリファイ有無の設定
Call controller.Execute("SetVerify", 1)
```

3.3.1.21. GetVerify

ベリファイ有無の設定値を取得します.

項目	型説明	
引数	なし	
戻り値	VT_I4	ベリファイ有無を取得します.

使用例

```
'2度読み防止設定値を取得
Dim verify As Long
verify = controller.Execute("GetVerify")
```

3.3.1.22. SetPowerSaving

RF 読み取り処理時に省電力モードを許可するかを指定します. 設定した値は, 内部フラッシュメモリに保存（3.3.1.26 参照）しないと端末電源 OFF した際に初期化されます.

本コマンドは, タグ制御開始コマンド実施前に実施してください. タグ制御開始コマンド実施中に本コマンドを実行した場合エラーが返却されます.

項目	型説明

項目	型説明	
引数	VT_I4	省電力設定を指定します. 値範囲：0-1 0:省電力禁止 1:省電力許可
戻り値	なし	

使用例

```
'省電力の設定
Call controller.Execute("SetPowerSaving", 1)
```

3.3.1.23. GetPowerSaving

省電力設定を取得します.

項目	型説明	
引数	なし	
戻り値	VT_I4	省電力設定を取得します.

使用例

```
'省電力設定の取得
Dim powerSaving As Long
powerSaving = controller.Execute("GetPowerSaving")
```

3.3.1.24. SetPolarization

アンテナ偏波としてどの偏波を使用するか指定します. 設定した値は, 内部フラッシュメモリに保存（3.3.1.26 参照）しないと端末電源 OFF した際に初期化されます.

本コマンドは, タグ制御開始コマンド実施前に実施してください. タグ制御開始コマンド実施中に本コマンドを実行した場合エラーが返却されます.

項目	型説明	
引数	VT_I4	使用する偏波を指定します. 値範囲：1-3 1:垂直のみ 2:水平のみ 3:垂直, 水平両方
戻り値	なし	

使用例

```
'偏波の設定
Call controller.Execute("SetPolarization", 2)
```

3.3.1.25. GetPolarization

アンテナ偏波で使用している状態を取得します.

項目	型説明	
引数	なし	
戻り値	VT_I4	使用しているアンテナ偏波を取得

使用例

```
'アンテナ偏波の取得
Dim polarization As Long
polarization = controller.Execute("GetPolarization")
```

3.3.1.26. SaveParameter

設定した UR30 のパラメータをフラッシュメモリに保存します. 設定値を保存しなかった場合は, 電源 OFF した際に設定値が前回保存した状態に戻ります.

本コマンドは, タグ制御開始コマンド実施前に実施してください. タグ制御開始コマンド実施中に本コマンドを実行した場合エラーが返却されます.

項目	型説明
引数	なし
戻り値	なし

使用例

```
'パラメータをフラッシュメモリに保存
Call controller.Execute("SaveParameter")
```

3.3.1.27. SetResponseFormat

OnMessage イベント受信時に取得データの設定をします. 禁止に設定されると対応するデータは VT_EMPTY で返却されるようになります.

項目	型説明		
引数	VT_ARRAY | VT_VARIANT		
	1	VT_I4	PC の転送設定を指定します. 0:禁止 1:許可
	2	VT_I4	RSSI の転送設定を指定します. 0:禁止 1:許可
	3	VT_I4	取得アンテナ情報の転送設定を指定します. 0:禁止 1:許可

項目	型説明		
引数	VT_ARRAY | VT_VARIANT		
	4	VT_I4	偏波情報の転送設定を指定します. 0:禁止 1:許可
戻り値	なし		

使用例

```
'タグ通信時の返却データの転送許可設定をおこなう
Dim responseFormat As Variant
'PCとRSSの転送設定を許可する
responseFormat = Array(1, 1, 0, 0)

Call controller.Execute("SetResponseFormat", responseFormat)
```

3.3.2. RF タグ制御コマンド

RF タグ制御コマンドでは，コマンドを実行することでスキャナの通信制御を開始・停止します． 通信制御が開始されることでスキャナが RF タグを検出できるようになり，RF タグを検出した際は実行中の制御内容で RF タグと通信を行います．スキャナが通信制御中に RF タグを検出した場合は，検出した RF タグの情報を通知するため情報が通知されるごとにメッセージイベントが発行されます．

各コマンドの詳細は下記を参照してください．

3.3.2.1. StartInventory

スキャナが RF タグを検出した際に検出した UII データを通知するようになります．RF タグとスキャナとの通信はトリガモード設定（3.2.1.1.2 参照）に従ったトリガ動作により通信します．

スキャナから UII データが通知された際は，Inventory メッセージイベント（3.5.1 参照）を発行します．

RF タグ制御を開始している状態では本コマンド実行するとエラーが返却されます．

項目	型説明	
引数	VT_BSTR	省略可能です. UII のデータを指定します. (UII データ長に応じて設定，最大 62 バイト分の 16 進数文字列)
戻り値	なし	

使用例

```
'タグ制御をUII取得モードで開始
Dim UII As String
'指定するUIIのデータ
UII = "AAAA00001234"
Call controller.Execute("StartInventory", UII)
```

3.3.2.2. StartRead

スキャナが RF タグを検出した際に指定した範囲のメモリデータ内容を通知するようになります.RF タグとスキャナとの通信はトリガモード設定（3.2.1.1.2 参照）に従ったトリガ動作により通信します．スキャナから読み出しデータが通知された際は，Read メッセージイベント（3.5.2 参照）を発行します．スキャナが RF タグ制御を開始している場合は，実行できません.

項目	型説明		
引数	VT_ARRAY | VT_VARIANT		
	1	VT_I4	バンクを指定 値範囲:0-3 0:Reserved Bank 1:UII Bank 2:TID Bank 3:User Bank
	2	VT_I4	読み出し開始位置を指定(2 の倍数のみ指定可能) 値範囲[3]:0-65535
	3	VT_I4	読み出しサイズを指定(2 の倍数のみ指定可能) 値範囲 [3]:2-256
	4	VT_BSTR	アクセスパスワードを 16 進数文字列で指定（"00000000"指定時は認証処理を省略） 値範囲："00000000"-"FFFFFFFF"
	5	VT_BOOL	読み取りデータをバイト配列か文字列のどちらで受け取るかを指定します. TRUE=バイト配列 FALSE=文字列
	6	VT_BSTR	省略可能です. UII のデータを指定します. (UII データ長に応じて設定，最大 62 バイト分の 16 進数文字列)
戻り値	なし		

使用例

```
'タグ制御を読み込みモードで開始
Dim readParam As Variant
'読み込むメモリの場所を指定
readParam = Array(0, 4, 4, "00000000", FALSE)

Call controller.Execute("StartRead", readParam)
```

[3] 実際の指定できる範囲は，お使いの RF タグの仕様を確認してください.

3.3.2.3. StartWrite

スキャナが RF タグを検出した際に指定したメモリ位置に指定データの書き込みをします. RF タグとスキャナとの通信はトリガモード設定（3.2.1.1.2 参照）に従ったトリガ動作により通信します.

スキャナからは書き込みが実行された場合は，書き込みが実行された UII データが通知されます. スキャナから UII データが通知された場合 Write メッセージイベント(3.5.3 参照)が発行されます.

スキャナが RF タグ制御を開始している場合は，実行できません.

項目	型説明		
引数	VT_ARRAY | VT_VARIANT		
	1	VT_I4	バンクを指定 値範囲:0-3 0:Reserved Bank 1:UII Bank 2:TID Bank 3:User Bank
	2	VT_I4	書き込み開始位置を指定(2 の倍数のみ指定可能) 値範囲[3]：0-65535
	3	VT_BSTR	アクセスパスワードを 16 進数文字列で指定（"00000000"指定時は認証処理を省略） 値範囲："00000000"-"FFFFFFFF"
	4	VT_BSTR or VT_UI1 | VT_ARRAY	書き込みデータ VT_BSTR の場合，バイト配列を文字列で設定する 長さ：2 の倍数のバイト数 内容：16 進数文字列 例：（VT_UI1 | VT_ARRAY）の書き込みデータが「0x01,0xFF,0xA0」の場合 VT_BSTR では「”01FFA0”」で設定する
	5	VT_BSTR	省略可能です. UII のデータを指定します. (UII データ長に応じて設定，最大 62 バイト分の 16 進数文字列)
戻り値	なし		

使用例

```
'タグ制御を書き込みモードで開始
Dim writeParam As Variant
'アクセスパスワードに値を設定
writeParam = Array(0, 4, "00000000", "1234ABCD")
```

```
Call controller.Execute("StartWrite", writeParam)
```

3.3.2.4. StartLock

スキャナが RF タグを検出した際に指定した対象領域の Lock 状態を変更します. RF タグとスキャナとの通信はトリガモード設定（3.2.1.1.2 参照）に従ったトリガ動作により通信します.

スキャナからロックされた UII のデータが通知された際は,Lock メッセージイベント(3.5.4 参照)を発行します. スキャナが RF タグ制御を開始している場合は, 実行できません.

項目	型説明		
引数	VT_ARRAY | VT_VARIANT		
	1	VT_I4	ロック対象をビットフラグで指定（複数同時設定可能） 0x01:キルパスワード 0x02:アクセスパスワード 0x10: UII Bank 0x20: TID Bank 0x40: User Bank
	2	VT_I4	ロックタイプを指定 0x00:アンロック 0x01:ロック 0x10:永久アンロック 0x11:永久ロック
	3	VT_BSTR	アクセスパスワードを 16 進数文字列で指定（"00000000"指定時は認証処理を省略） 値範囲："00000000"-"FFFFFFFF"(HEX)
	4	VT_BSTR	省略可能です. UII のデータを指定します. (UII データ長に応じて設定, 最大 62 バイト分の 16 進数文字列)
戻り値	なし		

使用例

```
'タグ制御をロックモードで開始
Dim lockParam As Variant
'アクセスパスワードとキルパスワードをロックする
lockParam = Array(3, 1, "00000000")

Call controller.Execute("StartLock", lockParam)
```

3.3.2.5. StartKill

スキャナが RF タグを検出した際に検出した RF タグを KILL し，キルした RF タグの情報を通知します．KILL されたタグは今後読み込み書き込みができなくなるため注意してください．

RF タグとスキャナとの通信はトリガモード設定（3.2.1.1.2 参照）に従ったトリガ動作により通信します．

スキャナから RF タグの情報が通知された際は,Kill メッセージイベント(3.5.5 参照)を発行します．

スキャナが RF タグ制御を開始している場合は，実行できません．

項目	型説明		
引数	VT_ARRAY | VT_VARIANT		
	1	VT_BSTR	キルパスワードを 16 進数文字列で指定（"00000000"指定時は認証処理を省略） 値範囲："00000000"-"FFFFFFFF"(HEX)
	2	VT_BSTR	省略可能です． UII のデータを指定します． (UII データ長に応じて設定，最大 62 バイト分の 16 進数文字列で指定)
戻り値	なし		

使用例

```
'キルモードでRFタグ制御を開始
Dim killParam As Variant
'キルパスワードを入力
killParam = Array("1234ABCD")

Call controller.Execute("StartKill", killParam)
```

3.3.2.6. Stop

RF タグ制御動作を終了します．

項目	型説明
引数	なし
戻り値	なし

使用例

```
'タグ制御動作の開始
Call controller.Execute("StartInventory")

'タグ制御動作の停止
Call controller.Execute("Stop")
```

3.3.2.7. ClearBuffer

2 度読み防止用バッファをクリアします.

項目	型説明
引数	なし
戻り値	なし

使用例

```
'接続
Call Connect
'タグ制御動作の停止
Call controller.Execute("ClearBuffer")
```

3.4. 変数一覧

各クラスで使用可能な変数一覧を定義します. なお変数は, CaoVariable クラスのオブジェクトを指します.

3.4.1. CaoController クラス変数

変数名	説明	Value		参照
		get	put	
@MAKER_NAME	メーカー名を取得します.	○	-	P.32
@VERSION	ファームウェアバージョンを取得します.	○	-	P.32
@SERIAL	シリアル番号を取得します.	○	―	P.32
@SERIAL_AND_PARTNO	品番とシリアル番号を取得します.	○	-	P.33
@BATTERYLEVEL	電池残量レベルを取得します.	○	-	P.33
@TRIGGERMODE	トリガモードを設定します.	-	○	P.33
@READTYPEOUTPUT	Read 系出力の設定/取得をします.	○	○	P.34
@WRITETYPEOUTPUT	Write 系出力の設定/取得をします.	○	○	P.34
@FREQUENCY	周波数設定の設定/取得をします.	○	○	P.35
@QVALUE	Q 値の設定/取得をします.	○	○	P.35
@SESSION	Session の設定/取得をします.	○	○	P.35
@SESSIONINIT	SessionInit の設定/取得をします.	○	○	P.36
@REREADPREVENTION	2 度読み防止設定の設定/取得をします.	○	○	P.36
@VERIFY	Write 時ベリファイの設定/取得をします.	○	○	P.37
@POWERSAVING	省電力設定の設定/取得をします.	○	○	P.37
@POLARIZATION	偏波設定の設定/取得をします.	○	○	P.37
@RESPONSE_FORMAT	レスポンスフォーマット設定します.	-	○	P.38

3.4.1.1. @MAKER_NAME

メーカー名の取得をします.

データ型

型説明	
VT_BSTR	メーカー名を取得します.

使用例

```
    変数追加
Dim var As CaoVariable
Set var = controller.AddVariable("@MAKER_NAME")
' 値取得
Dim strVal As String
strVal = var.value
```

3.4.1.2. @VERSION

UR30 のファームウェアバージョンの取得をします.

取得できるデータに関しては, 3.3.1.1 を参照してください.

データ型

型説明	
VT_BSTR	UR30 のファームウェアバージョンを取得します.

使用例

```
' 変数追加
Dim var As CaoVariable
Set var = controller.AddVariable("@VERSION")
' 値取得
Dim value As String
value = var.value
```

3.4.1.3. @SERIAL

UR30 のシリアル番号の取得をします.

取得できるデータに関しては, 3.3.1.2 してください.

データ型

型説明	
VT_BSTR	UR30 のシリアル番号を取得します.

使用例

```
' 変数追加
```

```
    Dim var As CaoVariable
    Set var = controller.AddVariable("@SERIAL")
    ' 値取得
    Dim value As String
    value = var.value
```

3.4.1.4. @SERIAL_AND_PARTNO

UR30 の品番とシリアル番号の取得をします.

取得できるデータに関しては, 3.3.1.3 を参照してください.

データ型

型説明	
VT_BSTR	UR30 の品番とシリアル番号を取得します.

使用例

```
    ' 変数追加
    Dim var As CaoVariable
    Set var = controller.AddVariable("@SERIAL_AND_PARTNO")
    ' 値取得
    Dim value As String
    value = var.value
```

3.4.1.5. @BATTERYLEVEL

UR30 のバッテリー残量レベルの取得をします.

取得できるデータに関しては, 3.3.1.4 を参照してください.

データ型

型説明	
VT_I4	UR30 のバッテリー残量レベルを取得します.

使用例

```
    ' 変数追加
    Dim var As CaoVariable
    Set var = controller.AddVariable("@BATTERYLEVEL")
    ' 値取得
    Dim value As Long
    value = var.value
```

3.4.1.6. @TRIGGERMODE

トリガモードを設定します.

トリガーモードの設定に関しては, 3.3.1.5 を参照してください.

データ型

型説明	
VT_I4	トリガモードを設定します.

使用例

```
' 変数追加
Dim var As CaoVariable
Set var = controller.AddVariable("@TRIGGERMODE")
' 値設定
var.value = 1
```

3.4.1.7. @READTYPEOUTPUT

Read 系出力の設定/取得します.

設定に関しては 3.3.1.6 を参照し，取得に関しては 3.3.1.7 を参照してください.

データ型

型説明	
VT_I4	Read 系出力の設定/取得します.

使用例

```
' 変数追加
Dim var As CaoVariable
Set var = controller.AddVariable("@READTYPEOUTPUT")
' 値取得
Dim value As Long
value = var.value
' 値設定
var.value = 1
```

3.4.1.8. @WRITETYPEOUTPUT

Write 系出力の設定/取得します.

設定に関しては 3.3.1.8 を参照し，取得に関しては 3.3.1.9 を参照してください.

データ型

型説明	
VT_I4	Write 系出力の設定/取得します.

使用例

```
' 変数追加
Dim var As CaoVariable
Set var = controller.AddVariable("@WRITETYPEOUTPUT")
' 値取得
Dim value As Long
value = var.value
```

```
    ' 値設定
    var.value = 1
```

3.4.1.9. @FREQUENCY

周波数設定の設定/取得します.

設定に関しては 3.3.1.10 を参照し，取得に関しては 3.3.1.11 を参照してください.

データ型

型説明	
VT_I4	周波数設定の設定/取得します.

使用例

```
    ' 変数追加
    Dim var As CaoVariable
    Set var = controller.AddVariable("@FREQUENCY")
    ' 値取得
    Dim value As Long
    value = var.value
    ' 値設定
    var.value = 1
```

3.4.1.10. @QVALUE

Q 値の設定/取得します.

設定に関しては 3.3.1.12 を参照し，取得に関しては 3.3.1.13 を参照してください.

データ型

型説明	
VT_I4	Q 値の設定/取得します.

使用例

```
    ' 変数追加
    Dim var As CaoVariable
    Set var = controller.AddVariable("@QVALUE")
    ' 値取得
    Dim value As Long
    value = var.value
    ' 値設定
    var.value = 1
```

3.4.1.11. @SESSION

セッションフラグの設定/取得をします.

設定に関しては 3.3.1.14 を参照し，取得に関しては 3.3.1.15 を参照してください.

データ型

型説明	
VT_I4	セッションフラグの設定/取得をします.

使用例

```
' 変数追加
Dim var As CaoVariable
Set var = controller.AddVariable("@SESSION")
' 値取得
Dim value As Long
value = var.value
' 値設定
var.value = 1
```

3.4.1.12. @SESSIONINIT

セッションフラグの初期化設定の設定/取得をします.

設定に関しては 3.3.1.16 を参照し，取得に関しては 3.3.1.17 を参照してください.

データ型

型説明	
VT_I4	セッションフラグの初期化設定の設定/取得をします.

使用例

```
' 変数追加
Dim var As CaoVariable
Set var = controller.AddVariable("@SESSIONINIT")
' 値取得
Dim value As Long
value = var.value
' 値設定
var.value = 1
```

3.4.1.13. @REREADPREVENTION

2 度読み防止設定の設定/取得をします.

設定に関しては 3.3.1.18 を参照し，取得に関しては 3.3.1.19 を参照してください.

データ型

型説明	
VT_I4	2 度読み防止設定の設定/取得をします.

使用例

```
' 変数追加
Dim var As CaoVariable
Set var = controller.AddVariable("@REREADPREVENTION")
' 値取得
```

```
    Dim value As Long
    value = var.value
    ' 値設定
    var.value = 1
```

3.4.1.14. @VERIFY

Write 時のベリファイ設定の設定/取得をします.

設定に関しては 3.3.1.20 を参照し，取得に関しては 3.3.1.21 を参照してください.

データ型

型説明	
VT_I4	Write 時のベリファイ設定の設定/取得をします.

使用例

```
    ' 変数追加
    Dim var As CaoVariable
    Set var = controller.AddVariable("@VERIFY")
    ' 値取得
    Dim value As Long
    value = var.value
    ' 値設定
    var.value = 1
```

3.4.1.15. @POWERSAVING

省電力設定の設定/取得をします.

設定に関しては 3.3.1.22 を参照し，取得に関しては 3.3.1.23 を参照してください.

データ型

型説明	
VT_I4	省電力設定の設定/取得をします.

使用例

```
    ' 変数追加
    Dim var As CaoVariable
    Set var = controller.AddVariable("@POWERSAVING")
    ' 値取得
    Dim value As Long
    value = var.value
    ' 値設定
    var.value = 1
```

3.4.1.16. @POLARIZATION

偏波設定の設定/取得をします.

設定に関しては 3.3.1.24 を参照し，取得に関しては 3.3.1.25 を参照してください.

データ型

型説明	
VT_I4	偏波設定の設定/取得をします.

使用例

```
' 変数追加
Dim var As CaoVariable
Set var = controller.AddVariable("@POLARIZATION")
' 値取得
Dim value As Long
value = var.value
' 値設定
var.value = 1
```

3.4.1.17. @RESPONSE_FORMAT

レスポンスフォーマットの設定をします.

レスポンスフォーマットの設定に関しては, 3.3.1.27 を参照してください.

データ型

型説明	
VT_UI4|VT_ARRAY	レスポンスフォーマットの設定をします.

使用例

```
' 変数追加
Dim var As CaoVariable
Set var = controller.AddVariable("@RESPONSE_FORMAT")
' 値設定
var.value = ARRAY(1, 1, 1, 1)
```

3.5. イベント一覧

スキャナが検出した RF タグの情報を通知します. メッセージは 1 つのタグに対して 1 回発行され, 複数のタグが読み込まれた場合は読み込まれた回数分のメッセージを通知します.

メッセージ内容は RF タグ制御コマンドごとに異なり, 詳細は下記を参照してください.

メッセージ番号	説明	参照
41	StartInventory による RF タグ制御実施中に検出された RF タグの情報を通知します.	P. 39
42	StartRead による RF タグ制御中に検出された RF タグの情報と指定メモリ領域の読み込みデータを通知します.	P. 39
43	StartWrite による RF タグ制御実施中に検出された RF タグの情報を通知します.	P. 40

44	StartLock による RF タグ制御実施中に検出された RF タグの情報を通知します.	P. 41
45	StartKill による RF タグ制御実施中に検出された RF タグの情報を通知します.	P. 42

3.5.1. Inventory メッセージ

スキャナが StartInventory による RF タグ制御中(3.3.2.1 参照)に検出した RF タグの情報を通知します.

項目	説明		
メッセージ番号	41		
メッセージ内容	VT_ARRAY | VT_VARIANT		
	1	VT_BSTR	検出した RF タグの UII データ
	2	VT_BSTR or VT_EMPTY	検出した RF タグの PC データ[4]
	3	VT_BSTR or VT_EMPTY	RF タグ検出時の RSSI[4]
	4	VT_BSTR or VT_EMPTY	RF タグ検出時に使用したアンテナ番号[4]
	5	VT_BSTR or VT_EMPTY	RF タグ検出時の偏波状態[4]

3.5.2. Read メッセージ

スキャナが StartRead による RF タグ制御中 (3.3.2.2 参照)に検出した RF タグの情報を通知します. RF タグ通信結果が正常終了の場合は指定したメモリ領域も一緒に通知します.

項目	説明		
メッセージ番号	42		
メッセージ内容	VT_ARRAY | VT_VARIANT		
	1	VT_BSTR	検出した RF タグの UII データ

[4] レスポンスフォーマット設定で許可の時のみデータを返却, 禁止の場合は VT_EMPTY を返します

項目	説明		
メッセージ番号	42		
メッセージ内容	VT_ARRAY | VT_VARIANT		
	2	VT_BSTR or VT_EMPTY	検出したRFタグのPCデータ[4]
	3	VT_BSTR or VT_EMPTY	RFタグ検出時のRSSI[4]
	4	VT_BSTR or VT_EMPTY	RFタグ検出時に使用したアンテナ番号[4]
	5	VT_BSTR or VT_EMPTY	RFタグ検出時の偏波状態[4]
	6	VT_BSTR	RFタグ通信結果 "0000":正常終了 "0203":メモリ範囲外へのアクセス "0204":ロックメモリへのアクセス "020F":その他のエラー
	7	VT_BSTR or VT_UI1 | VT_ARRAY	読み出しデータ バイト配列か文字列かは，StartRead 時のパラメータで指定した値によって変化します.

3.5.3. Writeメッセージ

スキャナがStartWriteによるRFタグ制御中(3.3.2.3参照)に検出したRFタグ情報を通知します. RFタグ通信結果が正常終了の場合は，指定のメモリ領域に書き込みがされます.

項目	説明		
メッセージ番号	43		
メッセージ内容	VT_ARRAY | VT_VARIANT		
	1	VT_BSTR	検出したRFタグのUIIデータ
	2	VT_BSTR or VT_EMPTY	検出したRFタグのPCデータ[4]

項目	説明		
メッセージ番号	43		
メッセージ内容	VT_ARRAY | VT_VARIANT		
	3	VT_BSTR or VT_EMPTY	RF タグ検出時の RSSI[4]
	4	VT_BSTR or VT_EMPTY	RF タグ検出時に使用したアンテナ番号[4]
	5	VT_BSTR or VT_EMPTY	RF タグ検出時の偏波状態[4]
	6	VT_BSTR	RF タグ通信結果 "0000":正常終了 "0203":メモリ範囲外へのアクセス "0204":ロックメモリへのアクセス "020F":その他のエラー

3.5.4. Lock メッセージ

スキャナが StartLock による RF タグ制御中(3.3.2.4 参照)に検出した RF タグの情報を通知します. RF タグ通信結果が正常終了の場合は, 検出したタグに指定のメモリ領域のロック状態が変更されます.

項目	説明		
メッセージ番号	44		
メッセージ内容	VT_ARRAY | VT_VARIANT		
	1	VT_BSTR	検出した RF タグの UII データ
	2	VT_BSTR or VT_EMPTY	検出した RF タグの PC データ[4]
	3	VT_BSTR or VT_EMPTY	RF タグ検出時の RSSI[4]
	4	VT_BSTR or VT_EMPTY	RF タグ検出時に使用したアンテナ番号[4]

項目	説明		
メッセージ番号	44		
メッセージ内容	VT_ARRAY | VT_VARIANT		
	5	VT_BSTR or VT_EMPTY	RF タグ検出時の偏波状態 [4]
	6	VT_BSTR	RF タグ通信結果 "0000":正常終了 "0203":メモリ範囲外へのアクセス "0204":ロックメモリへのアクセス "020F":その他のエラー

3.5.5. Kill メッセージ

スキャナが StartKill による RF タグ制御中(3.3.2.5)に検出した RF タグを通知します. RF タグ通信結果が正常終了の場合は, 検出したタグは Kill 状態になり使用ができなくなります.

項目	説明		
メッセージ番号	45		
メッセージ内容	VT_ARRAY | VT_VARIANT		
	1	VT_BSTR	検出した RF タグの UII データ
	2	VT_BSTR or VT_EMPTY	検出した RF タグの PC データ [4]
	3	VT_BSTR or VT_EMPTY	RF タグ検出時の RSSI[4]
	4	VT_BSTR or VT_EMPTY	RF タグ検出時に使用したアンテナ番号 [4]
	5	VT_BSTR or VT_EMPTY	RF タグ検出時の偏波状態 [4]

項目	説明		
メッセージ番号	45		
メッセージ内容	VT_ARRAY | VT_VARIANT		
	6	VT_BSTR	RF タグ通信結果 "0000":正常終了 "0203":メモリ範囲外へのアクセス "0204":ロックメモリへのアクセス "020F":その他のエラー

4. UR30 プロバイダによるプログラミング

UR30 プロバイダでは，以下の手順でクライアント PC とスキャナを接続することができます.

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

スキャナに接続した後は，CaoController の Execute メソッドを使用する，もしくは，CaoVariabe オブジェクトを生成することで，スキャナの情報にアクセスすることができます.

4.1. スキャナに接続し RF タグの UII データを読み込むサンプルプログラミング

ここでは例として UR30 のパラメータを設定しフラッシュメモリに保存するサンプルプログラムを示します. 表 4-1 にサンプルプログラムの要件を，にサンプルプログラムの流れをそれぞれ記述しています.

表 4-1 サンプルプログラムの要件

要件	説明
接続先	COM
	接続先 COM 番号は 3，その他項目は省略
処理内容	スキャナに連続モード 2,PC データ転送許可，RSSI データ転送許可で接続する
	スキャナを Inventory モードで通信制御を開始する
	スキャナから UII データの読み込みが走るとメッセージを表示する
	メッセージ表示後切断する

4.1.1. サンプルプログラム

以下にサンプルプログラムの全体像を示します.

Sample　SampleInventory.vb

```
' オブジェクト
Dim engine As CaoEngine
Dim workspace As CaoWorkspace
Dim WithEvents controller As CaoController
Dim MessageBuf As Variable

' 接続メソッド
Private Sub Connect()
    ' CaoEngine オブジェクトの生成
    Set engine = New CaoEngine
    ' CaoWorkspace オブジェクトの生成
    Set workspace = engine.AddWorkspace("NewWrks", "")
    ' CaoController オブジェクトの生成
```

```
    Set controller = workspace.AddController("SampleController", _
                                             "CaoProv.DENSO.UR30", _
                                             "", _
                                             "COM:3, ResponseFormat=(1, 1, 0, 0)")
End Sub

' 切断メソッド
Private Sub Disconnect()
    ' CaoWorkspace から CaoController を削除
    Call workspace.Controllers.Remove(controller.Index)
    ' CaoController の消去
    Set controller = Nothing

    ' CaoEngine から CaoWorkspace を削除
    Call engine.Workspaces.Remove(workspace.Index)
    ' CaoWorkspace の消去
    Set workspace = Nothing

    ' CaoEngine の消去
    Set engine = Nothing
End Sub

' Inventory モードの開始
Private Sub StartInventory()
    ' 接続
    Call Connected
    ' Inventory モードの開始
    controller.Execute("StartInventory")

End Sub

' プロバイダからのメッセージイベントの通知
Private Sub controller_OnMessage(ByVal pICaoMess As CAOLib.CaoMessage)
    MessageBuf = pICaoMess.Value
    'UII データ
    Dim uiiData As String
    uiiData = MessageBuf(0)
   ' PC データ
    Dim pc As String
    pc = MessageBuf(1)
    'RSSI データ
    Dim rssi As String
    rssi = MessageBuf(2)
    '使用アンテナ
    Dim antenna As String
    antenna = MessageBuf(3)
    '偏波状況
    Dim polarization As String
```

```
    polarization = MessageBuf(4)

    'UII データをメッセージボックスで表示
    MsgBox uiiData

    Call Dicconnect

End Sub
```

4.1.1.1. 接続

スキャナと接続するためには，以下の手順を取ります.

(1) オブジェクトを保持するための変数を用意します．コントローラ接続に必要なオブジェクトは，CaoEngineオブジェクトとCaoWorkspaceオブジェクトとCaoControllerオブジェクトです.
CaoWorkpaceオブジェクトは，CaoControllerオブジェクトをCaoWorkspacesから取得する場合には変数を用意する必要はありません．またWithEventsを変数名の前につけることでCaoControllerからOnMessageイベントを受信することができるようになります.

```
Dim engine As CaoEngine                 ' CaoEngineオブジェクト用の変数
Dim workspace As CaoWorkspace           ' CaoWorkspaceオブジェクト用の変数
Dim WithEvents controller As CaoController  ' CaoControllerオブジェクト用の変数
```

(2) CaoEngineオブジェクトを生成します．CaoEngineオブジェクトはNewキーワードを使って生成します.

```
' CaoEngine オブジェクトの生成
Set engine = New CaoEngine
```

(3) CaoWorkspaceオブジェクトを取得もしくは生成します．CaoEngineオブジェクトを生成すると，デフォルトでCaoWorkspacesオブジェクトとCaoWorkspaceオブジェクトを1つずつ生成しています．以下にCaoWorkspaceオブジェクトを新しく生成するコード例とデフォルトのCaoWorkspaceを示します.

```
' CaoWorkspace オブジェクトの生成
Set workspace = engine.AddWorkspace("NewWrks", "")
```

(4) CaoControllerオブジェクトを生成します．CaoControllerオブジェクトを生成するには，使用するプロバイダ名と使用するためのパラメータを設定します．UR30プロバイダでは，スキャナと接続した際にあらかじめレスポンスフォーマットをオプションで指定します．以下にコード例を示します.

```
' CaoController オブジェクトの生成
Set controller = workspace.AddController("SampleController", _
                                         "CaoProv.DENSO.UR30", _
                                         "", _
                                         "conn=COM:3 ,ResponseFormat=(1, 1, 0, 0)")
```

4.1.1.2. UR30 を UII データの取得モードで実行

UR30 を UII 取得モード（Inventory）で開始します.

```
' Inventory モードの開始
controller.Execute("StartInventory")
```

4.1.1.3. タグ読み込み時のメッセージを受信

UR30 からタグの情報を受信したら OnMessage イベントが発行されます．OnMessage イベントを受信したらメッセージ内容から UII データを MsgBox で表示します．

```
Private Sub controller_OnMessage(ByVal pICaoMess As CAOLib.CaoMessage)
    MessageBuf = pICaoMess.Value

    ' 受信データに格納されている UII データをメッセージボックスで表示
    MsgBox MessageBuf(0)

    Call Dicconnect

End Sub
```

4.1.1.4. 切断

コントローラと切断する場合には，生成したオブジェクトを消去すると共に，オブジェクトを管理するコレクションクラスから消去するオブジェクトを削除します．以下にコード例を示します．

```
    ' CaoWorkspace から CaoController を削除
    Call workspace.Controllers.Remove(controller.Index)
    ' CaoController の消去
    Set controller = Nothing
    ' CaoEngine から CaoWorkspace を削除
    Call engine.Workspaces.Remove(workspace.Index)
    ' CaoWorkspace の消去
    Set workspace = Nothing
    ' CaoEngine の消去
    Set engine = Nothing
```

5. UR30 プロバイダエラーコード

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

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

表 5-1 独自エラーコード表

エラー番号	説明
0x80110002	想定外のデータを受信 通信時にデータが破損した可能性があります．通信環境の見直しをしてください．
0x80110003	トリガモード指定エラー TriggerModeの指定が間違っています．
0x80110004	レスポンスフォーマットエラー プロバイダ側で保持しているレスポンスフォーマットの指定とUR30のレスポンスフォーマットの設定内容が違っています． SetResponseFormatを再度実行してください．
0x8010A300	動作モードエラー
0x8010A301	実行条件エラー 別の制御コマンド（StartInventory等）が実行中の可能性があります．一度Stopコマンドを実行してから再度制御コマンドを実行してください．
0x8010A303	フォーマットエラー 指定しているパラメータのフォーマットが違う可能性があります．16進数文字列で指定する項目の文字数が奇数になっていないか確認してください．
0x8010A304	引数範囲外エラー 指定したパラメータが範囲外です．
0x8010AA00	充電中エラー 充電中です．充電器から外して実行してください．
0x8010AA01	トリガモードエラー トリガモードを見直してください． StartWrite，StartLock，StartKillは，トリガモード"オートオフ"または"連続モード2"のときに実行可能です．
0x8010****	想定外の異常

また，本プロバイダは，UR30 のエラーコードを「0x8010****」でマスクして返します．

付録A. 通信プロトコルコマンド対応表

CaoWorkspace

メソッド	通信コマンド
AddController	RFUS
	RFUU
	RFURF2
Controllers::Remove	RFUS

CaoController::Execute

コマンド	通信コマンド
GetVersion	VER
GetSirial	ID
GetSirialAndPartno	IDF
GetBatteryLevel	VBAT
SetTriggerMode	RFUU
SetReadTypeOutput	RFUPS
GetReadTypeOutput	RFUPG
SetWriteTypeOutput	RFUPS
GetWriteTypeOutput	RFUPG
SetFrequency	RFUPS
GetFrequency	RFUPG
SetQValue	RFUPS
GetQValue	RFUPG
SetSession	RFUPS
GetSession	RFUPG
SetSessionInit	RFUPS
GetSessionInit	RFUPG
SetRereadPrevention	RFUPS
GetRereadPrevention	RFUPG
SetVerify	RFUPS
GetVerify	RFUPG
SetPoewerSaving	RFUPS

GetPoewerSaving	RFUPG
SetPolarization	RFUPS
GetPolarization	RFUPG
SaveParameter	RFUPW
StartInventory	RFUI
	RFUIU
StartRead	RFUR
	RFURU
StartWrite	RFUW
	RFUWU
StartLock	RFUL
	RFULU
StartKill	RFUK
	RFUKU
Stop	RFUS
ClearBuffer	RFUC
SetResponseFormat	RFURF2