Digital
Human Machine Interface

Pro-face®

データシェアリングAPI

ユーザーズマニュアル

はじめに

このたびは、(株)デジタル製Pro-Designerをお買い上げいただき、誠にありがとうございます。
この製品を正しくご使用いただくために、本書をよくお読みください。

お断り

(1)「Pro-Designer」(以下本製品といいます)のプログラムおよびマニュアル類は、すべて(株)デジタルの著作物であり、(株)デジタルがユーザーに対し「ソフトウェア使用条件」に記載の使用権を許諾したものです。当該「ソフトウェア使用条件」に反する行為は、日本国内外の法令により禁止されています。
(2) 本書の内容については万全を期して作成しておりますが、万一お気づきの点がありましたら、(株)デジタル営業担当窓口までご連絡ください。
(3) 前項にかかわらず、本製品を使用したことによるお客様の損害、および逸失利益、または第三者からのいかなる請求につきましても、当社はその責任を負いかねますので、あらかじめご了承ください。
(4) 製品の改良のため、本書の記述と本製品のソフトウェアとの間に異なった部分が生じることがあります。最新の説明は、別冊ないし電子的な情報として提供していますので、あわせてご参照ください。
(5) 本書は、(株)デジタルから日本国内仕様として発売された製品専用です。
(6) 本製品が記録・表示する情報の中に、(株)デジタルまたは第三者が権利を有する無体財産権、知的所有権に関わる内容を含むことがありますが、これは(株)デジタルがこれらの権利の利用について、ユーザーまたはその他の第三者に、何らの保証や許諾を与えるものではありません。

商標・商号の権利については「商標権などについて」を参照してください。

マニュアル表記について

このマニュアルでは、製品を正しく安全に使用するための重大な注意事項について説明しています。
製品使用前にこの注意事項を読み、製品を正しく使用してください。

絵表示について

このマニュアルでは、以下の絵表示を使用して、安全に関する重大な注意事項を説明しています。

⚠ 警告	**この指示を無視して誤った取り扱いをすると、使用者が死亡または重傷を負う恐れがあります。**
⚠ 注意	**この指示を無視して誤った取り扱いをすると、使用者が軽傷を負うか物的な損害を受ける恐れがあります。**
⃠	**正しく使用するために、してはいけない（禁止）事項です。**
❗	**正しく使用するために、しなくてはならない（強制）事項です。**

このマニュアルでは、説明の便宜のため以下のように表記します。

MEMO	**関連する情報や補足説明です。**
参照→	**関連する説明が掲載されている項目(マニュアル名、章・節・項)を示します。**
※	**脚注で説明している語句についています。**
PLC	**PLC（プログラマブルコントローラ、シーケンサ）、温調器やインバータなどの周辺機器を指します。**
GP	**(株)デジタル製プログラマブル表示器 GPシリーズの総称です。**
PS-P	**(株)デジタル製 PSシリーズPタイプの総称です。**
PS-G	**(株)デジタル製 PSシリーズGタイプの総称です。**
PL	**(株)デジタル製FAパネルコンピュータ PLシリーズの総称です。**

商標権などについて

本書に記載の会社名、商品名は、各社の商号、商標(登録商標を含む)またはサービスマークです。本製品の表示・記述の中では、これら権利に関する個別の表示は省略しております。

商　　標	権利者
Microsoft, MS, Windows, Windows 95, Windows 98, Windows NT, Windows 2000, Windows XP, Windows CE, Windows エクスプローラ, eMbedded Visual C++, eMbedded Visual Basic	米国マイクロソフト社
Intel, Pentium	米国インテル社
Pro-face	(株)デジタル
IBM, VGA, PC/AT	米国IBM社
Adobe, Acrobat	アドビシステムズ社

なお、上記商号･商標類で、本書での表記が正式な表記と異なるものは以下の通りです。

本書での表記	正式な表記
Windows NT	Microsoft® Windows NT® operating system
Windows 2000	Microsoft® Windows® 2000 operating system
Windows XP	Microsoft® Windows® XP operating system
Windows CE	Microsoft® Windows® CE operating system
MS-DOS	Microsoft® MS-DOS® operating system
Pentium	Intel® Pentium® processors
Acrobat Reader 5.0	Adobe® Acrobat® Reader 5.0

目次

はじめに 1
マニュアル表記について 2
商標権などについて 3

第1章　機能
1 機能の概要 1-2
2 変数へのアクセス 1-3
2.1 変数へのアクセス例 1-4
2.2 データシェアリング機能で参照可能な変数 1-5
2.3 データシェアリング最大アクセス数 1-5

第2章　データシェアリング機能の設定
1 データシェアリング機能の設定手順 2-2
2 データシェアリング設定 2-3
2.1 データシェアリング設定 2-3
2.2 変数の設定 2-4
2.3 スキャングループのデータシェアリング設定 2-4
3 データシェアリングの開始 2-5
3.1 ランタイムの実行 2-5

第3章　データシェアリングAPI機能の設定
1 データシェアリングAPI機能の設定手順 3-2
2 データシェアリングAPIとランタイムの接続 3-3
3 データシェアリングAPI設定 3-3
3.1 データシェアリング設定 3-3
3.2 スキャングループのデータシェアリング設定 3-3
4 外部アプリケーションの作成 3-4
4.1 Pro-Designerのインストール 3-4
4.2 サンプルコード 3-4
4.3 データシェアリングDLLファイルの指定 3-5
4.4 コンフィギュレーションファイルの指定 3-5
4.5 外部アプリケーションの配置 3-5

5 外部アプリケーションからのアクセス 3-6
5.1 データシェアリングAPI一覧 3-6
5.2 データシェアリングAPI初期化 3-6
5.3 ターゲット機との接続 3-7
5.4 データの読み出し/書き込み 3-7
6 データシェアリングAPIの使用例 3-8
7 関数仕様詳細 3-12
8 データシェアリングの開始 3-18
8.1 ランタイムおよび外部アプリケーションの実行 3-18

1 機能

1 機能の概要
2 変数へのアクセス

1 機能の概要

Pro-Designerでは、データシェアリング機能およびデータシェアリングAPI機能を利用できます。

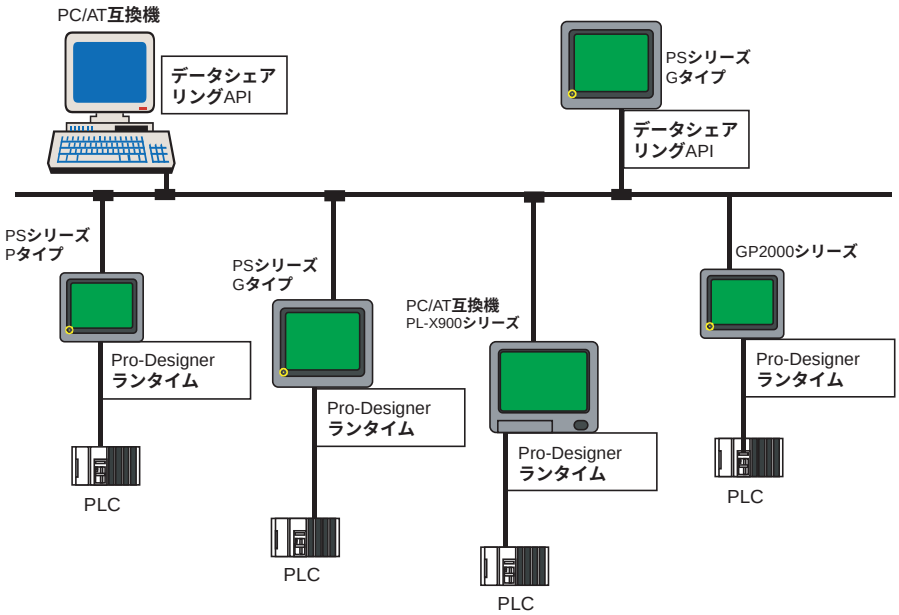
データシェアリング機能

Pro-Designerの同一プロジェクト内のターゲット間で変数を共有する機能です。変数はターゲットごとに作成しますが、ネットワークでつながっているターゲット機同士は、データシェアリング機能を使って他のターゲットの変数の値を参照したり、変更したりすることができます。外部変数を共有すれば他のターゲットに接続されているPLCの値を参照することができます。

データシェアリングAPI機能

ユーザーが独自に開発したアプリケーションプログラムから、Pro-Designerランタイムターゲット機内の変数にアクセスすることができます。
これによりPro-Designerランタイムと外部アプリケーションとの間で、お互いに機能的な補完ができます。

データシェアリング機能とデータシェアリングAPI機能を併用すると、下図のようにネットワークを経由して、各ターゲット機の変数に相互にアクセスすることができます。さらに外部アプリケーションからも各ターゲット機の変数にアクセスすることが可能となります。



データシェアリング機能およびデータシェアリングAPI機能は以下のターゲット機で対応しています。

ターゲット機	データシェアリング機能	データシェアリングAPI機能
PC/AT（PL）	対応	対応
PS-G		
PS-P		非対応
GP※1		

※1　イーサネットをサポートしている機種のみ

2　変数へのアクセス

データシェアリング機能では、データを共有するターゲット機間に、参照される側と参照する側という関係があります。データシェアリング機能での参照する側/参照される側の定義は以下のとおりです。

設定	参照される側ターゲット	参照する側ターゲット
データシェアリング設定	有効（使用する）	有効/無効（使用する/使用しない）を問わない
アニメーションやスクリプトなどでの変数指定	――――――	参照される側ターゲット内の変数

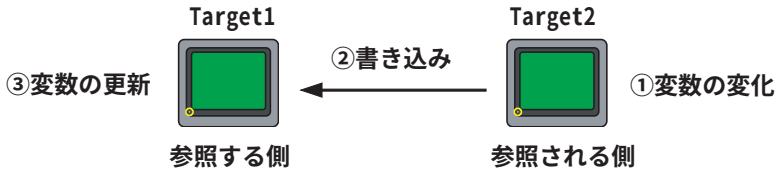
またデータシェアリングAPI機能では、ターゲット機と外部アプリケーションを実行するマシンとの間に以下のような関係があります。

設定	参照される側ターゲット	外部アプリケーションの実行マシン
データシェアリング設定	有効（使用する）	有効/無効（使用する/使用しない）を問わない
外部アプリケーションでの変数指定	――――――	参照される側ターゲット内の変数

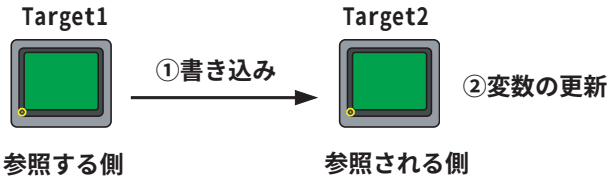
2.1　変数へのアクセス例

データシェアリング機能により、ターゲット機間で以下のような変数のアクセス機能が利用できます。

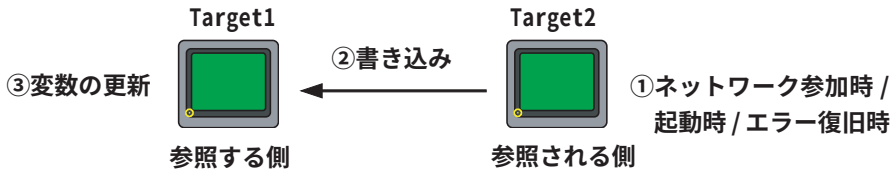
・ 参照される側の変数の値が変化すると、参照する側へ変化した変数の値が書き込まれます。



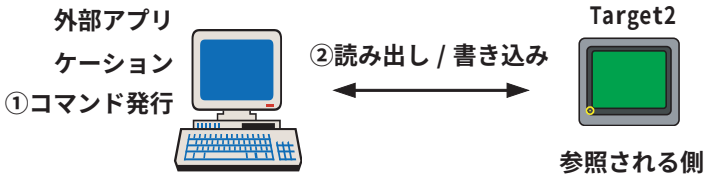
・ 各ターゲット機の変数の値の変化に関わらず、必要に応じて参照する側から参照される側へ変数の値を書き込むことができます。



・ 参照される側のターゲット機が新しくネットワークに参加すると、参照する側へ変数が書き込まれます。起動時またはエラー復旧時も同様です。



・ 各ターゲット機の変数の変化に関係なく、外部アプリケーションによりターゲット機の変数の読み出し/書き込みができます。この機能はデータシェアリングAPI使用時のみ有効です。



2.2　データシェアリング機能で参照可能な変数

データシェアリング機能で参照する側から参照される側の変数にアクセスするには、参照する側ターゲットでPro-Designerのアニメーションやスクリプトなどの機能を用いて参照される側の変数を指定します。参照→第2章 2.2 変数の設定

Pro-Designerのアニメーションやスクリプト、スマートパーツなどでデータシェアリング機能による変数の参照が可能です。

MEMO

・以下の機能ではデータシェアリング機能を用いて他のターゲットの変数を参照することはできません。

グラフ表示器
アラームサマリ
ヒストリカルトレンド
ヒストリカルデータ表示器
データグラフ

2.3　データシェアリング最大アクセス数

ターゲット機の最大アクセス台数

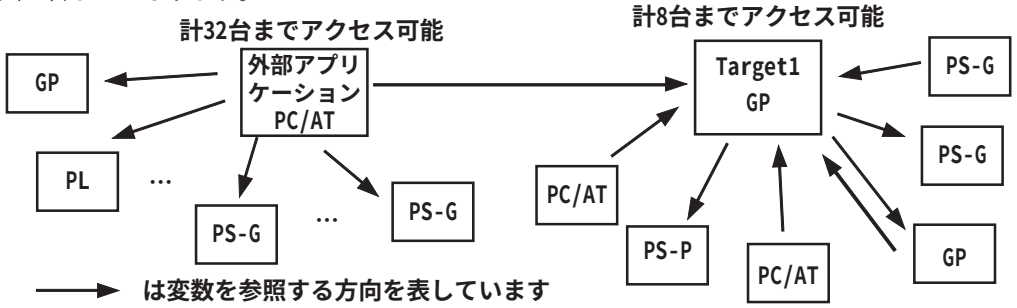
データシェアリング機能およびデータシェアリングAPI機能により同時にアクセスされるターゲット機の台数には制限があります。

ターゲット機の種類ごとの最大アクセス台数（同時にアクセスされるターゲット機の最大数）は以下のとおりです。

ターゲット機	最大アクセス台数
PC/AT （PLシリーズ）	32
PS-G	16
GP、PS-P	8
Factory Gateway	4

1台のターゲット機が参照する台数と参照される台数を合計した数がアクセス台数となります。

下図ではTarget1が参照する台数は3台、参照される台数は5台で、Target1のアクセス台数は8台となります。また、PC/AT互換機が外部アプリケーションで変数を参照するターゲット機の台数は最大32台までとなります。



変数の最大アクセス数

データシェアリング機能およびデータシェアリングAPI機能により同時にアクセスされる変数の数には制限があります。ターゲット機の種類ごとの最大アクセス数（同時にアクセスされる変数の最大数）の目安は以下のとおりです。

ただしこの目安は設計上の限界数ではなく、データ更新のパフォーマンス速度から設定した数です。これらの値は画面データ処理数などに依存します。

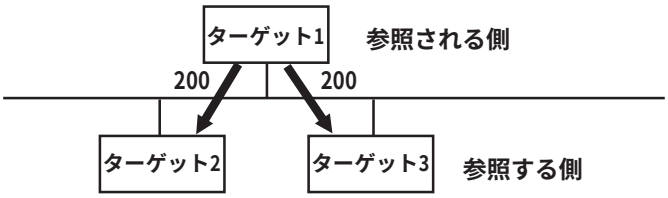
ターゲット機	最大アクセス数の目安
PC/AT （PLシリーズ）	400
PS-G	150
GP、PS-P	150
Factory Gateway	75

一つのターゲット機が参照する変数と参照される変数を合計した数がアクセス数となります。

以下にターゲット機がPC/ATである場合をモデルとして最大アクセス数の例を示します。

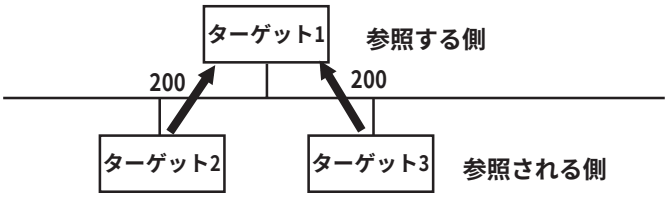
例1）参照される側の最大アクセス数

1台のターゲット機は2台のターゲット機から200個ずつ計400個まで同時に変数を参照されることができます。



例2）参照する側の最大アクセス数

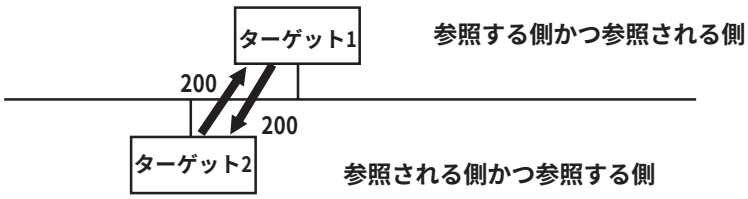
1台のターゲット機は2台のターゲット機から200個ずつ400個まで同時に変数を参照することができます。



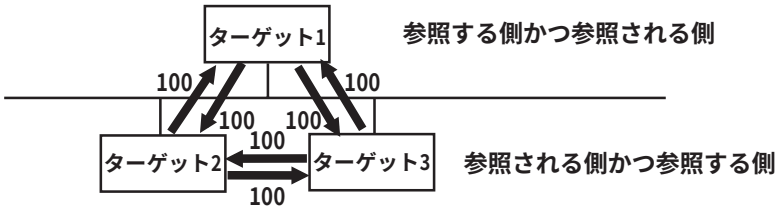
例3）参照される側および参照する側のどちらにもなる場合のアクセス数

1台のターゲット機が、参照される側および参照する側として相互に機能する場合、同時に参照する変数と参照される変数の合計がアクセス数となります。

2台のターゲット機が相互に参照する場合



3台のターゲット機が相互に参照する場合

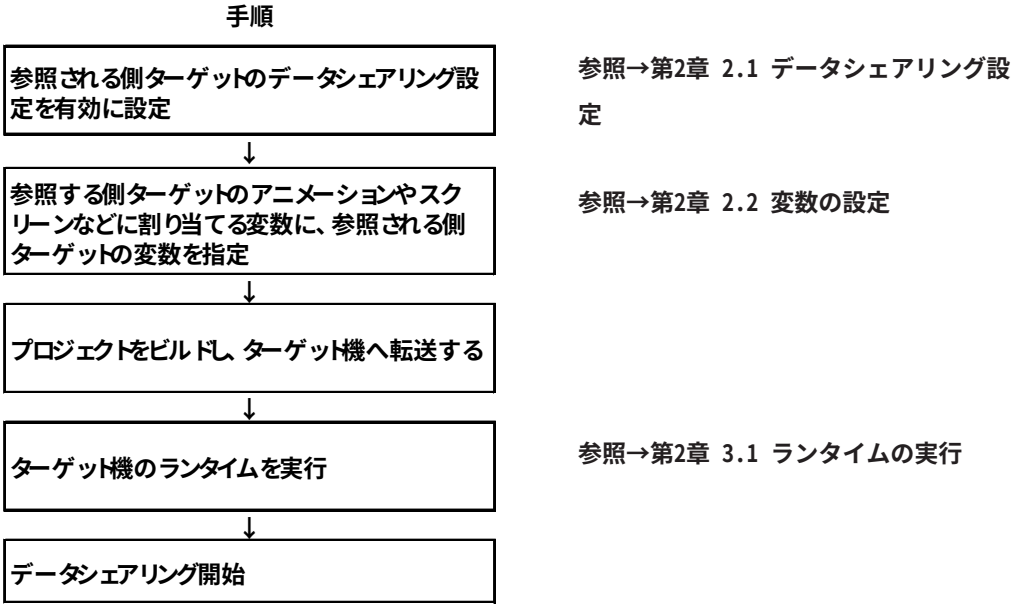


2 データシェアリング機能の設定

1 データシェアリング機能の設定手順
2 データシェアリング設定
3 データシェアリングの開始

1 データシェアリング機能の設定手順

データシェアリング機能を使用してターゲット機間でデータを参照する手順の概要を示します。



MEMO

- データシェアリングAPI機能の設定手順については「第3章 データシェアリングAPI機能の設定」を参照してください。

2　データシェアリング設定

2.1　データシェアリング設定

参照される側となるターゲット機のデータシェアリングのプロパティを有効にし、IPアドレスとポート番号を設定します。
データシェアリング有効時はターゲット内のすべての変数は同一プロジェクト内で共有されます。
データシェアリング無効時はターゲット内の変数は共有されません。
参照する側ターゲット機のデータシェアリング有効/無効に関わらず、参照される側ターゲット機がデータシェアリング有効であれば、参照する側のターゲット機からアクセスできます。

MEMO

・データシェアリング機能使用時にターゲット機上でランタイム設定メニューを使ってネットワークの設定変更を行うと通信できなくなります。IPアドレスを変更する場合はPro-Designerでプロジェクトを開き、ターゲットのプロパティで［IPアドレス］を変更後、ダウンロードしてください。ランタイム設定メニューについての詳細はオンラインヘルプを参照してください。

設定手順

(1)  ナビゲータの  ［プロジェクト］タブで、参照される側となる  ターゲットを右クリックし、表示されたショートカットメニューから［プロパティ］を選択します。

(2) データシェアリングを有効にするには  プロパティの［データシェアリング］プロパティで「使用する」を選択します。

(3) ［IPアドレス］と［ポート番号］を設定します。
［タイプ］には「イーサネット」が表示されます。このプロパティは読み取り専用です。

MEMO

・以下の場合はデータシェアリング機能を有効にするためにすべてのターゲットにプロジェクトの再ダウンロードが必要です。

1. **ターゲットの［データシェアリング］プロパティが変更された場合**
2. **ターゲットの［IPアドレス］プロパティが変更された場合**
3. **ターゲットが削除された場合**

2.2 変数の設定

アニメーションやスクリプトの変数を設定する箇所に「（Target名）.（ 変数名）」で指定します。

例） **Target1.Tank1** ：他のターゲット機からTarget1内のTank1の変数を参照する

スクリプトでの表記例 ：

```
int temp;
temp = 123;
Target1.Tank1.Write(temp);
```

MEMO

- **変数の作成は必ずターゲットごとに行ってください。**
- **以下の場合はデータシェアリング機能を有効にするためにすべてのターゲットにプロジェクトの再ダウンロードが必要です。**
  1. **共有されている変数の名前が変更または削除された場合**
  2. **共有されている変数のプロパティが変更された場合**

2.3 スキャングループのデータシェアリング設定

共有する変数が外部変数の場合、共有する変数の通信周期をスキャングループごとに設定します。スキャングループの［データシェアリング］プロパティを「ダイナミック」または「常時」から選択します。通常は「ダイナミック」を選択してください。

「ダイナミック」を選択すると、以下の場合に変数値が更新されます。

- 変数が使用されているパネルが表示されている場合
- 変数の［履歴データ］プロパティが「使用する」の場合
- 変数アラームが有効の場合
- 変数が折れ線グラフで使用されている場合
- 変数がスクリプトで使用されている場合

「常時」を選択すると、パネルが変更されてもスキャンタイプで設定された一定の周期で変数の値は更新され、共有設定されていない変数も永続的に通信されるため、ネットワークの負荷が大きくなります。変数の値が常に最新でなければならない場合のみ「常時」を選択してください。

設定手順

(1) ターゲットの接続機器内の スキャングループを右クリックし、表示されたショートカットメニューから［プロパティ］を選択します。
プロパティにターゲットのプロパティが表示されます。

(2) プロパティの［データシェアリング］プロパティから「常時」または「ダイナミック」のいずれかを選択します。
デフォルトでは「ダイナミック」が設定されています。

3　データシェアリングの開始

3.1　ランタイムの実行

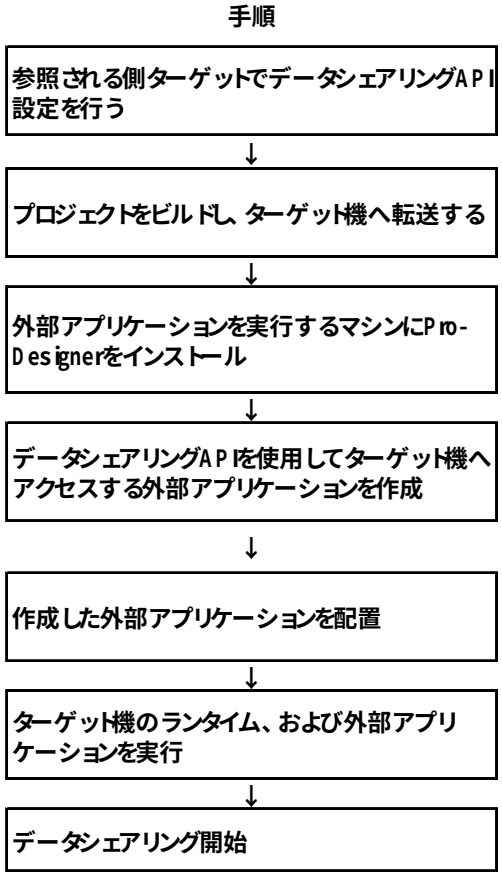
データシェアリング設定をしたターゲット機でランタイムを実行することにより、データシェアリングが開始されます。

3 データシェアリングAPI機能の設定

1 データシェアリングAPI機能の設定手順
2 データシェアリングAPIとランタイムの接続
3 データシェアリングAPI設定
4 外部アプリケーションの作成
5 外部アプリケーションからのアクセス
6 データシェアリングAPIの使用例
7 関数仕様詳細
8 データシェアリングの開始

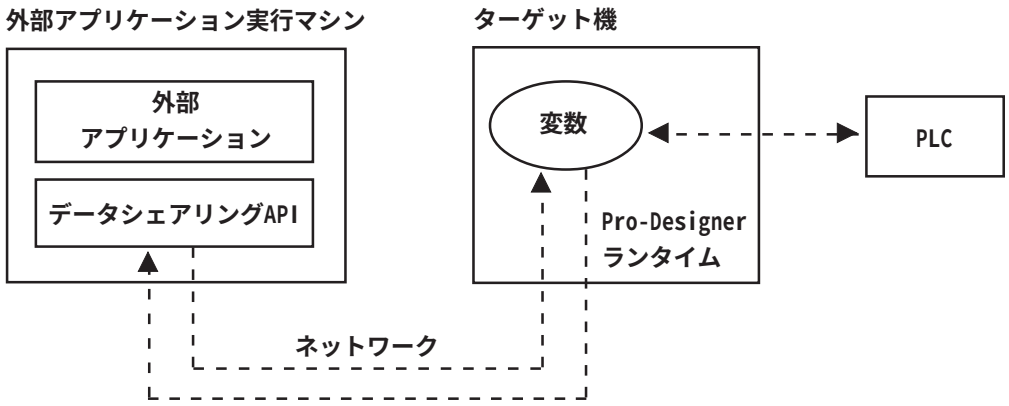
1 データシェアリングAPI機能の設定手順

データシェアリングAPI機能を使用してユーザーアプリケーションからターゲット機の変数へアクセスする手順を示します。



2　データシェアリングAPIとランタイムの接続

データシェアリングAPIを使用すると、外部アプリケーションはランタイムの変数やPLCのデバイスにアクセスすることができます。



3　データシェアリングAPI設定

3.1　データシェアリング設定

データシェアリングAPI機能を使用する場合にも、外部アプリケーションがアクセスするターゲット機にはデータシェアリング機能同様、データシェアリング設定が必要です。設定手順については「第2章 2.1 データシェアリング設定」を参照してください。

3.2　スキャングループのデータシェアリング設定

共有する変数が外部変数の場合、データシェアリングAPI機能を使用する場合にも、外部アプリケーションがアクセスするターゲット機にはデータシェアリング機能同様、共有する変数の通信周期をスキャングループごとに設定します。設定手順については「第2章 2.3 スキャングループ設定」を参照してください。

4 外部アプリケーションの作成

4.1 Pro-Designerのインストール

外部アプリケーション実行マシンには、Pro-Designerエディタ、Pro-Designerランタイム、またはPro-Server通信ツールのいずれかをインストールしておく必要があります。

MEMO

・インストール先には［\pro-face］フォルダが作成されます。
例えば[C:\Program Files]フォルダにインストールした場合、
[C:\Program Files\pro-face\...]となります。

4.2 サンプルコード

外部アプリケーションはC言語またはC++言語で作成します。外部アプリケーションを作成する際はサンプルコードを参考にしてください。

Pro-Designerエディタをインストールしたパソコンの[\pro-face\docs\samples\dsapi]フォルダには以下のサンプルコードが用意されています。これらのサンプルコードはMicrosoft® Visual C++® Ver.6.0で作成されています。

ファイルまたはフォルダ	内容
\Dsapi\main.cpp	サンプルコード
\Dsapi\Dsapi.dsw	プロジェクトワークスペース
\Dsapi\Dsapi.dsp	プロジェクトファイル
\Dsapi\RuntimeAdapter.h	ヘッダーファイル データシェアリングAPIで使われているデータ型が明記されています。参照→第3章 7 関数仕様詳細 [\pro-face\Docs\include]フォルダに格納されているファイルと同じものです
\Dsapi\Debug	デバッグ用のアプリケーションを格納
\Dsapi\Release	リリース用のアプリケーションを格納

4.3　データシェアリングDLLファイルの指定

データシェアリングAPIはWindowsのDLL（RuntimeAdapter.dll）として提供されます。データシェアリングAPIを使用するには、RuntimeAdapter.dllモジュールを呼び出し側プロセスのアドレス空間にマップし、DLL内のエクスポート関数のアドレスを取得する必要があります。

外部アプリケーション作成時に、Windows APIの関数“Load Libraly()”でDLLファイル（RuntimeAdapter.dll）を指定します。

RuntimeAdapter.dllは以下のフォルダに用意されています。

Windows NT 4.0、Windows 2000、またはWindows XPの場合

[\pro-face\Pro-Runtime\public\Bin]フォルダ

Windows CEの場合

[\Storage Card1\public\Bin\WinCE]フォルダ

4.4　コンフィギュレーションファイルの指定

コンフィギュレーションファイル（Project.cfg）はデータシェアリングAPIの設定ファイルです。InitRuntimeAdapterEx関数を使用する場合に指定します。InitRuntimeAdapter関数を使用する場合は不要です。参照→5.1 データシェアリングAPI一覧

InitRuntimeAdapterEx関数の引数“UNICHAR* ConfigFilename”でコンフィギュレーションファイル（Project.cfg）を指定します。その際は、絶対パスを含んだファイル名を指定してください。

Project.cfgはPro-Designerエディタをインストールしたパソコンの[\Pro-face\Docs\CFG］フォルダに用意されています。必要に応じてコピーしてご使用ください。

参照→「第3章 5.1 データシェアリングAPI一覧、7 関数仕様詳細」

4.5　外部アプリケーションの配置

外部アプリケーションの実行ファイル（exeファイル）は任意のフォルダに配置できます。

ただしWindows CEの場合は、外部アプリケーションでDLLファイルを絶対パスで指定するか、外部アプリケーションの実行ファイルを[\Storage Card1\public\BinWinCE]フォルダに配置する必要があります。

5 外部アプリケーションからのアクセス

5.1 データシェアリングAPI一覧

外部アプリケーションのAPIの概要は以下のとおりです。詳細については「第3章 5.7 関数仕様詳細」を参照してください。

API名	内容
InitRuntimeAdapter	データシェアリングAPIを初期化します。最初に呼び出される必要があります。 参照→5.2 データシェアリングAPIの初期化
InitRuntimeAdapterEX	データシェアリングAPI初期化のための拡張機能です。最初に呼び出される必要があります。 参照→5.2 データシェアリングAPIの初期化
ConnectToVars	外部アプリケーションの変数リストと参照されるターゲットの変数リストを接続します。接続が成功した変数には接続ハンドルが、接続に失敗した変数にはエラー値（-1）が返されます。接続ハンドルリストが返されるバッファの確保と解放は呼び出し側で設定します。
DisconnectFromVars	接続ハンドルリストに設定されているすべての接続を切断します。
WriteDataToVar	接続されている変数へデータを書き込みます。書き込みが成功するとTRUEが返されます。
RegConnectErrorInformCallback	変数への接続に成功/失敗した場合や、接続が切断された場合に、データシェアリングAPIが呼び出す処理を登録します。
RegUpdateDataCallback	ランタイムから新しいデータを読み込んだ場合、データシェアリングAPIが呼び出す処理を登録します。
ShutdownRuntimeAdapter	データシェアリングAPIをシャットダウンします。

5.2 データシェアリングAPI初期化

InitRuntimeAdapter関数またはInitRuntimeAdapterEX関数を使用してデータシェアリングAPIを初期化します。データシェアリングAPIを初期化した後、外部アプリケーションは必要なコールバック機能をデータシェアリングAPIに登録する必要があります。

ランタイムと外部アプリケーションが同一マシン上で実行される場合や、同一マシン上で複数の外部アプリケーションを実行する場合はInitRuntimeAdapter関数ではなくInitRuntimeAdapterEX関数を使用してください。

5.3　ターゲット機との接続

・ 外部アプリケーションを接続するには、ConnectToVars関数とDisconnectFromVars関数の2つの機能を使用します。ConnectToVars関数は変数接続要求を処理する機能、DisconnectFromVars関数は確立している接続を切断する機能です。

・ ConnectToVars関数は非同期です。この関数は呼び出しスレッドをブロックしません。参照されるターゲット機への接続を要求した後すぐにリターンします。その後、接続要求に対する応答が届くと、データシェアリングAPIはアプリケーションがRegConnectErrorInformCallback関数経由で登録したコールバック関数を使用し、接続状態を通知します。

5.4　データの読み出し/書き込み

データの読み出し

・ データシェアリングAPIからデータを受信するため、外部アプリケーションはRegUpdateDataCallback関数を使用してコールバック関数を登録する必要があります。この関数は、データシェアリングAPIが参照されるターゲット機から新しいデータを受信すると起動します。

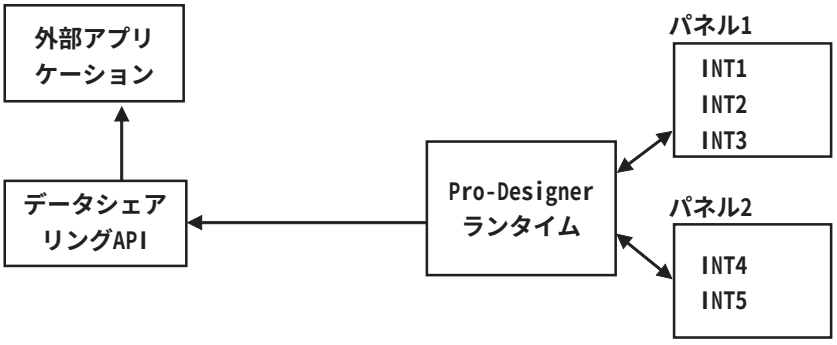
データの書き込み

・ WriteDataToVarを使用すると、常時、外部アプリケーションからターゲット機の変数にデータを書き込むことができます。アクセスが不要となった変数は、DisconnectFromVars関数により接続を終了するか、ShutdownRuntimeAdapter関数によりデータシェアリングAPIを終了します。

6 データシェアリングAPIの使用例

状況別に設定されたデータシェアリングAPIの使用例を以下に示します。

例1）外部アプリケーションがランタイムからデータを読み出します



InitRuntimeAdapterEX(ランタイムディレクトリ、コンフィギュレーションファイル)

- このAPIは1プロセスにつき1回呼び出されデータシェアリングAPIの初期化を1回実行します。
- ランタイムディレクトリはデータシェアリングAPIが使用するルートディレクトリです。
  例：\pro-face\Pro-Runtime\public
- ランタイムと外部アプリケーションが同じマシン上で実行されている場合、データシェアリングAPIのコンフィギュレーションファイルとしてproject.cfgを指定します。

RegUpdateDataCallBack(UpdateFn)

- UpdateFn()はデータが変更された場合にデータシェアリングAPIから呼び出されるコールバック関数です。
- 接続が確立されると各変数に初期値を設定するため、UpdateFn()は接続リストの順番で各変数ごとに呼び出されます。

ConnectToVars()

- 外部アプリケーションは現在のパネルやパネル変更を認識しないため、ランタイムのすべての変数に接続します。
- 変数に接続する場合は、VAR_READ_ONLY_ATTRIB （RuntimeAdapter.h） を使用します。
- ConnectToVarsは直ちにリターンします。変数への接続に失敗した（変数名がランタイムに存在しない）などのエラーが発生した場合や接続に成功した場合、RegConnectErrorInformで登録されたErrorFn()が呼び出されます。

DisconnectFromVars()

・ 指定した変数への接続を切断する機能です。

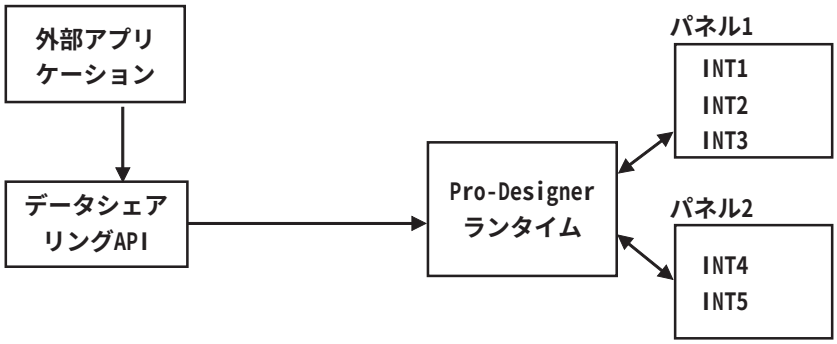
RegConnectErrorInformCallback(ErrorFn)

・ ErrorFn()は接続時および異常発生時に、データシェアリングAPIから呼び出されるコールバック関数です。

ShutdownRuntimeAdapter()

・ 1プロセスにつき1回呼び出され、プロセス終了時にデータシェアリングAPIよりアンロードされます。

例2）外部アプリケーションランタイムへデータを書き込みます



InitRuntimeAdapterEX(ランタイムディレクトリ、コンフィギュレーションファイル)

・ このAPIは1プロセスにつき1回呼び出されデータシェアリングAPIの初期化を1回実行します。
・ ランタイムディレクトリはデータシェアリングAPIが使用するルートディレクトリです。
例： \pro-face\Pro-Runtime\public
・ ランタイムと外部アプリケーションが同じマシン上で実行されている場合、データシェアリングAPIのコンフィギュレーションファイルとしてproject.cfgを指定します。

ConnectToVars()

・ 外部アプリケーションは現在のパネルやパネル変更を認識しないため、ランタイムのすべての変数に接続します。
・ 変数に接続する場合は、VAR_WRITE_ONLY_ATTRIB（RuntimeAdapter.h）を使用します。
・ ConnectToVarsは直ちにリターンします。変数への接続に失敗した（変数名がランタイムに存在しない）などのエラーが発生した場合や接続に成功した場合、RegConnectErrorInformで登録されたErrorFn()が呼び出されます。

WriteDataToVar()

・ ランタイム上の変数にデータ値を書き込むため、外部アプリケーションはこの機能を呼び出します。ConnectToVarsから返されたハンドルを使用して、変数を指定します。1度に1変数のみ書き込みます。2つ以上の変数を書き込む場合は複数回呼び出します。

DisconnectFromVars()

・ 指定した変数への接続を切断する機能です。

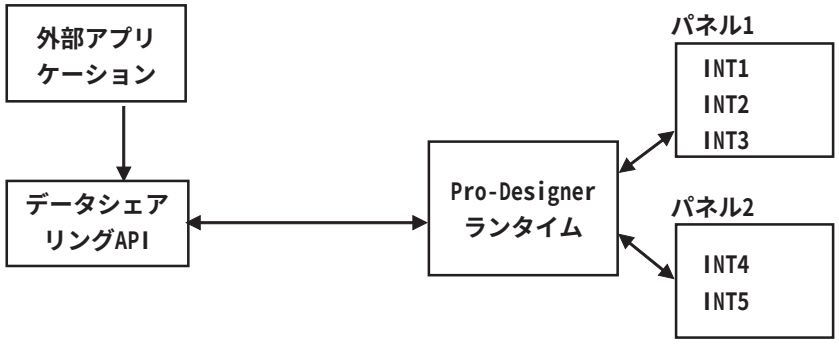
RegConnectErrorInformCallback(ErrorFn)

・ ErrorFn()は接続時および異常発生時に、データシェアリングAPIから呼び出されるコールバック関数です。

ShutdownRuntimeAdapter()

・ 1プロセスにつき1度呼び出され、プロセス終了時にデータシェアリングAPIよりアンロードされます。

例3）外部アプリケーションがランタイムへデータを読み書きします



InitRuntimeAdapterEX(ランタイムディレクトリ、コンフィギュレーションファイル)

・ このAPIは1プロセスにつき1回呼び出され、データシェアリングAPIの初期化を1回実行します。

・ ランタイムディレクトリとはデータシェアリングAPIが使用するルートディレクトリです。
例： \pro-face\Pro-Runtime\public

・ ランタイムと外部アプリケーションが同じマシン上で実行されている場合、データシェアリングAPIのコンフィギュレーションファイルにproject.cfgを指定します。

ConnectToVars()

- アプリケーションは現在のパネルやパネル変更を認識しないため、ランタイムのすべての変数に接続します。
- 変数に接続する場合は、VAR_READ_WRITE_ATTRIB （RuntimeAdapter.h） を使用します。
- ConnectToVarsは直ちにリターンします。変数への接続に失敗した（変数名がランタイムに存在しない）などのエラーが発生した場合や接続に成功した場合、RegConnectErrorInformで登録されたErrorFn()が呼び出されます。

DisconnectFromVars()

- 指定した変数への接続を切断する機能です。

WriteDataToVar()

- ランタイム上の変数にデータ値を書き込むため、外部アプリケーションはこの機能を呼び出します。ConnectToVarsから返されたハンドルを使用して、変数を指定します。1度に1変数のみ書き込みます。2つ以上の変数を書き込む場合は複数回呼び出します。

RegConnectErrorInformCallback(ErrorFn)

- ErrorFn()は接続時および異常発生時に、データシェアリングAPIから呼び出されるコールバック関数です。

RegUpdateDataCallBack(UpdateFn)

- UpdateFn()はデータが変更された場合、データシェアリングAPIから呼び出されるコールバック関数です。
- 接続が確立されると各変数に初期値を設定するため、UpdateFn()は接続リストの順番で各変数ごとに呼び出されます。

ShutdownRuntimeAdapter()

- 1プロセスにつき1度呼び出され、プロセス終了時にデータシェアリングAPIよりアンロードされます。

7 関数仕様詳細

データシェアリングAPIの関数仕様詳細を示します。
引数内の[in] はDLLが受け取る引数、[out]はDLLが渡す引数です。

関数仕様詳細で使用しているデータ型は、ヘッダーファイル（RuntimeAdapter.h）で定義されています。詳細は下表およびサンプルのRuntimeAdapter.hを参照してください。参照→第3章 4.2 サンプルコード

変数型	値	定数
unsigned char	―	UINT8;
unsigned short	―	UINT16;
unsigned int	―	UINT32;
signed char	―	INT8;
signed short	―	INT16;
signed int	―	INT32;

属性	値	定数
Write Only	1	UINT8;
Read Only	2	UINT16;
Read Write	3	UINT32;

要求データ型	値	定数
Integer	0	VAR_TYPE_INT
Float	1	VAR_TYPE_FLOAT
String	2	VAR_TYPE_STRING
Discrete	3	VAR_TYPE_DISCRETE

InitRuntimeAdapter

呼び出し形式

```
bool InitRuntimeAdapter
(
SystemPath
)
```

戻り値

bool
TRUE：正常に実行
FALSE：それ以外

引数

UNICHAR* SystemPath

[in] データシェアリングAPIのルートディレクトリへのパス

処理概要

データシェアリングAPIを初期化します。データシェアリングAPIで最初に呼び出される必要があります。

InitRuntimeAdapterEx

呼び出し形式

```
bool InitRuntimeAdapterEX
(
SystemPath,
ConfigFilename
)
```

戻り値

bool

TRUE：正常に実行

FALSE：それ以外

引数

UNICHAR* SystemPath

[in] データシェアリングAPIのルートディレクトリへのパス

UNICHAR* ConfigFilename

[in] データシェアリングAPIのコンフィギュレーションファイル（Project.cfg）へのパス

処理概要

データシェアリングAPIを初期化します。データシェアリングAPIで最初に呼び出される必要があります。Pro-Designerランタイムと外部アプリケーションが同じマシンで実行されている場合はこのAPIを使用します。

ConnectToVars

呼び出し形式

```
bool ConnectToVars
(
UNICHAR* ServerAddr,
UINT16          NumOfVars,
```

```
UNICHAR* VarNameList[],
BYTE*           DataTypeList,
BYTE*           DirAttribList,
UINT32*  AssignAppHandleList,
CONNECTHANDLES* RetAdapterHandleList
)
```

戻り値

bool

TRUE：正常に実行

FALSE：それ以外

引数

UNICHAR* ServerAddr,

[in] ターゲット機のIPアドレスとポート番号

UINT16 NumOfVars,

[in] 以下のリストの変数数

UNICHAR* VarNameList[],

[in] データシェアリング変数名のリスト

BYTE* DataTypeList,

[in] 変数の種類のリスト（Int、Discrete、Float、String）

BYTE* DirAttribList,

[in] 変数の属性リスト（R、W、R/W）

UINT32* AssignAppHandleList,

[in] 接続設定されたアプリケーションのハンドルリスト。データシェアリングAPIから更新されたデータはすべてこのハンドルを元にして参照されます。

CONNECTHANDLE* RetAdapterHandleList

[out] データシェアリングAPIからリターンされたハンドルリスト。データシェアリングAPIへの要求はすべてこのハンドルを元にして参照されます。

処理概要

外部アプリケーションの変数リストと参照されるターゲットの変数リストとを接続します。接続が成功した変数には接続ハンドルが、接続に失敗した変数にはエラー値（1）が返されます。接続ハンドルリストが返されるバッファの確保と解放は呼び出し側で設定します。

UNICHAR* VarNameList[]には、変数を「（ターゲット名）_（変数名）」で指定します。

└── **アンダーバー**

配列変数を指定する場合は「（ターゲット名）_（変数名$配列番号）」で指定します。

例）　ターゲット**Target1**の配列変数**Tank[10]**の1番目の配列を指定する場合

引数 UNICHAR* VarNameList[]での表記例 ： **Target1_Tank$0**

ランタイムと外部アプリケーションが同一マシン上で実行される場合、ConnectToVarsに与えるIPアドレスの引数は「INET:127.0.0.1:xxxx」とします（xxxxにはエディタで指定したポート番号が入ります）。IPアドレスを127.0.0.1にするとネットワークには参加しないで同じマシン上のランタイムにアクセスします（ターゲット機に別のIPアドレスが設定されていても問題ありません）。

DisconnectFromVars

呼び出し形式

```
bool DisconnectFromVars
(
NumOfHandles,
AppHandleList
)
```

戻り値

bool

TRUE：正常に実行

FALSE：それ以外

引数

UINT16　NumOfHandles,

　[in] 以下のリストのハンドル数

CONNECTHANDLE* AdapterHandleList

　[in] 登録後にデータシェアリングAPIからリターンされるハンドルリスト

処理概要

接続ハンドルリストに設定されているすべての接続を切断します。

WriteDataToVar

呼び出し形式

```
bool WriteDataToVar
(
AdapterHandle,
DataLen,
Data
)
```

戻り値

bool

TRUE：正常に実行

FALSE：それ以外

引数

CONNECTHANDLE* AdapterHandle,

[in] 接続ハンドル

UINT16 DataLen,

[in] データのバイト数

void* Data

[in] 書き込みデータ

処理概要

接続済みのデータシェアリング変数へデータを書き込みます。

同じ値が変数に書き込まれる時は、Pro-Designerランタイム側でデータの更新は行われません。

RegConnectErrorInformCallback

呼び出し形式

```
void RegConnectErrorInformCallback
( bool (
    CONNECT_STATUS  ConnectionStatus,
    UINT16      NumOfHandles,
    UINT32*     AppHandleList )
)
```

戻り値

void

引数

bool(*)

(CONNECT_STATUS ConnectionStatus,

データシェアリング接続の状態

UINT16 NumOfHandles,

AppHandleListのハンドル数

UINT32* AppHandleList)

アプリケーションハンドルリスト

処理概要

ランタイムへの接続時およびエラー発生時、データシェアリングAPIが呼び出すコールバック関数を登録します。ConnectionStatusは接続の状況をRTA_CONNECTING、RTA_CONNECTED、RTA_TAGNAME_ERROR、RTA_TOO_MANY_TAGS_ERROR、あるいはRTA_VERSION_ERRORのいずれかで表示します。ConnectionStatusが、RTA_TOO_MANY_TAGS_ERROR、RTA_ERROR、RTA_TAGNAME_ERRORの場合、NumOfHandlesとAppHandlListは有効です。また、サポートされている最大数を超えて変数にアクセスすると、RTA_TOO_MANY_TAGS_ERRORが返されます。さらに、変数リストにない変数にアクセスすると、RTA_TAGNAME_ERRORが返されます。

エラーが生じた場合、ConnectToVars機能はサーバにつながるまで接続要求を一定周期で行います。このとき、RTA_CONNECTINGが同じ周期でコールバック関数に渡されます。接続が正常に確立されすべての変数が見つかった場合、RTA_CONNECTEDが返されます。見つからなかった変数はRTA_ERRORにより表示されます。つまり、接続が切断されると、ErrorInformCallbackはRTA_CONNECTINGを、接続が確立されるとRTA_CONNECTEDを返します。

RegUpdateDataCallback

呼び出し形式

```
void RegUpdateDataCallback
( bool (
    UINT32      AppHandle,
    UINT16      DataByteLen,
    void*       Data   )
)
```

戻り値

void

引数

```
bool(*)
(UINT32  AppHandle,
    アプリケーションハンドル
UINT16   DataByteLen,
    Dataの合計データバイト数
void*    Data)
    更新するデータ
```

処理概要

ターゲット機から新しいデータを読み込んだ場合、データシェアリングAPIが呼び出すコールバック関数を登録します。

接続が確立した場合、または切断された場合、データシェアリングAPIはAppHandleListに、接続が確立、または切断されたランタイムターゲット機のIPアドレスとポート番号をUNICODE文字列へのポインタとして格納し、RegConnectErrorInformCallback関数をコールバックします。

ShutdownRuntimeAdapter

呼び出し形式

```
bool ShutdownRuntimeAdapter
(


)
```

戻り値

bool

TRUE：正常に実行

FALSE：それ以外

引数

void

処理概要

データシェアリングAPIをシャットダウンします。1プロセスにつき1度だけ呼び出してください。

8 データシェアリングの開始

8.1 ランタイムおよび外部アプリケーションの実行

データシェアリングAPI設定を行ったターゲット機および外部アプリケーション実行マシンのそれぞれで、ランタイムおよび外部アプリケーションを実行することにより、データシェアリングが開始されます。

同一マシン上でランタイムと外部アプリケーションを実行する場合は、先にランタイムを起動してから外部アプリケーションを起動するようにしてください。このような場合には、ランタイムをスタートアップに登録し、ランタイムからスクリプトCreateProcessを使って外部アプリケーションを起動することをお勧めします。