Digital
Human Machine Interface

Pro-face®

データシェアリングAPI ドライバ

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、まことにありがとうございます。

データシェアリングAPIドライバ（以下、DSAPIドライバと称します）は、Pro-Designerで提供されている、外部アプリケーションからPro-Designerの変数にアクセスするためのプログラムモジュールであるデータシェアリングAPIを、より使いやすくすることを目的として開発されています。

このドキュメントでは、ドライバが提供する関数仕様の説明を行いますが、実際にアプリケーションをプログラミングされる場合は、このドキュメントと共にサンプルソースを御覧いただければ、よりいっそうご理解が深まるものと思います。

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。

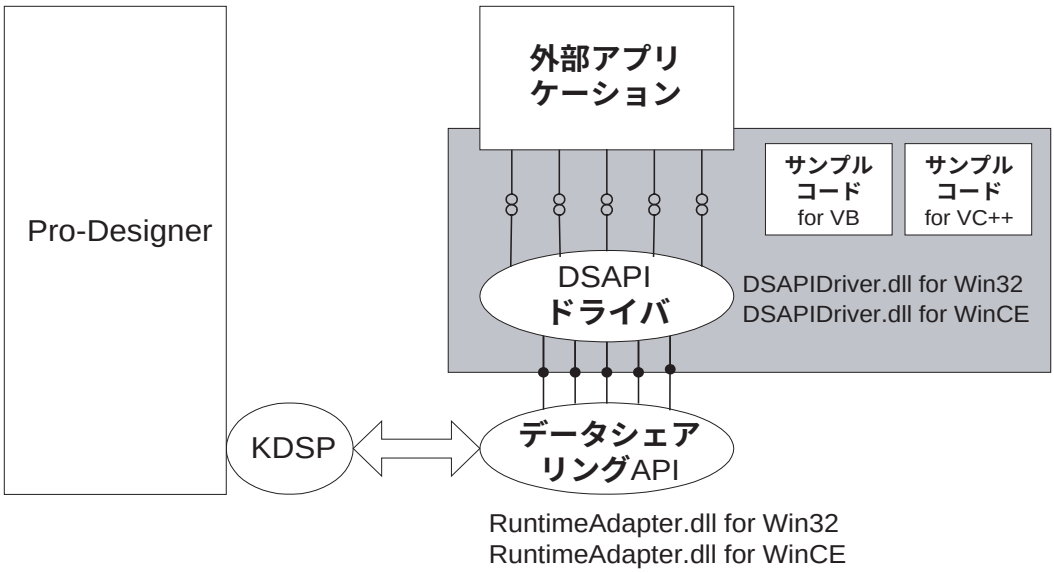
本書に記載の商品名は、それぞれの権利者の商標または登録商標です。

目次

はじめに 1
目 次 2
第 1 章 Pro-Designer と全体イメージ 3
第 2 章　DSAPI ドライバの構成 4
　環境設定 5
第 3 章　初期化 / オープン 6
　VC++ と VB との相違点 6
　使用手順 6
　通信を行う変数の登録変更 7
　再設定手順 8
第 4 章　データのリード / ライト 9
　Index（変数ハンドラ） 9
　DSAPI_AddVariable()を使用した場合 9
第 5 章　DSAPI ドライバの I/F 10
　Win32 用　DSAPIDriver.dll 10
　WinCE 用　DSAPIDriver.dll 18
第 6 章　各種制限事項 25

第 1 章 Pro-Designer と全体イメージ

データシェアリング API は DSAPI ドライバと共に、ユーザーが作成する外部アプリケーションのプロセスとして動作します。



図表 1-1 各プロセス・モジュールイメージ

外部アプリケーションを作成する時、直接データシェアリング API の関数を用いてプログラミングすることができますが、データシェアリング API はある程度経験のある Visual C++® (以下、VC++ と称します) ユーザーを想定して設計されていますので、若干複雑な構成になっており、Visual Basic (以下、VB と称します) ユーザー等には使いづらい API である感は否めません。そういった意味で簡単にプログラミングを行いたいという、ユーザーの要求に合わせた形で DSAPI ドライバでは、極力シンプルにまとめた API 構成に設計されています。

注意　　DSAPI ドライバはデータシェアリング API を読み込んで使用しており、仕様的な規制事項はデータシェアリング API に依存します。

第 2 章　DSAPI ドライバの構成

DSAPI ドライバが提供する関数は、大きく 5 つのカテゴリに分類することができます。

初期化・・・・・DSAPI を動作させるのに必要な各種設定値をドライバに登録します
オープン・・・登録された設定値を用いて DSAPI を初期化します
クローズ・・・終了処理を行います
リード・・・・・変数のハンドラを与え、それに該当するデータを読み込みます
ライト・・・・・変数のハンドラを与え、そこに値を書き込みます

これらの関数を使用するアプリケーションでの処理の流れとして、

初期化→オープン→｛リード、ライト｝→終了

のようにまとめることができます。

アプリケーションを作成する環境（VB や VC++）の違いによって、使用に適した関数は異なります。それぞれのケースでの処理手順についてはサンプルソースをご覧ください。

ドライバで提供される関数を使用するにはデータの構造を把握する必要があります。

以下に、簡単に説明します。

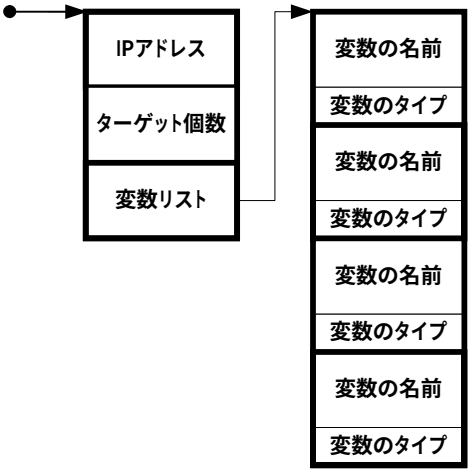
ターゲット

Pro-Designerで作成したアプリケーションファイルをダウンロードして実行するプラットフォームとなるコンピュータシステムを意味します。

設定として必要になるのは以下の 3 つです。

- ターゲットの IP アドレス
- アクセスする変数の個数
- 変数リスト

これらは構造体 ST_DSAPI_TARGET で定義されています。



図表 2-1 ターゲット情報のデータイメージ

変数

データを格納するためのメモリを意味し、整数型、実数型、文字列型、ディスクリート型の4種類に分けられます。

設定として必要になるのは以下の2つです。

- 変数の名称
- 変数のタイプ

これらは構造体 ST_DSAPI_VARIABLE で定義されています。

■環境設定

DSAPIドライバ使用時に、アプリケーションの動作環境上に必要なファイルは以下のようになります。

◆Project.cfg：データシェアリングAPI設定ファイル

詳細については「データシェアリングAPIユーザーズマニュアル」を参照してください。

Pro-Designerをインストールしたフォルダの[\Docs\CFG]フォルダにコンフィギュレーションファイルのサンプル（project.cfg）が存在します。必要に応じて、コピーしてご使用ください。

◆DSAPIDriver.dll：DSAPIドライバ本体

Pro-Designerをインストールしたフォルダの[\Docs\Samples\DSAPI Driver\DLL]フォルダにWin32®用、Windows® CE用のドライバ（DSAPIDriver.dll）が存在します。必要に応じて、パスの通ったフォルダにコピーしてご使用ください。

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がインストールされている必要があります。

注意　Windows CEにおいて、[Storage Card1]以外のフォルダに上記ファイルをコピーする場合は、必ず[マイコンピュータ]－[コントロールパネル]－[バックアップ]を実行してください。

第 3 章　初期化 / オープン

次に、DSAPI ドライバを使用する上で最も複雑な処理を行わねばならない初期化について説明します。

データシェアリング API の初期化に必要な項目として、パス等の環境に関する設定、IP アドレス等のターゲットの情報、実際にデータシェアリングを行う変数名の登録があげられます。複数のターゲットに対してデータシェリングを行う場合、設定用の変数リストに登録される変数は、指定されたターゲットに存在している必要があります。

■ VC++ と VB との相違点

VC++、VB のどちらの環境でユーザープログラムを開発したとしても、基本的に

初期化→オープン→ {リード、ライト} →終了

という DSAPI ドライバの処理手順に変わりはありません。

ただし、VB の言語的な制約により、VB ではユーザー定義型の変数の要素に、C や C++ 言語でいう構造体のポインタを設定することはできないため、DSAPI_Init()でのパラメータデータを作成できません。

したがって、DSAPI_Init()の代わりに、下記の VB 用初期化関数をサポートしました。

```
void  DSAPI_SetTarget(UINT32);
int   DSAPI_AddTarget(ST_DSAPI_TARGET*);
int   DSAPI_AddVariable(UINT32,ST_DSAPI_VARIABLE*);
void  DSAPI_Init_Ex(UCHAR*,UCHAR*,UCHAR*,UCHAR*);
```

■使用手順

① ターゲットの個数を DSAPI_SetTarget()で設定します。

② IP アドレスと変数の個数を設定したターゲット情報を作成します。

③ DSAPI_AddTarget()で、②で作成したターゲット情報を登録します。

この場合、ターゲットは登録した順に 0 から番号が割り振られますので、ユーザーアプリケーションで、その番号を管理する必要があります。

④ 変数の名前とデータタイプを持つ変数情報を作成します。

⑤ DSAPI_AddVariable()で、④で作成した変数情報をターゲット毎に変数個分登録します。

変数の登録順序が、リードやライト系関数を用いる際に使用する Index（ハンドラ）に対応されます。詳しくは Index（ハンドラ）の項目で説明します。

⑥ ターゲットの追加、変数の追加がすべて終了した時点で、DSAPI_Init_Ex()を呼び出して、環境に合わせてパス設定を行います。

⑦ DSAPI_Open()を呼び出して DSAPI をオープンします。

```
'ターゲット数の設定
  Call DSAPI_SetTarget(1)

'ターゲットの登録
  Call DSAPI_AddTarget(astTarget(0))
 ’Call DSAPI_AddTarget(astTarget(1))

'変数の登録
  Call DSAPI_AddVariable(0, astVarList1(0))
  Call DSAPI_AddVariable(0, astVarList1(1))
  Call DSAPI_AddVariable(0, astVarList1(2))
  Call DSAPI_AddVariable(0, astVarList1(3))

 ’Call DSAPI_AddVariable(1, astVarList2(0))
 ’Call DSAPI_AddVariable(1, astVarList2(1))
 ’Call DSAPI_AddVariable(1, astVarList2(2))
 ’Call DSAPI_AddVariable(1, astVarList2(3))

'各種設定項目の初期化（VB用）
  Call DSAPI_Init_Ex(strLibPath, strCurrentPath, strSystemPath,
  strConfigPath)

'DSAPIのオープン
  Call DSAPI_Open
```

■通信を行う変数の登録変更

DSAPI_Openは、一つのプロセスで1回のみ実行できるようになっております。アプリケーション動作中に通信を行う変数の設定を変更したい場合、2回目以降はDSAPI_Open、DSAPI_Closeを使用せず、DSAPI_ConnectとDSAPI_Disconnectを使用することで、設定を変更することができます。

■再設定手順

① DSAPI_Disconnect を行い、現在接続されている接続の切断を行います。

② ターゲットの個数を DSAPI_SetTarget()で設定します。

③ IPアドレスと変数の個数を設定したターゲット情報を作成します。

④ DSAPI_AddTarget()で、③で作成したターゲット情報を登録します。

この場合、ターゲットは登録した順に0から番号が割り振られますので、外部アプリケーションで、その番号を管理する必要があります。

⑤ 変数の名前とデータタイプを持つ変数情報を作成します。

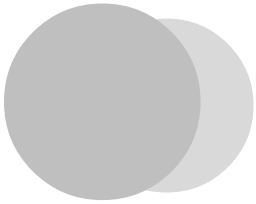
⑥ DSAPI_AddVariable()で、⑤で作成した変数情報をターゲット毎に変数個分登録します。

変数の登録順序が、リードやライト系関数を用いる際に使用する Index（ハンドラ）に対応されます。詳しくは Index（ハンドラ）の項目で説明します。

⑦ ターゲットの追加、変数の追加がすべて終了した時点で、DSAPI_Connect を呼び出して、再接続を行います。

```
'接続の切断
 Call DSAPI_Disconnect

'ターゲット数の設定
 Call DSAPI_SetTarget(1)

'ターゲットの登録
 Call DSAPI_AddTarget(astTarget(0))
'Call DSAPI_AddTarget(astTarget(1))

'変数の登録
 Call DSAPI_AddVariable(0, astVarList1(0))
 Call DSAPI_AddVariable(0, astVarList1(1))
 Call DSAPI_AddVariable(0, astVarList1(2))
 Call DSAPI_AddVariable(0, astVarList1(3))

'Call DSAPI_AddVariable(1, astVarList2(0))
'Call DSAPI_AddVariable(1, astVarList2(1))
'Call DSAPI_AddVariable(1, astVarList2(2))
'Call DSAPI_AddVariable(1, astVarList2(3))

'再接続
 Call DSAPI_Connect
```

注意　　初期化や再設定を行う場合は、必ず手順どおりの順番で行ってください。



第4章　データのリード / ライト

■ Index（変数ハンドラ）

以下に上げたデータの読み書きに使用する関数の第1パラメータには、Index（変数ハンドラ）が必要になります。ここでは、Index（変数ハンドラ）がどのように変数に対応されるかを説明します。

DSAPI_Read()、DSAPI_Write()
DSAPI_ReadInteger()、DSAPI_ReadDiscrete()、DSAPI_ReadFloat()、
DSAPI_ReadString()、DSAPI_WriteInteger()、DSAPI_WriteDiscrete()、
DSAPI_WriteFloat()、DSAPI_WriteString()

初期化の項目でも述べたように、Index（変数ハンドラ）は、初期化時の変数の登録順に依存します。もしターゲットがひとつの場合、単純に、変数の登録順がそのままIndex（変数ハンドラ）の数値に対応され、DSAPIドライバ中で、データシェアリングAPIが返すハンドラにそれぞれ対応されます。外部アプリケーションでは、例えば1番はじめに整数型の変数を登録した場合、その変数に対してデータを書き込みたい時は、

```
void* pvData;
INT32 nData;

nData = 128;
pvData = (void*)&nData;      // 整数型の変数をvoid型に変換
pDSAPI_Write( 0, pvData );  // 0番目の変数に対して書き込み
```

のような記述になります。

複数のターゲットが登録されている場合、次ターゲットの先頭の変数には、前ターゲットの終わりの変数から、連続した数値が対応されます。

■ DSAPI_AddVariable()を使用した場合

DSAPI_Init()ではなくDSAPI_AddVariable()を使用した場合、DSAPI_AddTarget()で登録されたターゲットごとの変数個数ではなく、その変数個数を上限とした、実際にDSAPI_AddVariable()で変数登録された数に対応されます。

基本的に、各々のターゲットで実際に変数がいくつ設定されているかを管理しておけば、各ターゲットの先頭Index（変数ハンドラ）は簡単に算出することができます。

第 5 章　DSAPI ドライバの I/F

DSAPI ドライバで提供されるメソッドは、すべて共通して下記の接頭語がつけられています。

Prefix	Category
DSAPI_	DSAPI Driver

■ Win32 用　DSAPIDriver.dll

```
void  DSAPI_Init(UINT32,ST_DSAPI_PATH,ST_DSAPI_TARGET*);
int   DSAPI_Open();
int   DSAPI_Close();
bool  DSAPI_Read(UINT32,void*,UINT16*);
bool  DSAPI_Write(UINT32,void*);
int   DSAPI_GetError();

int   DSAPI_Connect();
int   DSAPI_Disconnect();
int   DSAPI_Shutdown();

void  DSAPI_SetTarget(UINT32);
int   DSAPI_AddTarget(ST_DSAPI_TARGET_MB*);
int   DSAPI_AddVariable(UINT32,ST_DSAPI_VARIABLE_MB*);
void  DSAPI_Init_Ex(UCHAR*,UCHAR*,UCHAR*,UCHAR*);

int   DSAPI_ReadInteger(UINT32,INT32*);
int   DSAPI_ReadDiscrete(UINT32,UINT16*);
int   DSAPI_ReadFloat(UINT32,float*);
int   DSAPI_ReadString(UINT32,LPTSTR);
int   DSAPI_WriteInteger(UINT32,INT32*);
int   DSAPI_WriteDiscrete(UINT32,UINT16*);
int   DSAPI_WriteFloat(UINT32,float*);
int   DSAPI_WriteString(UINT32,LPCTSTR);
```

Name	Description	Return	Parameters
DSAPI_Init	DSAPIの初期化設定	void	UINT32, ST_DSAPI_PATH, ST_DSAPI_TARGET*
DSAPI_Open	DSAPIに対して初期化を行います	int	なし
DSAPI_Close	DSAPIをクローズします	int	なし
DSAPI_Read	データを読み出します	bool	UINT32, void*, UINT16*
DSAPI_Write	データを書き込みます	bool	UINT32, void*
DSAPI_GetError	エラーを取得します	int	なし
DSAPI_Connect	変数の接続を行います	int	なし
DSAPI_Disconnect	接続を切断します	int	なし
DSAPI_Shutdown	DSAPIを消去します	int	なし
DSAPI_SetTarget	ターゲット個数を設定します	void	UINT32
DSAPI_AddTarget	ターゲットの属性を登録します	int	ST_DSAPI_TARGET_MB*
DSAPI_AddVariable	変数の属性を登録します	int	UINT32, ST_DSAPI_VARIABLE_MB*
DSAPI_Init_Ex	VB用初期化関数	void	UCHAR*, UCHAR*, UCHAR*, UCHAR*
DSAPI_ReadInteger	VB用整数型変数読み出し	int	UINT32, INT32*
DSAPI_ReadDiscrete	VB用ディスクリート変数読み出し	int	UINT32, UINT16*
DSAPI_ReadFloat	VB用実数型変数読み出し	int	UINT32, float*
DSAPI_ReadString	VB用文字列型変数読み出し	int	UINT32, LPTSTR
DSAPI_WriteInteger	VB用整数型変数書き込み	int	UINT32, INT32*
DSAPI_WriteDiscrete	VB用ディスクリート変数書き込み	int	UINT32, UINT16*
DSAPI_WriteFloat	VB用実数型変数書き込み	int	UINT32, float*
DSAPI_ WriteString	VB用文字列型変数書き込み	int	UINT32, LPCTSTR

Name	:	**DSAPI_Init**
Category	:	**Win32**
Parameters	:	UINT32 nTargetCount, // ターゲット数を指定します ST_DSAPI_PATH stPath, // 初期化に必要な設定の構造体 ST_DSAPI_TARGET* pstTargetList // ターゲット情報リストのポインタ
Return	:	**void**
Remarks	:	DSAPIを使用する為の各種設定に対して値を登録します。 ここでは、ドライバの内部に設定値を登録するだけですので、使用手順としては、次にDSAPI_Open()を呼び出す必要があります。

ST_DSAPI_PATH

```
UNICHAR*  puncLibPath  // RuntimeAdapter.dllへのパス
UNICHAR*  puncCurrentPath  // カレント実行デイレクトリのパス
UNICHAR*  puncSystemPath  // システムデイレクトリのパス
UNICHAR*  puncConfigPath  // コンフィギュレーションファイルへのパス
```

例　C:\Program Files\pro-face\Docs\CFG\project.cfg

ST_DSAPI_VARIABLE

```
UNICHAR  auncVarName[60]  // 変数の名前
BYTE  byVarType  // 変数のタイプ
```

```
VAR_TYPE_INT = 0,
VAR_TYPE_FLOAT = 1,
VAR_TYPE_STRING = 2,
VAR_TYPE_DISCRETE = 3
```

ST_DSAPI_TARGET

```
UNICHAR*  puncIPAddress  // ターゲットマシンのIPアドレス
UINT32  nVariableCount  // 変数の個数
ST_DSAPI_VARIABLE*  pstVariable  // 変数のリストのポインタ
```

Name	:	**DSAPI_Open**
Category	:	**Win32**
Parameters	:	なし
Return	:	**int**
Remarks	:	**返り値** 成功した場合：true 失敗した場合：false

DSAPIのライブラリをメモリに展開し、各メソッドのアドレスを取得します。

また実際にシェアリングを行う変数の登録や、ドライバ内部では、データ更新時にコールバックされる関数の登録もここで行います。

これらの登録に必要な変数のリスト等は、この関数の呼び出し時には必ず必要になりますので、注意事項として、DSAPI_Open()を使用する前には、必ずDSAPI_Init()を呼び出して必要事項を設定しておいてください。

Name	:	**DSAPI_Close**
Category	:	**Win32**
Parameters	:	なし
Return	:	**int**
Remarks	:	**返り値** 成功した場合：true 失敗した場合：false データシェアリング機能をクローズします。 メモリに展開されていたライブラリは開放されます。

Name	:	**DSAPI_Read**
Category	:	**Win32**
Parameters	:	UINT32 nIndex, // データリストのハンドラ void* pvData, // データを格納したポインタを入れて返すための器です UINT16* pnDataType // データタイプを返します
Return	:	**bool**
Remarks	:	**返り値** データ更新された場合：true データ更新されていない場合：false **nIndex** 変数リストで何番目のデータが欲しいかを指定してください。 **pvData** **pnDataType** データ更新されていた場合は、ここにデータが格納されているポインタを入れて返しますので、同時にpnDataTypeに入っているデータタイプを判別してデータを扱ってください。

Name	:	**DSAPI_Write**
Category	:	**Win32**
Parameters	:	UINT32 nIndex, // データリストのハンドラ void* pvData // 書き込みデータを格納している器のポインタ
Return	:	**bool**
Remarks	:	**返り値** 成功した場合：true 失敗した場合：false ハンドラを指定して、それに対応する変数にデータを書き込みます。

Name	:	**DSAPI_GetError**
Category	:	**Win32**
Parameters	:	なし
Return	:	**int**
Remarks	:	この関数が呼び出されたときに持っているエラー情報を返します。 エラー情報は、その発生ごとに上書きされますので、常に最新のエラーが返されます。 RTA_CONNECTING = 0, RTA_CONNECTED = 1, RTA_TAGNAME_ERROR = 2, RTA_TOO_MANY_TAGS_ERROR = 3, RTA_VERSION_ERROR = 4

Name	:	**DSAPI_Connect**
Category	:	**Win32**
Parameters	:	なし
Return	:	**int**
Remarks	:	**返り値** 成功した場合：true 失敗した場合：false 現在設定されているターゲットや変数の情報で、DSAPIに接続を行います。

Name	:	**DSAPI_Disconnect**
Category	:	**Win32**
Parameters	:	なし
Return	:	**int**
Remarks	:	**返り値** 成功した場合：true 失敗した場合：false DSAPIとの接続を切断し、ターゲットや変数の情報をクリアします。 DSAPI_Connect()で再接続を行う場合は、必ずターゲットや変数の設定をDSAPI_SetTarget(),DSAPI_AddTarget(),DSAPI_AddVariable()で事前に行う必要があります。

Name	:	**DSAPI_Shutdown()**
Category	:	**Win32**
Parameters	:	なし
Return	:	**int**
Remarks	:	**返り値** 成功した場合：true 失敗した場合：false DSAPIそのものを消去します。通常は、DSAPI_Close()で終了処理を行いますが、DSAPI_Disconnect()で接続が切断されている状態から終了処理を行うときに、この関数を使用します。

Name	:	**DSAPI_SetTarget**
Category	:	**Win32**
Parameters	:	UINT32 nTargetCount　　//登録するターゲットの個数
Return	:	void
Remarks	:	DSAPI_Init_Ex()を用いる際、ターゲット個数の登録に使用します。

Name	:	**DSAPI_AddTarget**
Category	:	**Win32**
Parameters	:	ST_DSAPI_TARGET_MB* pstTargetList // ターゲット情報リストのポインタ
Return	:	int
Remarks	:	**返り値** 成功した場合：true (not 0) 失敗した場合：false (0) IPアドレスや変数の個数といったターゲット情報を登録します。

Name	:	**DSAPI_AddVariable**
Category	:	**Win32**
Parameters	:	UINT32 nTargetNum, // ターゲット番号を指定します ST_DSAPI_VARIABLE_MB* pstVariable // 変数のポインタ
Return	:	int
Remarks	:	**返り値** 成功した場合：true (not 0) 失敗した場合：false (0) 変数の名前と型を、指定したターゲットの変数情報として登録します。

Name	:	**DSAPI_Init_Ex**
Category	:	**Win32**
Parameters	:	UCHAR* pucLibPath, // RuntimeAdapter.dllへのパス UCHAR* pucCurrentPath, // カレント実行デイレクトリのパス UCHAR* pucSystemPath, // システムデイレクトリのパス UCHAR* pucConfigPath // コンフィギュレーションファイルへのパス
Return	:	void
Remarks	:	VBユーザー等がDSAPI_Init()を使用しない場合、DSAPI_SetTarget()、DSAPI_AddTarget()、DSAPI_AddVariable()とあわせてDSAPI_Open()に必要な情報をDSAPI Driverに登録することに使用します。

Name	:	DSAPI_ReadInteger
Category	:	Win32
Parameters	:	UINT32 nIndex, // データリストのハンドラ INT32* pnData // データを格納したポインタを入れて返すための器です
Return	:	int
Remarks	:	**返り値** データ更新された場合：true データ更新されていない場合：false 整数型変数のデータ読み込み時に使用します。

Name	:	DSAPI_ReadDiscrete
Category	:	Win32
Parameters	:	UINT32 nIndex, // データリストのハンドラ UINT16* psData // データを格納したポインタを入れて返すための器です
Return	:	int
Remarks	:	**返り値** データ更新された場合：true データ更新されていない場合：false ディスクリート型変数のデータ読み込み時に使用します。

Name	:	DSAPI_ReadFloat
Category	:	Win32
Parameters	:	UINT32 nIndex, // データリストのハンドラ float* pfData // データを格納したポインタを入れて返すための器です
Return	:	int
Remarks	:	**返り値** データ更新された場合：true データ更新されていない場合：false 実数型変数のデータ読み込み時に使用します。

Name	:	DSAPI_ReadString
Category	:	Win32
Parameters	:	UINT32 nIndex, // データリストのハンドラ LPTSTR pucData // データを格納したポインタを入れて返すための器です
Return	:	int
Remarks	:	**返り値** データ更新された場合：true データ更新されていない場合：false 文字列型変数のデータ読み込み時に使用します。

Name	:	**DSAPI_WriteInteger**
Category	:	**Win32**
Parameters	:	UINT32 nIndex, // データリストのハンドラ INT32* pnData // 32bitの型データのポインタ
Return	:	int
Remarks	:	**返り値** 成功した場合：true 失敗した場合：false 整数型変数のデータ書き込み時に使用します。

Name	:	**DSAPI_WriteDiscrete**
Category	:	**Win32**
Parameters	:	UINT32 nIndex, // データリストのハンドラ UINT16* psData // ブール型データのポインタ
Return	:	int
Remarks	:	**返り値** 成功した場合：true 失敗した場合：false ディスクリート型変数のデータ書き込み時に使用します。

Name	:	**DSAPI_WriteFloat**
Category	:	**Win32**
Parameters	:	UINT32 nIndex, // データリストのハンドラ float* pfData // フロート型データのポインタ
Return	:	int
Remarks	:	**返り値** 成功した場合：true 失敗した場合：false 実数型変数のデータ書き込み時に使用します。

Name	:	**DSAPI_WriteString**
Category	:	**Win32**
Parameters	:	UINT32 nIndex, // データリストのハンドラ LPCTSTR pucData // 文字列型データのポインタ
Return	:	int
Remarks	:	**返り値** 成功した場合：true 失敗した場合：false 文字列型変数のデータ書き込み時に使用します。

■ WinCE 用　DSAPIDriver.dll

```
void  DSAPI_Init(UINT32,ST_DSAPI_PATH,ST_DSAPI_TARGET*);
int   DSAPI_Open();
int   DSAPI_Close();
bool  DSAPI_Read(UINT32,void*,UINT16*);
bool  DSAPI_Write(UINT32,void*);
int   DSAPI_GetError();

void  DSAPI_SetTarget(UINT32);
int   DSAPI_AddTarget(UNICHAR*,UINT32);
int   DSAPI_AddVariable(UINT32,UNICHAR*,BYTE);
void  DSAPI_Init_Ex(UNICHAR*,UNICHAR*,UNICHAR*,UNICHAR*);

int   DSAPI_ReadInteger(UINT32,INT32*);
int   DSAPI_ReadDiscrete(UINT32,UINT16*);
int   DSAPI_ReadFloat(UINT32,float*);
int   DSAPI_ReadString(UINT32,LPTSTR);
int   DSAPI_WriteInteger(UINT32,INT32*);
int   DSAPI_WriteDiscrete(UINT32,UINT16*);
int   DSAPI_WriteFloat(UINT32,float*);
int   DSAPI_WriteString(UINT32,LPCTSTR);
```

Name	Description	Return	Parameters
DSAPI_Init	DSAPIの初期化設定	void	UINT32, ST_DSAPI_PATH, ST_DSAPI_TARGET*
DSAPI_Open	DSAPIに対して初期化を行います	int	なし
DSAPI_Close	DSAPIをクローズします	int	なし
DSAPI_Read	データを読み出します	bool	UINT32, void*, UINT16*
DSAPI_Write	データを書き込みます	bool	UINT32, void*
DSAPI_ GetError	エラーを取得します	int	なし
DSAPI_ SetTarget	ターゲット個数を設定します	void	UINT32
DSAPI_ AddTarget	ターゲットの属性を登録します	int	UNICHAR*, UINT32
DSAPI_ AddVariable	変数の属性を登録します	int	UINT32, UNICHAR*, BYTE
DSAPI_ Init_Ex	VB用初期化関数	void	UNICHAR*, UNICHAR*, UNICHAR*, UNICHAR*
DSAPI_ ReadInteger	VB用整数型変数読み出し	int	UINT32, INT32*
DSAPI_ ReadDiscrete	VB用ディスクリート変数読み出し	int	UINT32, UINT16*
DSAPI_ ReadFloat	VB用実数型変数読み出し	int	UINT32, float*
DSAPI_ ReadString	VB用文字列型変数読み出し	int	UINT32, LPTSTR
DSAPI_ WriteInteger	VB用整数型変数書き込み	int	UINT32, INT32*
DSAPI_ WriteDiscrete	VB用ディスクリート変数書き込み	int	UINT32, UINT16*
DSAPI_ WriteFloat	VB用実数型変数書き込み	int	UINT32, float*
DSAPI_ WriteString	VB用文字列型変数書き込み	int	UINT32, LPCTSTR

Name	:	**DSAPI_Init**
Category	:	**WinCE**
Parameters	:	UINT32 nTargetCount, // ターゲット数を指定します ST_DSAPI_PATH stPath, // 初期化に必要な設定の構造体 ST_DSAPI_TARGET* pstTargetList // ターゲット情報リストのポインタ
Return	:	**void**
Remarks	:	DSAPIを使用する為の各種設定に対して値を登録します。 ここでは、ドライバの内部に設定値を登録するだけですので、使用手順としましては、次にDSAPI_Open()を呼び出す必要があります。

ST_DSAPI_PATH

```
UNICHAR*  puncLibPath  // RuntimeAdapter.dllへのパス
UNICHAR*  puncCurrentPath  // カレント実行デイレクトリのパス
UNICHAR*  puncSystemPath  // システムデイレクトリのパス
UNICHAR*  puncConfigPath  // コンフィギュレーションファイルへのパス
```

例　Storage Card1/project.cfg

ST_DSAPI_VARIABLE

```
UNICHAR  auncVarName[60]  // 変数の名前
BYTE  byVarType  // 変数のタイプ
```

```
VAR_TYPE_INT = 0,
VAR_TYPE_FLOAT = 1,
VAR_TYPE_STRING = 2,
VAR_TYPE_DISCRETE = 3
```

ST_DSAPI_TARGET

```
UNICHAR*  puncIPAddress  // ターゲットマシンのIPアドレス
UINT32  nVariableCount  // 変数の個数
ST_DSAPI_VARIABLE*  pstVariable  // 変数のリストのポインタ
```

Name	:	**DSAPI_Open**
Category	:	**WinCE**
Parameters	:	なし
Return	:	**int**
Remarks	:	**返り値** 成功した場合：true 失敗した場合：false

データシェアリングAPIのライブラリをメモリに展開し、各メソッドのアドレスを取得します。

また実際にシェアリングを行う変数の登録やデータ更新時にコールバックされる関数の登録もここで行います。

これらの登録に必要な変数のリスト等は、この関数の呼び出し時に必要になりますので、注意事項として、DSAPI_Open()を使用する前には、必ずDSAPI_Init()を呼び出して必要事項を設定しておいてください。

Name	:	**DSAPI_Close**
Category	:	**WinCE**
Parameters	:	なし
Return	:	**int**
Remarks	:	**返り値** 成功した場合：true 失敗した場合：false データシェアリング機能をクローズします。 メモリに展開されていたライブラリは開放されます。

Name	:	**DSAPI_Read**
Category	:	**WinCE**
Parameters	:	UINT32 nIndex, // データリストの何番目かを示します void* pvData, // データを格納したポインタを入れて返すための器です UINT16* pnDataType // データタイプを返します
Return	:	**bool**
Remarks	:	**返り値** データ更新された場合：true データ更新されていない場合：false **nIndex** 変数リストで何番目のデータが欲しいかを指定してください。 **pvData** **pnDataType** データ更新されていた場合は、ここにデータが格納されているポインタを入れて返しますので、同時にpnDataTypeに入っているデータタイプを判別してデータを扱ってください。

Name	:	**DSAPI_Write**
Category	:	**WinCE**
Parameters	:	UINT32 nIndex, // データリストのハンドラ void* pvData // 何の型かわからないデータのポインタ
Return	:	**bool**
Remarks	:	**返り値** 成功した場合：true 失敗した場合：false ハンドラを指定して、それに対応する変数にデータを書き込みます

Name	:	**DSAPI_GetError**
Category	:	**WinCE**
Parameters	:	なし
Return	:	int
Remarks	:	この関数が呼び出されたときに持っているエラー情報を返します。 エラー情報は、その発生ごとに上書きされますので、常に最新のエラーが返されます。 RTA_CONNECTING = 0, RTA_CONNECTED = 1, RTA_TAGNAME_ERROR = 2, RTA_TOO_MANY_TAGS_ERROR = 3, RTA_VERSION_ERROR = 4

Name	:	**DSAPI_SetTarget**
Category	:	**WinCE**
Parameters	:	登録するターゲットの個数
Return	:	void
Remarks	:	DSAPI_Init_Ex()を用いる際、ターゲット個数の登録に使用します。

Name	:	**DSAPI_AddTarget**
Category	:	**WinCE**
Parameters	:	UNICHAR* pucIPAddress, // ターゲットマシンのIPアドレス UINT32 nVariableCount // 変数の個数
Return	:	int
Remarks	:	**返り値** 成功した場合：true (not 0) 失敗した場合：false (0) IPアドレスや変数の個数といったターゲット情報を登録します。

Name	:	**DSAPI_AddVariable**
Category	:	**WinCE**
Parameters	:	UINT32 nTargetNum, // ターゲット番号を指定します UNICHAR aucVarName[], // 変数の名前 BYTE byVarType // 変数のタイプ
Return	:	int
Remarks	:	**返り値** 成功した場合：true (not 0) 失敗した場合：false (0) 変数の名前と型を、指定したターゲットの変数情報として登録します。

Name	:	**DSAPI_Init_Ex**
Category	:	**WinCE**
Parameters	:	UNICHAR* pucLibPath, // ライブラリファイルのパス UNICHAR* pucCurrentPath, // カレント実行デイレクトリのパス UNICHAR* pucSystemPath, // システムデイレクトリのパス UNICHAR* pucConfigPath // コンフィギュレーションファイルへのパス
Return	:	void
Remarks	:	VBユーザー等がDSAPI_Init()を使用しない場合、DSAPI_SetTarget()、DSAPI_AddTarget()、DSAPI_AddVariable()とあわせてDSAPI_Open()に必要な情報をDSAPI Driverに登録するのに使用します。

Name	:	**DSAPI_ReadInteger**
Category	:	**WinCE**
Parameters	:	UINT32 nIndex, // データリストのハンドラ INT32* pnData // データを格納したポインタを入れて返すための器です
Return	:	int
Remarks	:	**返り値** データ更新された場合：true データ更新されていない場合：false 整数型変数のデータ読み込み時に使用します。

Name	:	**DSAPI_ReadDiscrete**
Category	:	**WinCE**
Parameters	:	UINT32 nIndex, // データリストのハンドラ UINT16* psData // データを格納したポインタを入れて返すための器です
Return	:	int
Remarks	:	**返り値** データ更新された場合：true データ更新されていない場合：false ディスクリート型変数のデータ読み込み時に使用します。

Name	:	**DSAPI_ReadFloat**
Category	:	**WinCE**
Parameters	:	UINT32 nIndex, // データリストのハンドラ float* pfData // データを格納したポインタを入れて返すための器です
Return	:	int
Remarks	:	**返り値** データ更新された場合：true データ更新されていない場合：false 実数型変数のデータ読み込み時に使用します。

Name	:	DSAPI_ReadString
Category	:	WinCE
Parameters	:	UINT32 nIndex, // データリストのハンドラ LPTSTR pucData // データを格納したポインタを入れて返すための器です
Return	:	int
Remarks	:	**返り値** データ更新された場合：true データ更新されていない場合：false 文字列型変数のデータ読み込み時に使用します。

Name	:	DSAPI_WriteInteger
Category	:	WinCE
Parameters	:	UINT32 nIndex, // データリストのハンドラ INT32* pnData // 32bitの型データのポインタ
Return	:	int
Remarks	:	**返り値** 成功した場合：true 失敗した場合：false 整数型変数のデータ書き込み時に使用します。

Name	:	DSAPI_WriteDiscrete
Category	:	WinCE
Parameters	:	UINT32 nIndex, // データリストのハンドラ UINT16* psData // ブール型データのポインタ
Return	:	int
Remarks	:	**返り値** 成功した場合：true 失敗した場合：false ディスクリート型変数のデータ書き込み時に使用します。

Name	:	DSAPI_WriteFloat
Category	:	WinCE
Parameters	:	UINT32 nIndex, // データリストのハンドラ float* pfData // フロート型データのポインタ
Return	:	int
Remarks	:	**返り値** 成功した場合：true 失敗した場合：false 実数型変数のデータ書き込み時に使用します。

Name	:	**DSAPI_WriteString**
Category	:	**WinCE**
Parameters	:	UINT32 nIndex, // データリストのハンドラ LPCTSTR pucData // 文字列型データのポインタ
Return	:	int
Remarks	:	**返り値** 成功した場合：true 失敗した場合：false 文字列型変数のデータ書き込み時に使用します。

第6章　各種制限事項

DSAPIドライバはデータシェアリングAPIを使用していますので、仕様的な制限事項はDSAPIに依存します。ここでは主な項目について記述しますので、その詳細は「データシェアリングAPIユーザーズマニュアル」を参照してください。

- 日本語を含む環境設定のパスは使用できません。
- 同時に接続できるターゲット数は最大16台です。
- データシェリングでき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の目安は以下のとおりです。

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

  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。

- VBでこのデータシェアリングAPIドライバを使用する場合、VBの言語仕様でサポートされていない型への対応はできません。(符号なし整数等)