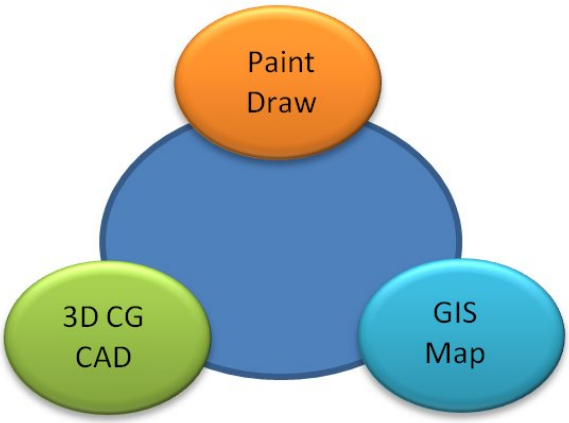
DelPlotter



Version 1.9.0

プラグインＳＤＫガイド

目次

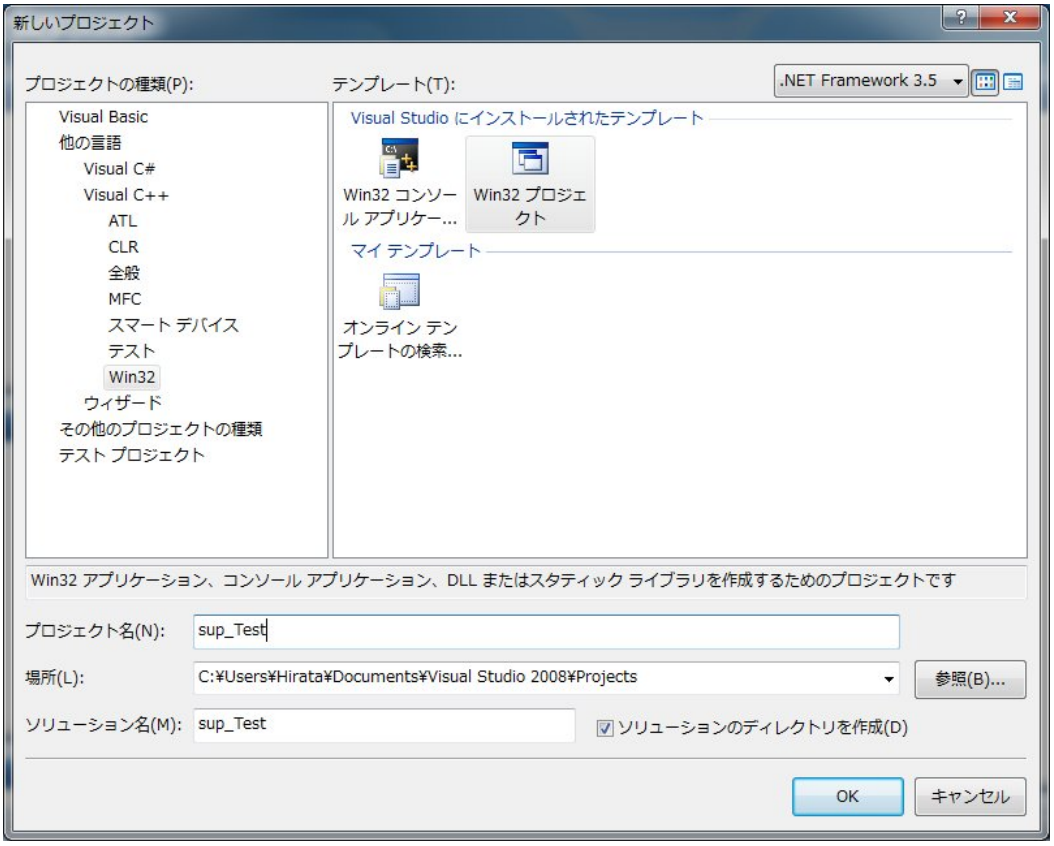
©2003-2012. Table Soft Presents.

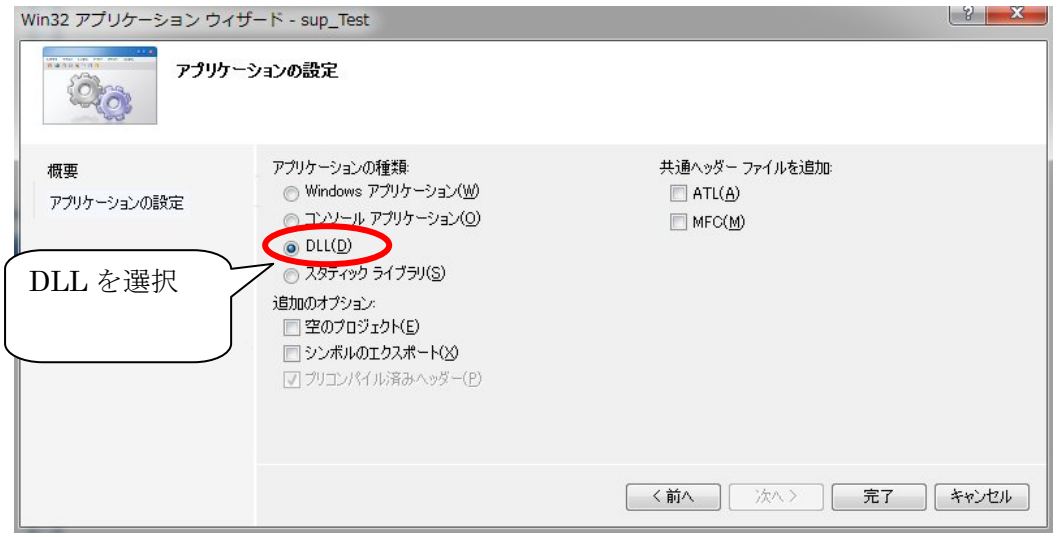
プラグイン作成方法

VisualStudio 2005、2008 など

プロジェクト作成

新しいプロジェクト - Win32 プロジェクトを選択します。





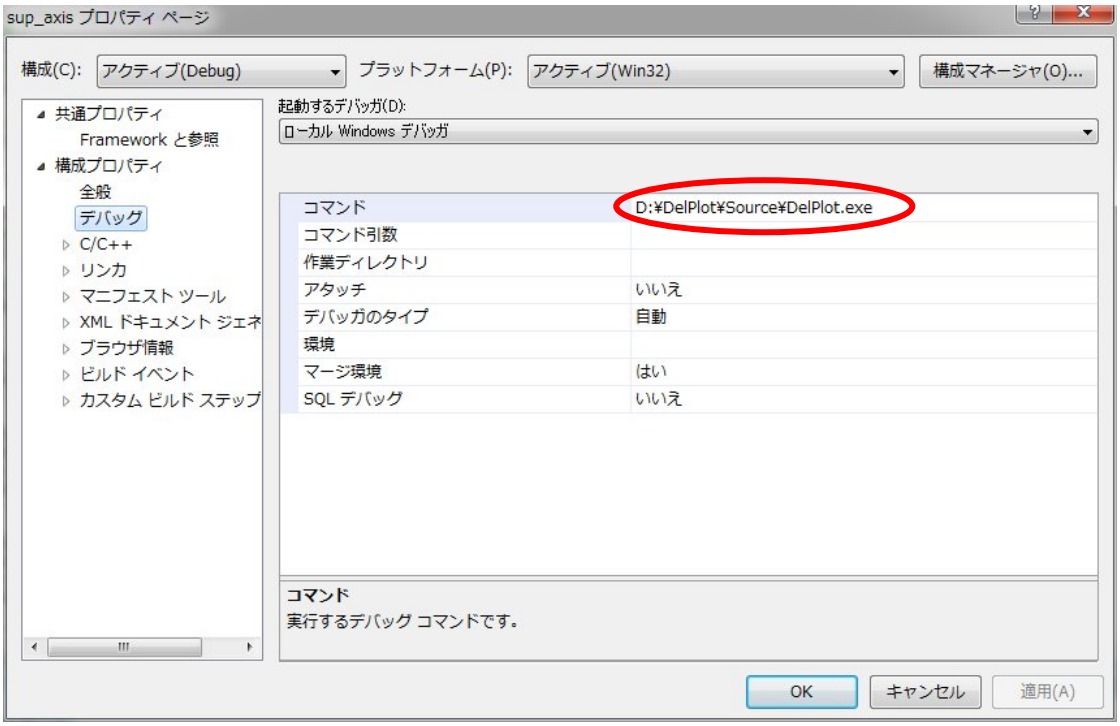
これでプロジェクトが作成されます。

プロジェクト – 既存の項目を追加 にて DelplotPlugin.cpp、DelplotPlugin.h の２ファイルを追加します。

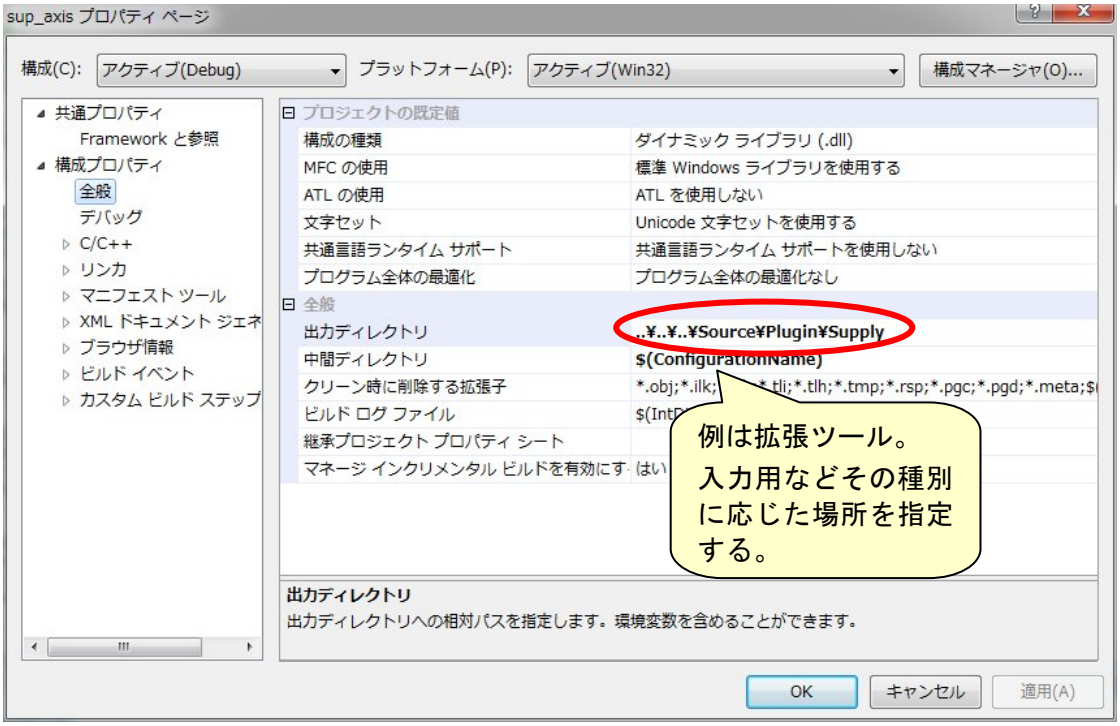
プロジェクトのプロパティ設定

● デバッグ用の設定（設定画面は一例です）

プラグインのデバッグを行う際の DelPlot のあるフォルダを指定します。

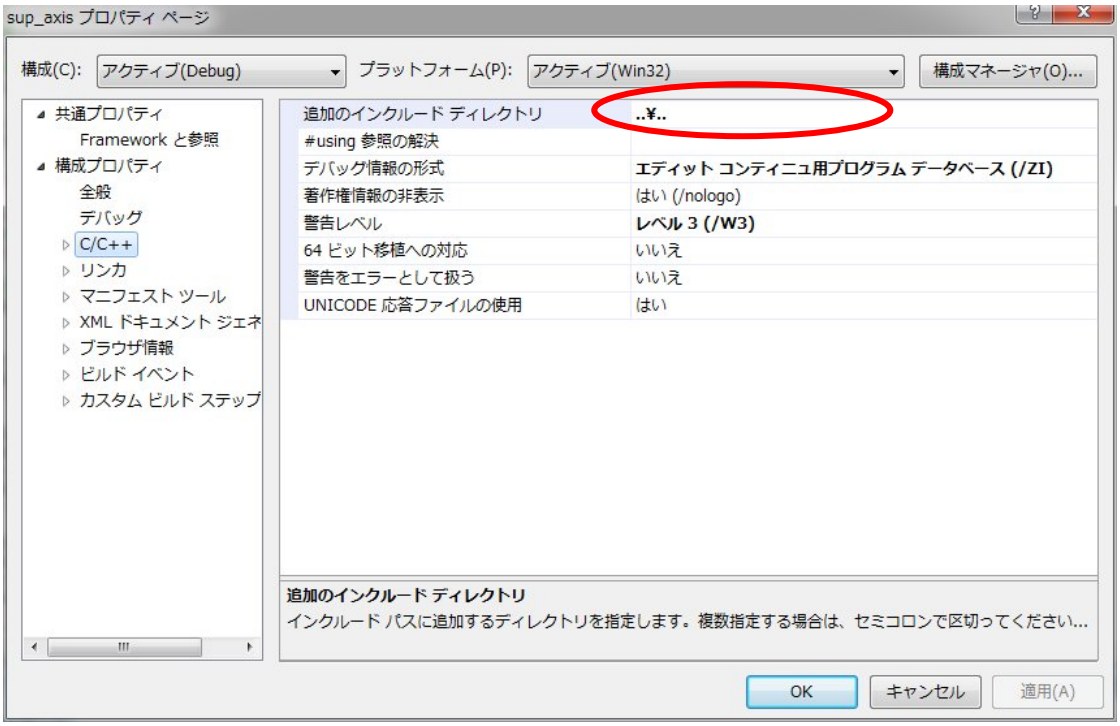


プラグインの出力先を DelPlot が読込みするフォルダに指定します。



● DelplotPlugin.cpp のフォルダの設定

DelplotPlugin.cpp、DelplotPlugin.h がプラグイン作成フォルダにない場合は、その存在するフォルダを指定します。



● プラグイン接続情報の設定

サンプルの中から適当なものをひな形としてコピー貼り付けます。

例では「sup_dialog」を説明に使用します。

uses 句に　DelplotPlugin in '..¥DelplotPlugin.pas',（DelplotPlugin.pas のフォルダ位置により適宜指定して下さい）があることを確認してください。なければ追加します。

void dpDocument::SetDllInfomation() の中の設定項目に作成するプラグインの情報を設定します。

```
//---------------------------------------------------------------------------
// sup_dialog.cpp                                          Hirata.o
//
// ダイアログ表示サンプル（一番単純なプラグイン）
//

#include "stdafx.h"
#include "DelplotPlugin.h"


//---------------------------------------------------------------------------
// Plugin インターフェース
//---------------------------------------------------------------------------
// Pluginの起動処理（任意）
void dpDocument::PluginSynchronize(){return;}
// Pluginの解放処理（任意）
void dpDocument::PluginPurge(){return;}
// Plugin情報の設定（必須）
void dpDocument::SetDllInfomation()
{
    // Plugin Usage Type
    Style = DPPLUGIN_STYLE_SUPPLY;
    // Plugin GUID（ツール(T) - GUIDの作成 で生成可能）
    Guid  = L"{BCE17F2B-9F90-439c-A811-7CC28B030B47}";
    // Plugin 固有名称
    Name  = L"Message Dialog";
    // Plugin ListCaption 表示名
    Caption = L"MsgDialog";
}
// Pluginオプション項目の設定（任意）
void dpDocument::PluginOption()
{
    return;
}

// Pluginの開始処理（任意）
void dpDocument::PluginInitialize(){return;}
// Pluginの終了処理（任意）
void dpDocument::PluginTerminate(){return;}
// Pluginの実行（必須）
void dpDocument::PluginExecute(long EventNo)
{
    MessageBox(NULL,(LPCTSTR)"OK!",(LPCTSTR)"MessageDialg",MB_OK);
    return;
}
```

Style： プラグインの機能種別を指定します。 ※ 詳細は「プラグイン規約」を参照
Guid： 作成するプラグインの固有 ID を指定するため GUID を設定します。 ツール　-　GUID の作成 が利用できます。 もし、「GUID の作成」が無い場合（2008 など）は「外部ツール(E)...」から

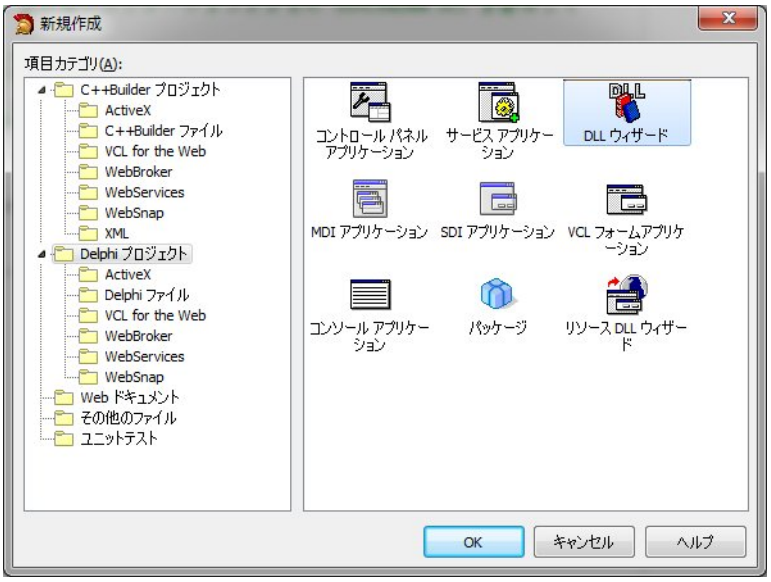
…（C:¥Program Files 等）…¥Microsoft Visual Studio 8¥Common7¥Tools >guidgen.exe を選択（参照 http://msdn.microsoft.com/ja-jp/library/76712d27(v=vs.80).aspx） メニューに外部ツールを登録できます。
Name： プラグインをデータから呼び出す際に使用する名称を指定します。データファイル（SYSTEM_CONTROL 命令など）もしくはユーザー定義のメニュー・ボタンにこのプラグイン登録する時にはこの名前が使用されます。
Caption： プラグイン一覧などに表示する簡易名称を指定します。名称が長いと表示しきれないなどは起こりますが、上記 Name と同一でも問題ありません。

あとはサンプルソースコードなどを参考にプラグインの機能を作成します。

RAD Studio 2009

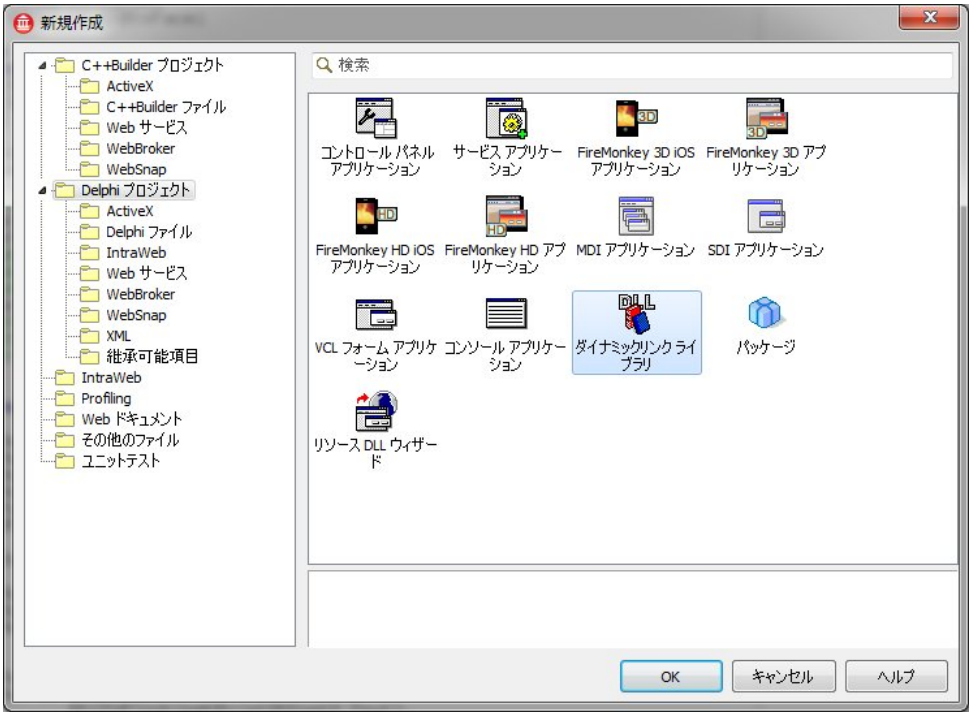
プロジェクト作成

新規作成 – DLL ウイザードを選択します。



RAD Studio EX2

新規作成 – ダイナミックリンクライブラリを選択します。



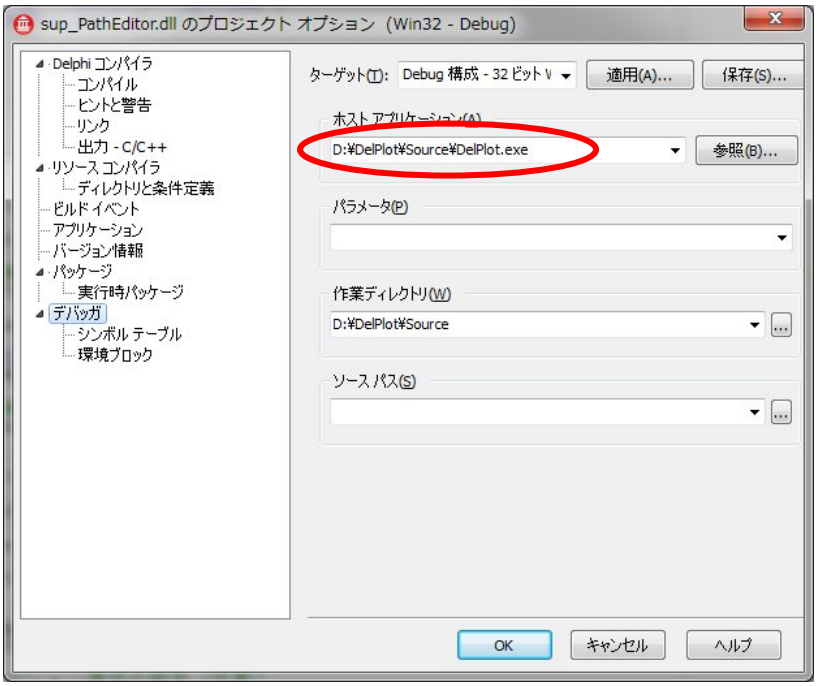
注意）RAD Studio 2009 以前の Delphi で Win32 アンマネージメント Dll を作成される場合、Record 構造にある class operator 指定などの構文が実装されていない時代のバージョンがあります。

また、WindowsVista 対応は RAD Studio 2009 から保障されています。ご注意ください。

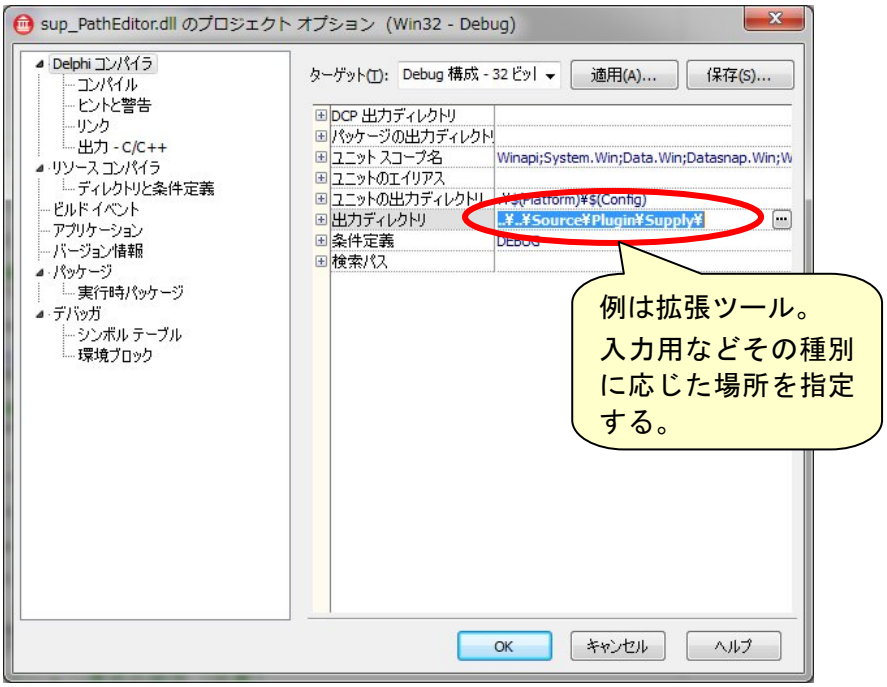
プロジェクトのプロパティ設定

● デバッグ用の設定（設定画面は一例です）

プラグインのデバッグを行う際の DelPlot のあるフォルダを指定します。



プラグインの出力先を DelPlot が読込みするフォルダに指定します。



● プラグイン接続情報の設定

サンプルの中から適当なものをひな形としてコピー貼り付けます。

例では「sup_point picker」を説明に使用します。

uses 句に　DelplotPlugin in '..¥DelplotPlugin.pas',（DelplotPlugin.pas のフォルダ位置により適宜指定して下さい）があることを確認してください。なければ追加します。

procedure SetDllInfomation(); **cdecl;** の中の設定項目に作成するプラグインの情報を設定します。

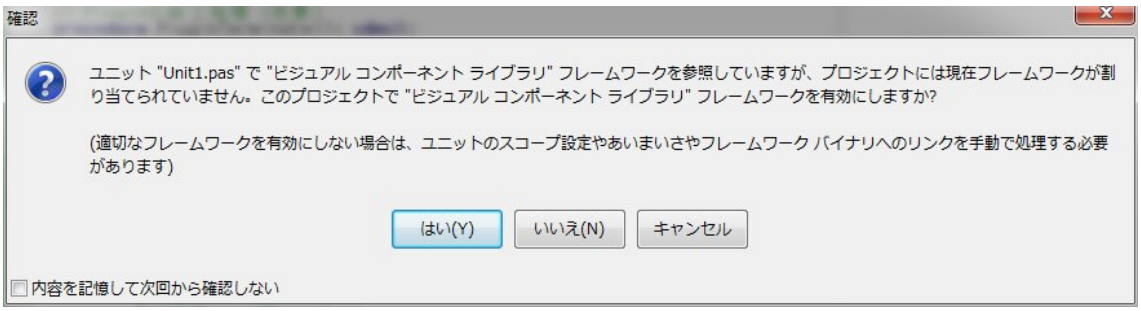
```
uses
  System.SysUtils,
  System.Classes,
  DelplotPlugin in '..¥DelplotPlugin.pas';

{$R *.res}

var
    PointStr: String;

    // 返り値受け取り処理
    procedure ReturnPlace(Points: PDouble);
    var
      P  : TDoubleArray;
    begin
      P := TDoubleArray(Points);
      PointStr := FloatToStr(P[0])
            +','+FloatToStr(P[1]);
      dpDoc.Tbl.PlotText := PChar(PointStr);
    end;

//---------------------------------------------------------------------------
// Plugin インターフェース
//---------------------------------------------------------------------------
    // Pluginの起動処理 (任意)
    procedure PluginSynchronize(); cdecl;
    begin
      PointStr := '';
    end;
    // Pluginの解放処理 (任意)
    procedure PluginPurge(); cdecl;
    begin
      ;
    end;
    // Plugin情報の設定 (必須)
    procedure SetDllInfomation(); cdecl;
    begin
      // Plugin Usage Type
      dpDoc.Style := DPPLUGIN_STYLE_SUPPLY;
      // Plugin GUID (Shift+Ctrl+G で生成可能)
      dpDoc.Guid  := '{4F15B29B-AC68-4B3F-848C-440DF7CD459E}';
      // Plugin 固有名称
      dpDoc.Name  := 'PointPicker';
      // Plugin ListCaption 表示名
      dpDoc.Caption := 'PointPicker';

    end;
    // Pluginオプション項目の設定 (任意)
    procedure PluginOption(); cdecl;
    begin
      SetLength(dpDoc.Option, 1);
      dpDoc.Option[0].Comnd := DPPLUGIN_MENU_TEXTBOX;
      dpDoc.Option[0].Title := 'コメント:';
      dpDoc.Option[0].Value := 'Delplot本体の画面座標を取り込み、結果をCSV生成、閲覧可能にします。';
      dpDoc.Option[0].Enable  := DPPLUGIN_MENU_FALSE;
      dpDoc.Option[0].Visible := DPPLUGIN_MENU_TRUE;
    end;
    // Pluginの開始処理 (任意)
    procedure PluginInitilaize(); cdecl;
    begin

    end;
    // Pluginの終了処理 (任意)
    procedure PluginTerminate(); cdecl;
    begin

    end;
    // Pluginの実行 (必須)
    procedure PluginExecute(EventNo: Integer); cdecl;
    begin

      // ViewWindow Pointing demand
      if Assigned(dpDoc.Tbl.dpPointAction) then
        dpDoc.Tbl.dpPointAction(dpDoc.Tbl.SerialID, 1, @ReturnPlace);

    end;

begin
```

Style： 　プラグインの機能種別を指定します。 　※ 詳細は「プラグイン規約」を参照
Guid： 　作成するプラグインの固有 ID を指定するため GUID を設定します。 　Shift+Ctrl+G で生成可能です。

Name : プラグインをデータから呼び出す際に使用する名称を指定します。データファイル（SYSTEM_CONTROL 命令など）もしくはユーザー定義のメニュー・ボタンにこのプラグインを登録する時にはこの名前が使用されます。
Caption : プラグイン一覧などに表示する簡易名称を指定します。名称が長いと表示しきれないなどは起こりますが、上記 Name と同一でも問題ありません。

あとはサンプルソースコードなどを参考にプラグインの機能を作成します。

プラグインにフォームを追加する場合、Dll にフォームを追加すると下記のダイアログが表示されます。



ライブラリの追加設定を要求されますので、「はい」を選択します。

DelPlot プラグイン規約

全体規約

- プラグインを起動時には dpDocument クラスのグローバル変数 dpDoc を作成し、Delplot からの情報を受け取る処理を既定の手順に基づき行う事を必要とする。
  （サンプルをひな形として作成する場合は気にする必要がありません）
- プラグインを Delplot で利用可能にするため以下の 7 つの処理を必要とする。
  C++は dpDocument クラスのメソッドとして実装し、Delphi ではプロジェクト ソースファイルに Procedure として実装します。具体的な追加方法はサンプルを参考にして下さい。

PluginSynchronize：**プラグインの起動処理 （任意）**

Delplotがプラグインを読込み時の処理を指定できます。
任意で指定します。

PluginPurge：**プラグインの解放処理（任意）**

Delplotがプラグインを解放時の処理を指定できます。
任意で指定します。

SetDllInfomation：**プラグイン情報の設定（必須）**

Delplotがプラグインを認識するために必要な情報（以下の 4 つの変数）を指定する必要があります。
必ず処理を指定する必要があります。

Style：プラグインの機能種別を指定します。（各種別についてはマニュアル参照）

DPPLUGIN_STYLE_IMPORT	入力用プラグイン
DPPLUGIN_STYLE_EXPORT	出力用プラグイン
DPPLUGIN_STYLE_SUPPLY	拡張ツール用プラグイン
DPPLUGIN_STYLE_ACTION	アクション用プラグイン

Guid：作成するプラグインの固有 ID を指定するため Windows GUID を設定します。設定方法は前述、各ツールのプラグイン作成方法を参照。

Name：プラグインをデータから呼び出す際に使用する名称を指定します。データファイル（SYSTEM_CONTROL 命令など）もしくはユーザー定義のメニュー・ボタンにこのプラグインを登録する時にはこの名前が使用されます。

Caption：プラグイン一覧などに表示する簡易名称を指定します。名称が長いと表示しきれないなどは起こりますが、上記 Name と同一でも問題ありません。

Delplot での各プラグイン認識は Guid（GUID）で識別され、プラグイン読込み時に**前優先**で排他処理を行い読込まれません。Name は重複が可能です。重複した名前をデータなどで指定した場合、プラグインの選択は**後登録**優先で処理されます。

PluginOption：**プラグイン・オプション項目の設定（任意）** Delplotでプラグインのオプションを設定する時の表示方法、フックしたいイベントの登録などを設定します。設定方法の詳細は後述、プラグイン提供構造のデータ接続構造で行われています。 任意で指定します。
PluginInitialize：**プラグインの開始処理（任意）** Delplotからプラグインによるサービス処理を起動する直前に行う処理を指定できます。 任意で指定します。
PluginTerminate：**プラグインの終了処理（任意）** Delplotがプラグインによるサービス処理を完了した直後に行う処理を指定できます。 任意で指定します。
PluginExecute：**プラグインの実行（必須）** プラグインによるサービス処理本体です。 必ず処理を指定する必要があります。 **引数**：EventNo（Long**値**）の値の詳細は後述、プラグイン提供構造のデータ接続構造でイベント登録に行う数値と同じ。 提供されるDelplotからの情報は後述、提供ライブラリを参照。

- 入力用プラグイン、出力用プラグインを作成する場合、PluginOption においてオプション設定する時に１番最初の項目は拡張子指定に予約されている。(オプションの指定方法は後述：プラグイン提供構造で説明) この項目は Comnd に DPPLUGIN_MENU_TEXTBOX を指定、Value に該当の拡張子を指定する。
  Value を空欄にするとファイルを開く、保存の描くダイアログに追加しない指定となる。

拡張子指定方法：１項目を“（ダブルクオート）で囲み、**項目説明 | *.xxx(拡張子)** 書式で指定する。２つ以上設定する場合は CSV 形式（カンマで区切り）で記述する。

（指定例）**C++**

```
Value = "¥"DelPlot CSV File(*.csv)|*.csv¥"";
```

Delphi

```
Value := '"DelPlot Data File(*.plt)|*.plt","DelPlot CSV File(*.csv)|*.csv"';
```

- プラグインの動作には**メニューコマンドモード**と**データアイテムモード**の２種類が存在する。

メニューコマンドモード	**Delplot のメニュー、ツールバーボタンに登録したプラグインを起動した時。**
データアイテムモード	**データ中に「拡張ツール」のアイテム（描画項目）として登録したプラグインを起動した時。**

各モードの動作：

	イベント利用	データ追加
メニューコマンドモード	○	通常描画命令でアイテム追加
データアイテムモード	×	全て描画のみ

イベントの説明は後述「プラグイン提供構造」を参照

データモードでデータ追加が不可なのは再描画実行の度にデータ項目数が増え続ける事が理由。また、現在表示中のページ以外への干渉はどちらのモードでも不可。

- 使用する文字はユニコード（UniCode）を使用する。例外的に入力用プラグインで使用する dpPlugin_ImportParameter（**C++**） _DelPlotImportBase（**Delphi**）の ReadBuffer は Ascii コードを使用する。
  （Shift-JIS などの文字コードによるデータファイル情報を無変換で直接送信するため）

- デフォルト（起動時）動作を指定する場合、Delplot の存在フォルダ内にある Define サブフォルダに設定内容を記述した plt ファイルを入れる事、Delplot のオプション画面にて起動時読込ファイルの指定を行う事で利用可能となる。

- プラグインをインストールにあたって Delplot では以下のファイルを定義する。必須、任意の別はあるが、すべてそろう事が望ましい。

<table>
<tr><th>拡張子</th><th>名称</th><th></th><th>解　説</th></tr>
<tr><td>dll</td><td>プラグイン本体</td><td>必須</td><td>Delplot 用プラグイン本体です。
Delplot は Win32 アンマネージメント DLL に対応しています。</td></tr>
<tr><td>jpg</td><td>アイコン</td><td>推奨</td><td>Delplot で拡張ツール選択ダイアログのアイコン表示やユーザー定義ボタン登録時にアイコンとして使用されます。32×32 のサイズを使用します。</td></tr>
<tr><td>key</td><td>検索用キーワード</td><td>必須</td><td>Delplot で拡張ツール選択ダイアログにて利用します
<table>
<tr><td>MajorKay</td><td>種別選択タブのジャンル
<table>
<tr><th>指定文字</th><th>分類</th></tr>
<tr><td>editor</td><td>データ編集</td></tr>
<tr><td>analysis</td><td>分析･表示</td></tr>
<tr><td>communication</td><td>コミュニケーション</td></tr>
<tr><td>import</td><td>入力プラグイン用（予約）</td></tr>
<tr><td>export</td><td>入力プラグイン用（予約）</td></tr>
<tr><td>action</td><td>アクションプラグイン用（予約）</td></tr>
</table></td></tr>
<tr><td>MinorKey</td><td>キーワード検索に入力された内容と照合するキーワード。
CSV 形式で複数指定が可能</td></tr>
</table>
ダイアログの検索機能で追加されたキーワードを保存す</td></tr>
</table>

			るため、ファイル属性を読み取り専用にしないでください。
txt	利用コメント	**推奨**	プラグインの内容紹介・利用方法を記述します。Delplot 内の各プラグイン選択機能にて表示されます。
pdf	利用ヘルプ	**推奨**	プラグインの詳細説明を記述します。ファイルが存在すれば Delplot 内で**プラグイン設定画面**、**拡張ツールパラメータダイアログ**のヘルプボタンクリック時に表示されます。
url	DownLoad URL	**推奨**	プラグインの入手先を記述します。 Delplot のデータファイル中にプラグインの利用が記述されていても利用環境では未インストールである場合などに拡張ツール収集ダイアログなどにて表示します。
plt	ひな形 （基本利用方法）	**任意**	実際のプラグインの使用例サンプルです。 Delplot のエディタモードではデータファイルにプラグイン（拡張ツール）を登録する時、このファイルがあればこのひな形を利用する事が出来ます。 初めてプラグインを利用してもらう時などの手引きや設定データを追加する作業の軽減に役立ちます。

※各プラグインはその利用方法（上記：プラグインの機能種別）によって Delplot の存在フォルダ内にある Plugin サブフォルダの下にあるそれぞれのフォルダに入れること。

用語説明

本ガイド内で解説に使用されている Delplot 特有の用語を説明します。

コマンド
Delplot 描画命令。定義と描画の機能があり各命令は整数値によるコマンド No とパラメータで構成される。

ペン
描画に使用するライン情報。線種、線幅、色がある。文字色の指定にも利用される。Delplot では情報の保存領域としてペン・ストッカー数分確保されている。

アイテム
コマンドにより Delplot データ構造に登録された１描画データ単位。ものによってはパラメータ情報だけのものからラベル情報、パス・パラメータ情報、アイテム固有の塗りつぶし情報などを持つものもある。

レイヤ
アイテムをまとめるグループ単位。それ自体が階層構造を持つ。オフセット、拡大率などの補助情報を持ち１レイヤ単位ごとの拡大縮小操作が可能。

ラベル
アイテムに添付する文字列情報。この文字列を利用してアイテムの呼び出しを行うなどの利便性に使用される。添付するアイテムによりレイヤ構造などのアイテムグループひとかたまりを呼び出す事も可能。ラベル No を使用して検索する事が出来る。

パス
PLOT_PATH、PLOT_POLYGON、PLOT_POLYLINE、PLOT_BEZIER、PLOT_FREESURFACE 描画命令で登録される座標点配列情報。Delplot の内部では上記の描画命令はパスという同じデータとして処理される。座標点ごとにパラメータという文字列を持つ事ができ、これに CSV 形式、Key=Value 形式などにより複数のデータ値を持たせるなど自由度を持つ。

メッシュ
PLOT_MESH 描画命令で登録される 3D 図形情報。上記パスと同じく座標点ごとにパラメータ文字列を保有可能。

アクション
Delplot 本体（エディタモード）でツールパレットに表示されている１つ１つの編集機能（図形、文字など）の事。これをユーザーカスタマイズ可能な様にプラグインに提供されている。

拡張ツール
Delplot のデータ描画時、ユーザー定義ボタンなどに設定する各種処理機能の事。既定の描画から、データ値の操作など幅広い機能を Delplot に追加可能にしている。

コマンドおよび設定値の詳細については描画命令ガイドを参照

サンプル一覧

Delplot プラグインの作成サンプルとして収録されているソースコードです。区分が Supply のものは初期ツールとして本体とともに配布されている物のソースコードです。

	フォルダ	適　用	区分	種別
1	act_visio	吹出しシェープ入力	Action	Delphi
2	exp_bmp	ビットマップ　出力	Export	C++
3	exp_csv	CSV 形式テキスト出力	Export	C++
4	imp_bmp	ビットマップ入力	Import	C++
5	imp_csv	CSV 形式テキスト入力	Import	C++
6	sup_axis	グラフ軸描画	Supply	C++
7	sup_contour	等高線(コンター)描画	Supply	C++
8	sup_contourPre	データ補完処理(コンター前処理)	Supply	C++
9	sup_dialog	ダイアログ表示	Supply	C++
10	sup_distribution	分布図描画	Supply	C++
11	sup_line_graph	線グラフ描画	Supply	C++
12	sup_point picker	画面上 座標点 取得	Supply	Delphi
13	sup_mesh extention	メッシュ編集機能 拡張	Supply	Delphi
14	sup_path editor	パスデータ編集	Supply	Delphi

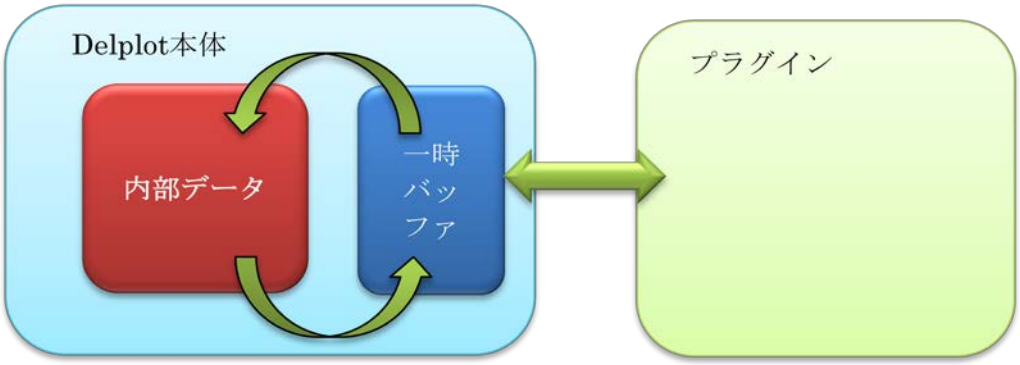
Import	入力用プラグイン
Export	出力用プラグイン
Supply	拡張ツール用プラグイン
Action	アクション用プラグイン

プラグイン提供構造

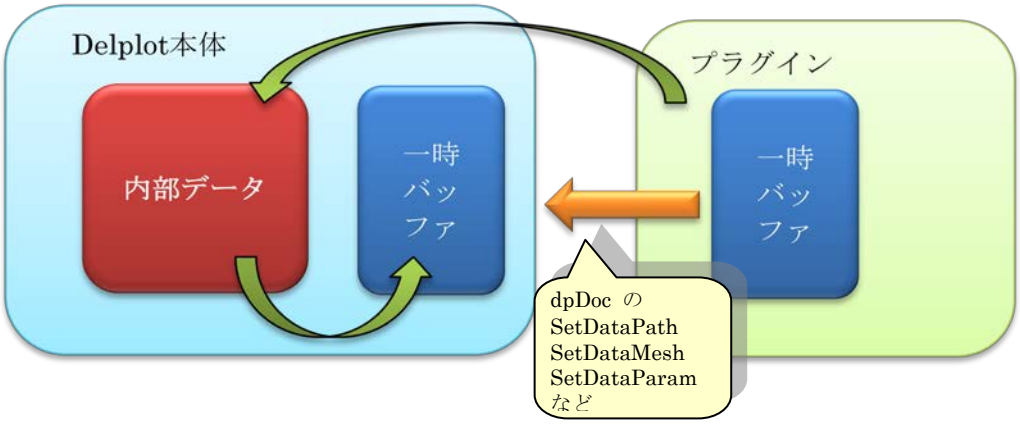
プラグイン インターフェース説明

Delplot では本体が起動時にプラグイン呼出しと接続が行われ、初期化が行われます。
この際、プラグインの識別情報（識別値、GUID、名称、各種設定）の受け渡しとともに、dpDocument クラスの変数 dpDoc（C++、Delphi ともに共通）がグローバルに作成され、その内部変数 Tbl とその他の参照変数の設定が行われます。プラグイン作成者は dpDoc の諸機能を利用する事で Delplot 本体との情報交換を容易に行えるよう便宜が図られています。

また、Delplot 本体とプラグイン間のデータの受け渡しには一時バッファが設けられ、このバッファを介して受け渡しが行われるため、直接本体データ構造にアクセスする事はできません。



文字列、パス、メッシュデータなど可変領域を持つデータを変更する場合、言い換えれば、一時バッファのサイズを変更する必要が出た、もしくは新たな描画項目（アイテム）を登録したい場合には自領域内に一時バッファを設け、Delplot 本体へ情報の更新要求を行う事でデータ更新を行う事が出来ます。



プラグインの解放時期について、Delplot では本体の終了時にプラグインの解放処理を行います。このため Delplot 起動中はプラグインの更新作業を行う事はできません。

データ接続構造

Delplot 本体とプラグイン間を接続するデータ構造について説明します。

● オプション・パラメータ・イベント情報

SetDllInfomation において設定を行い、プラグインの初期化時に参照されます。

オプションは Delplot 本体で**プラグイン設定画面**に表示される設定項目で同じデータファイル内で２つ同じプラグイン命令をデータ登録されていた場合、共通して使用されます。

項目	データ型	適　用
Comnd	long	入力項目の種別 DPPLUGIN_MENU_COLOR_SELECT　カラー設定 DPPLUGIN_MENU_TRACKBAR　トラックバー DPPLUGIN_MENU_CHECKBOX　チェックボックス DPPLUGIN_MENU_RADIOBUTTON　ラジオボタン DPPLUGIN_MENU_TEXTBOX　テキストボックス DPPLUGIN_MENU_FUNC_BUTTON　ボタン
Value	wchar_t*	データ初期値
Title	wchar_t*	項目の説明表示
Enable	long	DPPLUGIN_MENU_TRUE　有効 DPPLUGIN_MENU_FALSE　無効
Visible	long	DPPLUGIN_MENU_TRUE　表示 DPPLUGIN_MENU_FALSE　非表示

C++は std::vector **コンテナ**として実装され、Delphi では可変長レコード array として実装されています。必要な数を設定します。具体的な追加方法はサンプルを参考にして下さい。

特殊な用法：DPPLUGIN_MENU_CHECKBOX：チェックボックスでは Title の最後に“＾”の文字を指定すると改行されません。直後の入力項目の Title の文字にチェックボックスを付けたい場合に利用してください。

パラメータはプラグイン命令ごとに持つオプション文字列です。この設定はプラグインの種別が拡張ツールの場合、**拡張ツール パラメータダイアログ**を表示した時の入力方法に便宜を図るためのものです。

元来、プラグイン命令ではオプション文字のフォーマットは決まっておりません。その書式は各プラグインに依存としています。(各プラグイン作成者の責任においてパラメータ文字列のフォーマットを明記するようにしてください)

しかし、種別が拡張ツールの場合 Delplot の**拡張ツールパラメータダイアログ**に表示される都合上、以下の書式で扱います。

“Key1=Value1, Key2=Value2, Key3=Value3, …”

項目	データ型	適　用
Name	wchar_t*	項目の Key となる文字列
Memo	wchar_t*	項目の説明表示
Kind	Long	入力方法 DPPLUGIN_PARAM_INTEGER　整数値 DPPLUGIN_PARAM_FLOAT　実数値

		DPPLUGIN_PARAM_STRING 文字列 DPPLUGIN_PARAM_DATALABEL 項目ラベル
Need	Long	Kind が項目ラベルの場合、指定可能なコマンド No それ以外は-1
Optional	Long	DPPLUGIN_MENU_TRUE 省略可 DPPLUGIN_MENU_FALSE 省略不可

C++は std::vector コンテナとして実装され、Delphi では可変長レコード array として実装されています。必要な数を設定します。具体的な追加方法はサンプルを参考にして下さい。

イベントはそのプラグインで使用するイベント処理の種別を登録します。拡張ツール用、アクション用プラグインではイベントとして Delplot 本体の処理中にプラグインの処理を追加する事が出来ます。EventNo を登録されたイベントが Delplot 本体でフックされプラグイン呼出しを行わせる事ができます。

項目	データ型	適　用
EventNo	long	フックするイベント No DPPLUGIN_EVENT_INITIAL 初期化 DPPLUGIN_EVENT_CHANGE データ変更 DPPLUGIN_EVENT_CLICK クリック DPPLUGIN_EVENT_DBLCLICK ダブルクリック DPPLUGIN_EVENT_ITEMCLICK 項目選択 DPPLUGIN_EVENT_POINTCLICK 座標点選択 DPPLUGIN_EVENT_FILE_IMPORT ファイル読込完了 DPPLUGIN_EVENT_FILE_EXPORT ファイル書込完了 DPPLUGIN_EVENT_PAGE_IN ページ表示前 DPPLUGIN_EVENT_PAGE_OUT ページ移動前 DPPLUGIN_EVENT_DLAGDROP ドラッグドロップ DPPLUGIN_EVENT_DLAGOVER ドラッグオーバー DPPLUGIN_EVENT_KEYDOWN キー押下 DPPLUGIN_EVENT_KEYUP キー放す DPPLUGIN_EVENT_MOUSEDOWN マウスボタン押下 DPPLUGIN_EVENT_MOUSEMOVE マウス移動 DPPLUGIN_EVENT_MOUSEUP マウスボタン放す DPPLUGIN_EVENT_MOUSEWHEEL マウスホイール DPPLUGIN_EVENT_HBARCHANGE 水平スクロールバー DPPLUGIN_EVENT_VBARCHANGE 垂直スクロールバー DPPLUGIN_EVENT_TIMEUP インターバルタイマ
Value	long	イベント設定用パラメータ インターバルタイマ(DPPLUGIN_EVENT_TIMEUP)：割り込み発生時間（ミリ秒）

C++は std::vector コンテナとして実装され、Delphi では可変長レコード array として実装されています。必要な数を設定します。具体的な追加方法はサンプルを参考にして下さい。

- **アイテム情報**

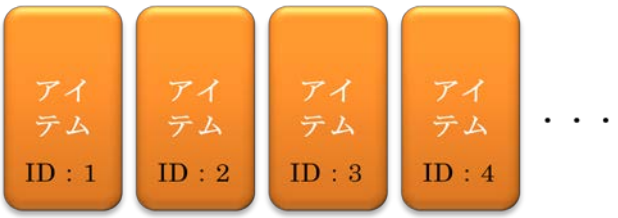
Delplot のアイテム情報は GetDataItem にて取得する事が出来ます。取得された情報は dpDoc. Item から参照できます。取得される内容について以下の通り

項目	適　用
SerialID	アイテム No
GroupID	（予約）
Value	アイテムのパラメータ情報構造体
LblID	アイテムの持つラベル No → GetDataLabel
TextID	アイテムが持つ文字列 No →GetDataText
ItemArea	アイテムの描画範囲
Path	パス情報（項目による）
Mesh	メッシュ情報（項目による）
Param	パラメータ情報（項目による）
Trans	アイテム固有のトランス（変形）情報（DEFINE_TRANS）
Shade	アイテム固有の影表示情報（EFFECT_SHADE）
Paint	アイテム固有の塗りつぶし情報（EFFECT_FILL）
Image	アイテム固有のイメージ（画像変形）情報（EFFECT_IMAGE）
Face	アイテム固有の質感情報（持つ場合）（予約）
LayerID	アイテムの所属するレイヤ No →GetDataLayer
Visible	アイテムの表示属性
Lock	アイテムのロック状態

GetDataItem を行った場合、GetDataLabel 、GetDataLayer、GetDataPath、GetData-Mesh を行わなくても自動的に情報を取得します。

Delplot アイテム構造

Delplot のアイテム構造は ID 番号により管理され、取得も ID 番号で行います。



アイテムに指定したパラメータは Value 内に保管されていますが、アイテムによって特殊な項目（DEFINE_TRANS、EFFECT_SHADE、EFFECT_FILL、EFFECT_IMAGE）は別の構造体で情報を提供しています。また、パス・メッシュなどの多数の座標値を持つデータは専用のデータ構造（Path、Mesh）にデータが提供されています。

以下、Value 内部 共用体 利用状況

名　称	使用状況
C++ dpPlugin_Data_PenColor Cl; **Delphi** CL : TDelPlotPlugin_Data_PenColor	DEFINE_PEN_COLOR

C++ dpPlugin_Data_Shape Ar; **Delphi** AR : TDelPlotPlugin_Data_Arc	PLOT_ELLIPSE、PLOT_CHORD、PLOT_ARC 、PLOT_PIE 、PLOT_RECTANGLE、PLOT_ROUNDRECT、PLOT_SHAPE
C++ dpPlugin_Data_TextInfo Sy; **Delphi** SY : TDelPlotPlugin_Data_TextInfo	DEFINE_SCALE、PLOT_SYMBOL、PLOT_GRAPH_SYMBOL、DEFINE_ROTATE、PLOT_IMAGE、PLOT_LABEL、DEFINE_CAMERA
C++ dpPlugin_Data_Define Ta; **Delphi** TA : TDelPlotPlugin_Data_Define	DEFINE_TEXT_ALIGN
C++ dpPlugin_Data_Pointer Ma; **Delphi** MA : TDelPlotPlugin_Data_Pointer	PLOT_MASK

※pen,fwd,bak,typ,x,y,z の各項目はすべてのアイテムで利用。

アイテム構造は１ページ内で完結され、ページ間にまたがりません。
ページ内で表示させたくないアイテム・レイヤ構造は非表示（Visible を使用）にしてページ内に置きます。

● 拡張ツール用イベントで使用可能なイベント

拡張ツール用プラグインでイベントを設定可能なのはエディタモードとビュアーモードの２つです。それぞれのモードにて指定できるイベントには若干違いがあります。

		エディタ	ビュアー
DPPLUGIN_EVENT_CHANGE	データ変更	○	×
DPPLUGIN_EVENT_CLICK	クリック	○	○
DPPLUGIN_EVENT_DBLCLICK	ダブルクリック	○	○
DPPLUGIN_EVENT_ITEMCLICK	項目選択	○	×
DPPLUGIN_EVENT_POINTCLICK	座標点選択	○	×
DPPLUGIN_EVENT_FILE_IMPORT	ファイル読込完了	○	○
DPPLUGIN_EVENT_FILE_EXPORT	ファイル書込完了	○	○
DPPLUGIN_EVENT_PAGE_IN	ページ表示前	○	○
DPPLUGIN_EVENT_PAGE_OUT	ページ移動前	○	○
DPPLUGIN_EVENT_DLAGDROP	ドラッグドロップ	○	○
DPPLUGIN_EVENT_DLAGOVER	ドラッグオーバー	○	○
DPPLUGIN_EVENT_KEYDOWN	キー押下	○	○
DPPLUGIN_EVENT_KEYUP	キー放す	○	○
DPPLUGIN_EVENT_MOUSEDOWN	マウスボタン押下	○	×
DPPLUGIN_EVENT_MOUSEMOVE	マウス移動	○	×
DPPLUGIN_EVENT_MOUSEUP	マウスボタン放す	○	×
DPPLUGIN_EVENT_MOUSEWHEEL	マウスホイール	○	○
DPPLUGIN_EVENT_HBARCHANGE	水平スクロールバー	○	○
DPPLUGIN_EVENT_VBARCHANGE	垂直スクロールバー	○	○
DPPLUGIN_EVENT_TIMEUP	インターバルタイマ	○	○

- **アクション用イベントで使用可能なイベント**

（アクション用プラグインではイベントの設定を行う必要はありません）

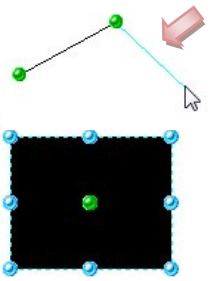
DPPLUGIN_EVENT_INITIAL	初期化
DPPLUGIN_EVENT_CHANGE	データ変更
DPPLUGIN_EVENT_CLICK	クリック
DPPLUGIN_EVENT_DBLCLICK	ダブルクリック
DPPLUGIN_EVENT_ITEMCLICK	項目選択
DPPLUGIN_EVENT_DLAGDROP	ドラッグドロップ
DPPLUGIN_EVENT_DLAGOVER	ドラッグオーバー
DPPLUGIN_EVENT_KEYDOWN	キー押下
DPPLUGIN_EVENT_KEYUP	キー放す
DPPLUGIN_EVENT_MOUSEDOWN	マウスボタン押下
DPPLUGIN_EVENT_MOUSEMOVE	マウス移動
DPPLUGIN_EVENT_MOUSEUP	マウスボタン放す
DPPLUGIN_EVENT_MOUSEWHEEL	マウスホイール
DPPLUGIN_EVENT_HBARCHANGE	水平スクロールバー
DPPLUGIN_EVENT_VBARCHANGE	垂直スクロールバー

また、アクション用プラグインでは Delplot 本体で使用されているガイドカーソルの利用が可能です。（DPPLUGIN_EVENT_INITIAL 初期化イベント内で PlotText へ下記、設定用文字列を指定）

(* 外部 Plugin で利用可能なカーソルの利用指定方法

GuidCursol: 指定２点間のガイドカーソル
0 : なし
1 : 矩形
2 : 直線

SelectBox : アイテム選択時 範囲指定 Box 表示
0 : なし
1 : あり



（記述例）

C++

```
CursolOption = L" GuidCursol=2; SelectBox=1" ;
dpDoc->Tbl->PlotText = CursolOption;
```

Delphi

```
CursolOption := 'GuidCursol=2; SelectBox=1';
dpDoc.Tbl.PlotText := PWideChar(CursolOption);
```

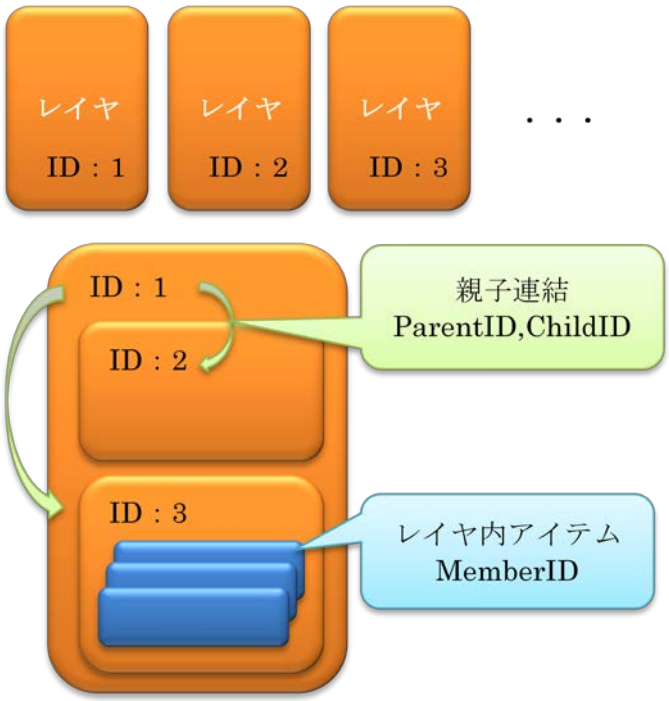
- **レイヤ情報**

Delplot では１ページ中のアイテムを一括してグループにまとめ、移動・拡大・回転などの操作や、レイヤグループを１つの描画項目としてマクロのように扱う事ができます。また、レイヤ構造は階層構造を作り何重にも入れ子状態にすることも可能です。

Delplot の持つレイヤ（データブロック・階層構造）の情報は GetDataLayer にて取得する事が出来ます。取得された情報は dpDoc. Layer から参照できます。

Delplot **レイヤ構造**

Delplot のレイヤ構造は ID 番号により管理・連結され、取得も ID 番号で行います。

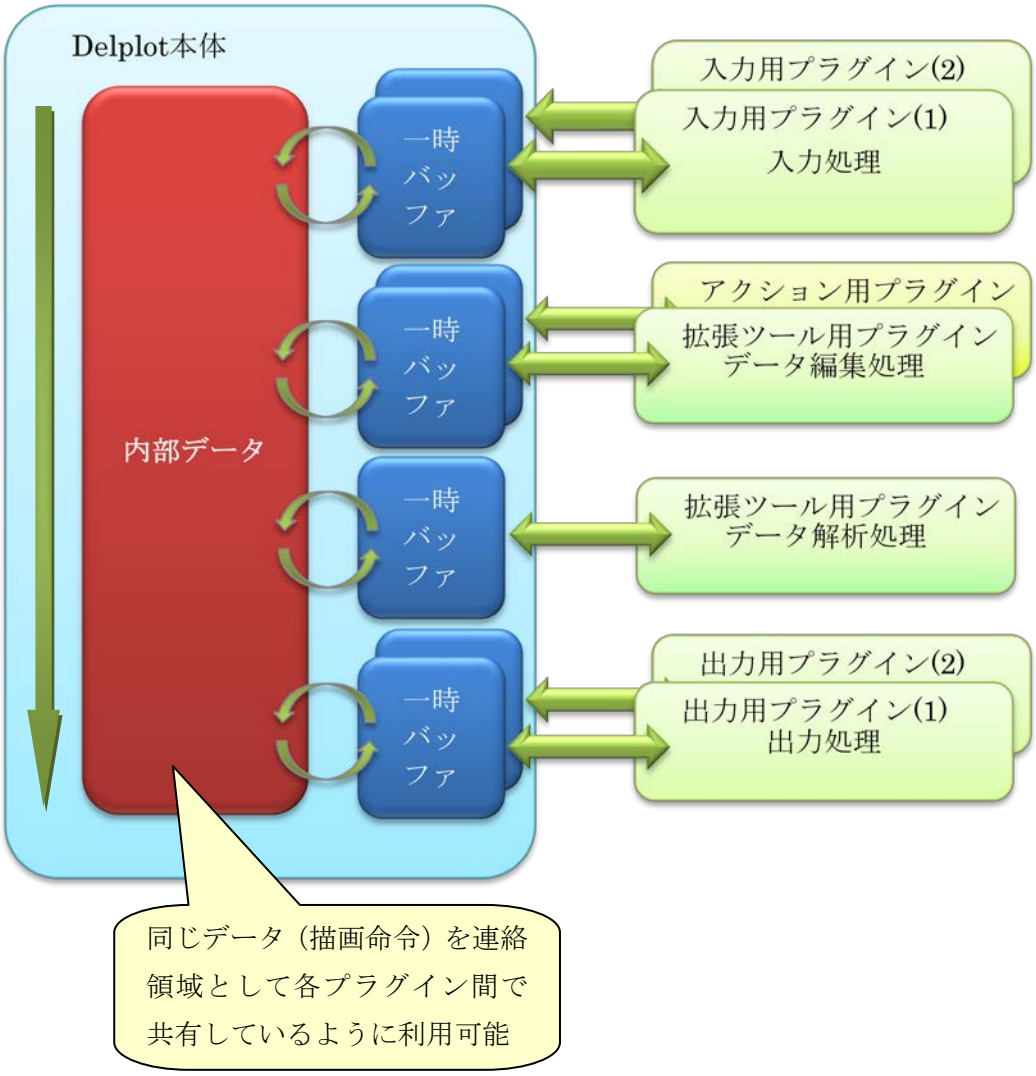


レイヤ構造は１ページ内で完結され、ページ間の接続はありません。

Plugin データ共通化のメリット

Delplot ではパス、メッシュ描画命令（PLOT_POLYGON、PLOT_POLYLINE、PLOT_BEZIER、PLOT_FREESURFACE、PLOT_PATH、PLOT_MESH 詳細は描画命令ガイド参照）が持つパラメータ情報などを本体⇔プラグインを繋ぐデータ構造として利用できます。
また、本体に登録したデータを共有領域としてプラグイン間で受け渡し別々のプラグインが一連の処理を行うようデータファイルを作成する事が出来ます。
この事でプラグインの機能を分割、それぞれを小さく作成する事で機能の共有化、プラグイン１つ当たりの作成作業量を軽減します。
また、既存の機能を利用する事は、作業量の軽減以外にその機能専用ゆえの安定性や対応条件の豊富さを利用可能な利便性もメリットとなります。

参考図



提供ライブラリー覧

DelPlot 本体とのインターフェースユーティリティとして提供されている dpDocument クラスについて解説します。

変数

	名称	適用
	Style	プラグイン種別値
	Guid	プラグイン識別 GUID
	Name	プラグイン識別名称
	Caption	プラグイン表示名称（略称）
	Option	プラグイン共通オプション情報 ※詳細はプラグイン規約参照
	Param	プラグイン個別パラメータ用情報 ※詳細はプラグイン規約参照
	Event	プラグインイベント処理情報 ※詳細はプラグイン規約参照
	FileName	本体(Delplot)からのパラメータ文字列 ファイル入出力の時はファイル名
	ExeOption	本体(Delplot)からのパラメータ文字列 汎用パラメータ

DelPlot 内部情報参照構造

	名称	適用
1	Tbl	Dll 間情報メイン構造への参照
2	Item	アイテム構造への参照
3	Layer	レイヤ構造への参照
4	Pen	ペン構造への参照
5	Form	フォーム構造への参照
6	Trans	トランス（変形）構造への参照
7	Paper	用紙（印刷情報）構造への参照
8	Paint	塗りつぶし情報 構造への参照
9	Shade	影表示情報 構造への参照
10	Image	イメージ（画像変形）構造への参照
11	Face	質感情報 構造への参照（予約）

12	Imp	インポート機能プラグイン用ブランチ
13	Exp	エクスポート機能プラグイン用ブランチ
14	Mouse	Delplot 上の現在マウス情報
15	FontName	使用フォント現在設定値

関数

（引数などの詳細は後述）

	名称	適用
Utility Function		
1	isDrawComnd	指定コマンドの種別判定（描画、定義）
2	ImportPluginSetup	入力用プラグイン構造体初期化
3	ExportPluginSetup	出力用プラグイン構造体初期化
4	BackupPenTable	ペン情報を退避領域に保存
5	ResumePenTable	ペン情報を保存時状態に戻す
DataControl Function		
1	GetDataItem	指定アイテムの情報を取得する
2	GetDataPath	指定アイテムのパス情報を取得する
3	GetDataMesh	指定アイテムのメッシュ情報を取得する
4	SetDataPath	指定アイテムのパス情報を変更する
5	SetDataMesh	指定アイテムのメッシュ情報を変更する
6	SetDataParam	指定アイテムのパラメータ情報を変更する
7	GetDataText	指定アイテムの保有文字列を取得する
8	GetDataLabel	指定アイテムのラベル情報を取得する
9	GetDataLayer	指定アイテムのレイヤ情報を取得する
10	ChangePathSize	指定アイテムのパス領域を変更する
11	ChangeMeshSize	指定アイテムのメッシュ領域を変更する
12	AddDataItem	現在ページにアイテムを追加する
13	DelDataItem	現在ページからアイテムを削除する
14	MovDataItem	指定アイテムの描画順を移動する
15	EdtDataItem	指定アイテムの情報を編集する
SettingCheck Function		
1	CheckCurrentPen	現在のペン番号を取得する
2	CheckPenColor	現在のペンの色を取得する
3	CheckPenStyle	現在のペンの線種を取得する
4	CheckPenWidth	現在のペンの線幅を取得する
5	CheckHatchNo	現在のペンのハッチ No を取得する

6	CheckTextArign	現在のテキスト配置情報を取得する
7	CheckLineType	現在のドット、ダッシュ幅を取得する
8	SystemControl	Delplot のシステム設定を変更する
9	PluginControl	別プラグインの起動を行う
10	OptionReflesh	オプション情報の更新を通知する
11	PageReflesh	現在ページを再描画する
12	LayerReflesh	現在レイヤを再描画する
13	SetAutoDraw	自動再描画機能設定
14	IntervalTimer	インターバルタイマー設定
15	PathPointSelect	現在パスの選択点を変更する
16	MeshPointSelect	現在メッシュの選択点を変更する
17	EventEnable	イベントの制御を行う

Define Function

1	DefineNewPen	現在のペンを変更する
2	DefineSetPen	現在のアイテムにペン No で色設定する
3	DefinePenColor	現在のペンの色を変更する
4	DefinePenStyle	現在のペンの線種を変更する
5	DefinePenWidth	現在のペンの線幅を変更する
6	DefineHatchNo	現在のペンのハッチを変更する
7	DefineFontName	現在のフォントを変更する
8	DefineLinkScale	（予約）
9	DefineDrawScale	スケール設定を変更する
10	DefineDrawExtend	拡大率を変更する
11	DefineDrawOffset	オフセットを変更する
12	DefineDrawRotate	回転角を変更する
13	DefineTextArign	テキスト配置情報を変更する
14	DefineLineType	ドット、ダッシュ幅を変更する
15	DefineLabel	現在のアイテムにラベルを設定する
16	DefineLayer	レイヤ設定を行う
17	DefineCamera	カメラ設定を行う
18	EffectPaint	塗りつぶし設定を行う
19	EffectShade	影表示設定を行う
20	EffectImage	イメージ（画像変形）設定を行う
21	EffectFace	質感設定を行う（予約）

Draw Function

1	PlotMove	ペンを移動する

2	PlotLine	線を描く
3	PlotRect	四角形を描く
4	PlotRRect	角の丸い四角形を描く
5	PlotEllips	楕円形を描く
6	PlotArc	円弧を描く
7	PlotPie	扇形を描く
8	PlotChord	弓型を描く
9	PlotShape	シェープ（基本図形）を描く
10	PlotPolyline	ポリライン（多点線分）を描く
11	PlotPolygon	ポリゴン（多角形）を描く
12	PlotPath	パス（自由図形）を描く
13	PlotMesh	メッシュ（3D 図形）を描く
14	PlotImage	画像ファイルを描画する
15	PlotText	文字列を描く
16	PlotExText	グラフ文字列を描く
17	PlotLabel	ラベル指定アイテムを描く
18	PlotMaskOn	マスクを使用する
19	PlotMaskOff	マスクを終了する
20	PlotTransOn	トランス（変形）を使用する
21	PlotTransOff	トランス（変形）を終了する

その他 （dpDocument 以外）ユーティリティ クラス

	名称	適用
1	IntPoint DblPoint	2D 座標値クラス
2	IntRect DblRect	2D 矩形範囲値クラス（Delphi には IntRect はありません TRect クラスを使用してください）
3	WcsPoint	3D 座標値クラス
4	Delphi _DelPlotTokenInfo C++ TokenInfo, TokenInfoA	トークンカッター（CSV, Key 文字列分解）
5	_DelPlotImportBase	入力用プラグイン基本クラス
6	_DelPlotExportBase	出力用プラグイン基本クラス

isDrawComnd

指定コマンドの種別判定（描画、定義）
描画コマンドの時 True が返されます。

構文

C++

```
bool isDrawComnd(int Command);
```

Delphi

```
function isDrawComnd(Command: Integer): Boolean;
```

Command　判定させるコマンド No を指定します。

ImportPluginSetup

入力用プラグイン構造体初期化
入力用プラグイン基本クラスを使用する事で入力用プラグインを作成する場合、Dll 間情報メイン構造に行う設定を簡略化します。

構文

C++

```
void ImportPluginSetup(_DelPlotImportBase* IP);
```

Delphi

```
procedure ImportPluginSetup(IP : _DelPlotImportBase);
```

IP　入力用プラグイン基本クラス DelPlotImportBase を指定します。

ExportPluginSetup

出力用プラグイン構造体初期化
出力用プラグイン基本クラスを使用する事で出力用プラグインを作成する場合、Dll 間情報メイン構造に行う設定を簡略化します。

構文

C++

```
void ExportPluginSetup(_DelPlotExportBase* EP);
```

Delphi

```
procedure ExportPluginSetup(EP : _DelPlotExportBase);
```

EP　　出力用プラグイン基本クラス DelPlotImportBase を指定します。

解説

BackupPenTable

ペン情報を退避領域に保存
プラグイン動作終了後、プラグイン動作によってペン情報が変えたくない場合プラグイン内部でペン操作を行う前に設定し、ResumePenTable で情報を復帰させることで元に戻します。

構文

C++

```
long BackupPenTable();
```

Delphi

```
function　BackupPenTable(): Longint;
```

ResumePenTable

ペン情報を保存時状態に戻す
プラグイン動作終了後、プラグイン動作によってペン情報が変えたくない場合プラグイン内部でペン操作を行う前に BackupPenTable で設定し、情報を復帰させることで元に戻します。

構文

C++

```
long ResumePenTable();
```

Delphi

```
function ResumePenTable(): Longint;
```

GetDataItem

指定アイテムの情報を取得する

Delplot 内部情報参照 dpDoc.Item の各構造体に現在のアイテム情報を設定、利用可能な状態にする。アイテム情報の詳細はプラグイン提供構造の解説を参照

構文

C++

```
long GetDataItem(long Index);
long GetDataItem(const wchar_t* Label);
```

Delphi

```
function GetDataItem(Index: Integer): Longint; overload;
function GetDataItem(const Label: String): Longint; overload;
```

Index　　アイテム No を指定します。

Label　　アイテムのラベルを指定します。

i GetDataPath

指定アイテムのパス情報を取得する

Delplot 内部情報参照の dpDoc.Item.Path 構造体に現在のパス設定を反映させる

構文

C++

```
long GetDataPath(long Index);
long GetDataPath(const wchar_t* Label);
```

Delphi

```
function GetDataPath(Index: Integer): Longint; overload;
```

function GetDataPath(const Label: String): Longint; overload;

Index　　アイテム No を指定します。

Label　　アイテムのラベルを指定します。

GetDataMesh

伸指定アイテムのメッシュ情報を取得する
Delplot 内部情報参照の dpDoc.Item.Mesh 構造体に現在のメッシュ設定を反映させる

構文

C++

```
long GetDataMesh(long Index);
long GetDataMesh (const wchar_t* Label);
```

Delphi

```
function GetDataMesh(Index: Integer): Longint; overload;
function GetDataMesh(const Label: String): Longint; overload;
```

Index　　アイテム No を指定します。

Label　　アイテムのラベルを指定します。

SetDataPath

指定アイテムのパス情報を変更する
dpDoc.Item.Path 構造体に設定した内容を Delplot のデータ構造に反映させる

構文

C++

```
long SetDataPath(long Index);
long SetDataPath(const wchar_t* Label);
```

Delphi

```
function SetDataPath(Index: Integer): Longint; overload;
function SetDataPath(const Label: String): Longint; overload;
```

Index　　アイテム No を指定します。

Label　　アイテムのラベルを指定します。

 最終的な Delplot データベースへの変更内容の反映は EdtDataItem で行います。

SetDataMesh

指定アイテムのメッシュ情報を変更する
dpDoc.Item.Mesh 構造体に設定した内容を Delplot のデータ構造に反映させる

構文

C++

```
long SetDataMesh(long Index);
long SetDataMesh(const wchar_t* Label);
```

Delphi

```
function SetDataMesh(Index: Integer): Longint; overload;
function SetDataMesh(const Label: String): Longint; overload;
```

Index　　アイテム No を指定します。

Label　　アイテムのラベルを指定します。

 最終的な Delplot データベースへの変更内容の反映は EdtDataItem で行います。

SetDataParam

指定アイテムのパラメータ情報を変更する
PLOT_POLYGON、PLOT_POLYLINE、PLOT_BEZIER、PLOT_FREESURFACE、PLOT_PATH、PLOT_MESH の各命令が持つパラメータ文字列を dpDoc.Item.Param 構造体に設定した内容を Delplot のデータ構造に反映させる

構文

C++

```
long SetDataParam(long Index);
long SetDataParam(const wchar_t* Label);
```

Delphi

```
function SetDataParam(Index: Integer): Longint; overload;
function SetDataParam(const Label: String): Longint; overload;
```

Index　　アイテム No を指定します。

Label　　アイテムのラベルを指定します。


最終的な Delplot データベースへの変更内容の反映は EdtDataItem で行います。

GetDataText

指定アイテムの保有文字列を取得する
アイテムが持つ文字列情報を dpDoc.Tbl.PlotText へ設定、利用可能な状態にする

構文

C++

```
wchar_t* GetDataText(long Index);
wchar_t* GetDataText(const wchar_t* Label);
```

Delphi

```
function GetDataText(Index: Integer): String; overload;
function GetDataText(const Label: String): String; overload;
```

Index　　アイテム No を指定します。

Label　　アイテムのラベルを指定します。

GetDataLabel

指定アイテムのラベル情報を取得する
アイテムが持つラベルを dpDoc.Tbl.ItemLabel へ設定する

構文

C++

```
wchar_t* GetDataLabel(long Index);
```

Delphi

```
function GetDataLabel(Index: Integer): String;
```

Index　　アイテム No を指定します。

GetDataLayer

指定アイテムのレイヤ情報を取得する
アイテムが持つレイヤ情報を dpDoc.Layer へ設定する

構文

C++

```
long GetDataLayer(long Index);
long GetDataLayer(const wchar_t* Label);
```

Delphi

```
function GetDataLayer(Index: Integer): Longint; overload;
function GetDataLayer(const Label: String): Longint; overload;
```

Index　　アイテム No を指定します。

Label　　アイテムのラベルを指定します。

ChangePathSize

指定アイテムのパス領域を変更する

構文

C++

```
long ChangePathSize(long Index);
long ChangePathSize(const wchar_t* Label);
```

Delphi

```
function ChangePathSize(Index: Integer): Longint; overload;
function ChangePathSize(const Label: String): Longint; overload;
```

Index　　アイテム No を指定します。

Label　　アイテムのラベルを指定します。

ChangeMeshSize

指定アイテムのメッシュ領域を変更する

構文

C++

```
long ChangeMeshSize(long Index);
long ChangeMeshSize(const wchar_t* Label);
```

Delphi

```
function ChangeMeshSize(Index: Integer): Longint; overload;
function ChangeMeshSize(const Label: String): Longint; overload;
```

Index　　アイテム No を指定します。

Label　　アイテムのラベルを指定します。

AddDataItem

現在ページにアイテムを追加する
Item で直接指定した情報のアイテムを Delplot データ構造に追加する

構文

C++

```
long AddDataItem(dpPlugin_DataItem* Item);
```

Delphi

```
function AddDataItem(Item: PDelPlotPlugin_DataItem): Longint;
```

Item　　データ構造情報を指定します。
　　しかし、後述の各定義・描画命令を使っても追加を行う事はできる。

DelDataItem

現在ページからアイテムを削除する
指定したアイテム No を Delplot データ構造から削除する

構文

C++

```
long DelDataItem(long Index);
```

Delphi

```
function DelDataItem(Index: Integer): Longint;
```

Index　　アイテム No を指定します。

MovDataItem

指定アイテムの描画順を移動する
Delplot データ構造内のアイテムの並び位置を変更し、描画順を操作します。

構文

C++

```
Long MovDataItem(long FromIndex, long ToIndex);
```

Delphi

```
function MovDataItem(FromIndex, ToIndex: Integer): Longint;
```

FromIndex、ToIndex　　アイテム No を指定します。

EdtDataItem

指定アイテムの情報を編集する

構文

C++

```
long EdtDataItem(long Index, dpPlugin_DataItem* Item);
```

Delphi

```
function EdtDataItem(Index: Integer; Item: PDelPlotPlugin_DataItem): Longint;
```

Index　　アイテム No を指定します。

Item　　データ構造情報を指定します。

CheckCurrentPen

現在のペン番号を取得する

構文

C++

```
long CheckCurrentPen(void);
```

Delphi

```
function CheckCurrentPen(): Longint;
```

CheckPenColor

現在のペンの色を取得する

構文

C++

```
Long CheckPenColor(long Index);
```

Delphi

```
function CheckPenColor(Index: Integer): Longint;
```

Index　　ペン No を指定します。

CheckPenStyle

現在のペンの線種を取得する

構文

C++

```
Long CheckPenStyle(long Index);
```

Delphi

```
function CheckPenStyle(Index: Integer): Longint;
```

Index　　ペン No を指定します。

 コマンドおよび設定値の詳細については描画命令ガイドを参照

CheckPenWidth

現在のペンの線幅を取得する

構文

C++

```
Long CheckPenWidth(long Index);
```

Delphi

```
function CheckPenWidth(Index: Integer): Longint;
```

Index　　ペン No を指定します。

 コマンドおよび設定値の詳細については描画命令ガイドを参照

CheckHatchNo

現在のペンのハッチ No を取得する

構文

C++

```
long CheckHatchNo (long Index);
```

Delphi

```
function CheckHatchNo(Index: Integer): Longint;
```

Index　　ペン No を指定します。

 コマンドおよび設定値の詳細については描画命令ガイドを参照

CheckTextArign

現在のテキスト配置情報を取得する

構文

C++

```
IntPoint CheckTextArign(void);
```

Delphi

```
function CheckTextArign(): TIntPoint;
```

 コマンドおよび設定値の詳細については描画命令ガイドを参照

CheckLineType

現在のドット、ダッシュ幅を取得する

構文

C++

```
DblPoint CheckLineType (void);
```

Delphi

```
function CheckLineType(): TDblPoint;
```

 コマンドおよび設定値の詳細については描画命令ガイドを参照

SystemControl

Delplot のシステム設定を変更する

RequestNo, Param 設定方法詳細は描画命令ガイド SYSTEM_CONTROL の項を参照

構文

C++

```
long SystemControl(long RequestNo, wchar_t* Param);
```

Delphi

```
function SystemControl(RequestNo: Integer; Param: String): Longint;
```

RequestNo　操作する種別を指定します。

Param　　種別ごとに操作内容を指定します。

 コマンドおよび設定値の詳細については描画命令ガイドを参照

PluginControl

別プラグインの起動を行う

構文

C++

```
long PluginControl(wchar_t* Name, wchar_t* Param);
```

Delphi

function PluginControl(Name, Param: String): Longint;

Name　　プラグイン名称を指定します。

Param　　パラメータを指定します。

OptionReflesh

オプション情報の更新を通知する
プラグインがオプションとして取得するデータの同期通知を行います

構文

C++

```
long OptionReflesh();
```

Delphi

function OptionReflesh(): Longint;

PageReflesh

現在ページを再描画する

構文

C++

```
long PageReflesh();
```

Delphi

function PageReflesh(): Longint;

LayerReflesh

現在レイヤを再描画する

構文

C++

```
long LayerReflesh(long LayerID);
```

Delphi

```
function LayerReflesh(LayerID: Longint): Longint;
```

LayerID　　対象レイヤのレイヤ ID を指定します。

SetAutoDraw

自動再描画機能設定

構文

C++

```
long SetAutoDraw(bool On);
```

Delphi

```
function SetAutoDraw(OnF: Boolean): Longint;
```

OnF　　自動再描画の ON/OFF を True/False で指定します。

IntervalTimer

インターバルタイマー設定

構文

C++

```
long IntervalTimer(long Enable , long Interval);
```

Delphi

```
function IntervalTimer(Enable: Boolean; Interval: Longint): Longint;
```

Enable　　有効(True)/無効(False)を指定します。

Interval　　インターバルタイマーイベントの間隔として基本値を指定することができます。
デフォルト値は 1000（1 秒）です。Enable が False の場合は無視されます。

PathPointSelect

現在パスの選択点を変更する
Delplot 本体で現在の動作が、エディタモードの線分（パスライン）編集機能であり、その編集モードがポイント編集中の場合、本関数で選択中のポイントを変更します。

構文

C++

```
long PathPointSelect(long Index, long Shift);
```

Delphi

```
function PathPointSelect(Index, Shift: Longint): Longint;
```

Index　　パス内の座標点 Index を指定します。

Shift　　Shift キーによる複数指定を指定します。

MeshPointSelect

現在メッシュの選択点を変更する
Delplot 本体で現在の動作が、エディタモードの 3D（立体形状）編集機能であり、その編集モードが範囲選択以外の場合、本関数で選択中のポイントを変更します。

構文

C++

```
Long MeshPointSelect(long Index, long Shift);
```

Delphi

```
function MeshPointSelect(Index, Shift: Longint): Longint;
```

Index　　メッシュ内の座標点 Index を指定します。

Shift　　Shift キーによる複数指定を指定します。

EventEnable

イベントの制御を行う
Delplot のイベント処理は基本的に PluginExecute を開始してから終了する時点までが１つの区切りです。
プラグインをメニューやボタンに登録した時（メニューモード）でフォームを使用する時など、PluginExecute 終了後もフォームでイベントを監視したい場合などプラグインがこの制限を超えてイベント処理を実行する必要がある時に切り替えます。
メニューモード実行時でこのイベント制御を稼働 Enable＝True にした場合、DefineFunction、DrawFunction のライブラリを実行すると現在ページ内にデータが追加されていきます。

構文

C++

```
Long EventEnable(bool Value);
```

Delphi

```
function EvantEnable(Value: Boolean): Longint;
```

Value　　イベント処理の稼働・停止を指定します。

DefineNewPen

現在のペンを変更する

構文

C++

```
long DefineNewPen (long Index);
```

Delphi

```
function DefineNewPen(Index: Integer): Longint;
```

Index　　ペン No を指定します。

DefineSetPen

現在のアイテムにペン No で色設定する

構文

C++

```
long DefineSetPen (long pen, long fwd, long bak);
```

Delphi

```
function DefineSetPen(pen, fwd, bak: Integer): Longint;
```

pen	枠線色、ペン No を指定します。
fwd	塗りつぶし前景色、ペン No を指定します。
bak	塗りつぶし背景色、ペン No を指定します。

DefinePenColor

現在のペンの色を変更する

構文

C++

```
long DefinePenColor(long Index, long Color);
```

Delphi

```
function DefinePenColor(Index, Color: Integer): Longint;
```

Index	ペン No を指定します。
Color	色値(BGR)を指定します。

DefinePenStyle

現在のペンの線種を変更する

構文

C++

```
Long DefinePenStyle(long Index, long Style);
```

Delphi

```
function DefinePenStyle(Index, Style: Integer): Longint;
```

Index　　ペン No を指定します。

Style　　線種 No を指定します。

 コマンドおよび設定値の詳細については描画命令ガイドを参照

DefinePenWidth

現在のペンの線幅を変更する

構文

C++

```
Long DefinePenWidth(long Index, long Width);
```

Delphi

```
function DefinePenWidth(Index, Width: Integer): Longint;
```

Index　　ペン No を指定します。

Width　　線幅を指定します。

 コマンドおよび設定値の詳細については描画命令ガイドを参照

DefineHatchNo

現在のペンのハッチを変更する

構文

C++

```
long DefineHatchNo (long Index, long Hatch);
```

Delphi

```
function DefineHatchNo(Index, Hatch: Integer): Longint;
```

Index　　ペン No を指定します。

Hatch　　ハッチ No を指定します。

 コマンドおよび設定値の詳細については描画命令ガイドを参照

DefineFontName

現在のフォントを変更する

構文

C++

```
long DefineFontName(const wchar_t* Name);
```

Delphi

```
function DefineFontName(const Name: String): Longint;
```

Name　　フォント名を指定します。

DefineLinkScale

（予約）

構文

C++

```
long DefineLinkScale (long F);
```

Delphi

```
function DefineLinkScale(F: Integer): Longint;
```

DefineDrawScale

スケール設定を変更する

構文

C++

```
long DefineDrawScale (long F, double x, double y);
```

Delphi

```
function DefineDrawScale(F: Integer; x,y: Double): Longint;
```

F　　スケールの単位指定値を指定します。

x, y　　拡大率 X,Y を指定します。

 コマンドおよび設定値の詳細については描画命令ガイドを参照

DefineDrawExtend

拡大率を変更する
現在レイヤ情報の拡大率を変更します

構文

C++

```
Long DefineDrawExtend(double x, double y);
```

Delphi

```
function DefineDrawExtend(x, y: Double): Longint;
```

x, y　　拡大率 X,Y を指定します。

 コマンドおよび設定値の詳細については描画命令ガイドを参照

DefineDrawOffset

オフセットを変更する
現在レイヤ情報のオフセット（原点位置）を変更します

構文

C++

```
Long DefineDrawOffset(double x, double y);
```

Delphi

```
function DefineDrawOffset(x, y: Double): Longint;
```

x, y　　レイヤ原点位置 X,Y を指定します。

 コマンドおよび設定値の詳細については描画命令ガイドを参照

DefineDrawRotate

回転角を変更する
現在レイヤ情報の回転角を変更します

構文

C++

```
long DefineDrawRotate(double x, double y);
```

Delphi

```
function DefineDrawRotate(x, y: Double): Longint;
```

x, y　　レイヤ回転角 X,Y を指定します。(現在は X のみ利用)

 コマンドおよび設定値の詳細については描画命令ガイドを参照

DefineTextArign

テキスト配置情報を変更する

構文

C++

```
Long DefineTextArign (long   x, long   y);
```

Delphi

```
function DefineTextArign(x, y: Integer): Longint;
```

x, y　　X(横),Y(縦)方向の文字原点位置を指定します。

 コマンドおよび設定値の詳細については描画命令ガイドを参照

DefineLineType

ドット、ダッシュ幅を変更する

構文

C++

```
long DefineLineType (double x, double y);
```

Delphi

```
function DefineLineType(x, y: Double): Longint;
```

x, y　　線表示の X(ドット),Y(ダッシュ)の長さを指定します。

 コマンドおよび設定値の詳細については描画命令ガイドを参照

DefineLabel

現在のアイテムにラベルを設定する

構文

C++

```
Long DefineLabel (long Hid, wchar_t* Name);
```

Delphi

```
function DefineLabel(Hid: Integer; Name: String): Longint;
```

Hid　　隠す(1)、表示(0)を指定します。

Name　　ラベル名を指定します。

 コマンドおよび設定値の詳細については描画命令ガイドを参照

DefineLayer

レイヤ設定を行う
dpDoc.Layer に設定した内容を Delplot データ構造（現在レイヤ）に反映する

構文

C++

```
long DefineLayer (void);
```

Delphi

```
function DefineLayer(): Longint;
```

 コマンドおよび設定値の詳細については描画命令ガイドを参照

DefineCamera

カメラ設定を行う
現在のカメラ（3D 図形座標の視点）を変更する

構文

C++

```
Long DefineCamera(double zoom, double px, double py, double pz, double rx, double ry, double rz, double ox, double oy);
```

Delphi

function DefineCamera(zoom, px, py, pz, rx, ry, rz, ox, oy: Double): Longint;

px,py,pz　　3D 座標内の注視点座標を指定します。

rx,ry,rz　　3D 座標内の視点回転方向角を指定します。

ox,oy　　3D 座標を 2D 座標に投影する座標を指定します。

 コマンドおよび設定値の詳細については描画命令ガイドを参照

EffectPaint

塗りつぶし設定を行う

塗りつぶし情報 構造 dpDoc.Paint への設定を Delplot データ構造に反映する

構文

C++

```
long EffectPaint (void);
```

Delphi

function EffectPaint(): Longint;

 コマンドおよび設定値の詳細については描画命令ガイドを参照

EffectShade

影表示設定を行う

影表示情報 構造 dpDoc.Shade への設定を Delplot データ構造に反映する

構文

C++

```
long EffectShade (void);
```

Delphi

function EffectShade(): Longint;

 コマンドおよび設定値の詳細については描画命令ガイドを参照

EffectImage

イメージ（画像変形）設定を行う

イメージ（画像変形）構造 dpDoc.Image への設定を Delplot データ構造に反映する

構文

C++

```
long EffectImage (void);
```

Delphi

function EffectImage(): Longint;

 コマンドおよび設定値の詳細については描画命令ガイドを参照

EffectFace

質感設定を行う（予約）

質感情報構造 dpDoc.Face への設定を Delplot データ構造に反映する

構文

C++

```
long EffectFace (void);
```

Delphi

function EffectFace(): Longint;

 コマンドおよび設定値の詳細については描画命令ガイドを参照

PlotMove

ペンを移動する

構文

C++

```
Long PlotMove (double x1,double y1);
```

Delphi

```
function PlotMove(x1, y1: Double): Longint;
```

x1, y1　　移動後の座標を指定します。

 コマンドおよび設定値の詳細については描画命令ガイドを参照

PlotLine

線を描く

構文

C++

```
Long PlotLine (double x1,double y1);
long PlotLine (double x1,double y1,double x2,double y2);
long PlotLine (DblPoint p1,DblPoint p2);
```

Delphi

```
function PlotLine(x1, y1: Double): Longint; overload;
function PlotLine(x1, y1, x2, y2: Double): Longint; overload;
function PlotLine(p1, p2: TDblPoint): Longint; overload;
```

x1, y1,x2,y2, p1,p2　　線座標を指定します。

 コマンドおよび設定値の詳細については描画命令ガイドを参照

PlotRect

四角形を描く

構文

C++

```
long PlotRect (double L, double T,double R, double B, double Rot);
long PlotRect (DblRect Rect, double Rot);
```

Delphi

```
function PlotRect(L, T, R, B, Rot: Double): Longint; overload;
function PlotRect(Rect: TDblRect; Rot: Double): Longint; overload;
```

L, T, R, B,Rect　Left,Top,Right,Bottom 座標を指定します。

Rot　回転角を指定します。

 コマンドおよび設定値の詳細については描画命令ガイドを参照

PlotRRect

角の丸い四角形を描く

構文

C++

```
long PlotRRect (double L, double T,double R, double B, double Rot, double W, double H);
long PlotRRect (DblRect Rect, double Rot, double W, double H);
```

Delphi

```
function PlotRRect(L, T, R, B, Rot, W, H: Double): Longint; overload;
function PlotRRect(Rect: TDblRect; Rot, W, H: Double): Longint; overload;
```

L, T, R, B,Rect　Left,Top,Right,Bottom 座標を指定します。

Rot　回転角を指定します。

W,H　　角の半径 W(X 方向)、H(Y 方向)を指定します。

 コマンドおよび設定値の詳細については描画命令ガイドを参照

PlotEllips

楕円形を描く

構文

C++

```
Long PlotEllips (double L, double T,double R, double B, double Rot);
long PlotEllips (DblRect Rect, double Rot);
```

Delphi

```
function PlotEllips(L, T, R, B, Rot: Double): Longint; overload;
function PlotEllips(Rect: TDblRect; Rot: Double): Longint; overload;
```

L, T, R, B,Rect　L,T:中心座標、R,B XY 方向半径を指定します。

Rot　　回転角を指定します。

 コマンドおよび設定値の詳細については描画命令ガイドを参照

PlotArc

円弧を描く

構文

C++

```
Long PlotArc (double L, double T,double R, double B, double Rot, double StRad, double EdRad);
Long PlotArc (DblRect Rect, double Rot, double StRad, double EdRad);
```

Delphi

```
function PlotArc(L, T, R, B, Rot, StRad, EdRad: Double): Longint; overload;
```

function PlotArc(Rect: TDblRect; Rot, StRad, EdRad: Double): Longint; overload;

L, T, R, B,Rect　L,T:中心座標、R,B XY 方向半径を指定します。

Rot　回転角を指定します。

StRad,EdRad　描画開始角、描画終了角を指定します。

 コマンドおよび設定値の詳細については描画命令ガイドを参照

PlotPie

扇形を描く

構文

C++

```
long PlotPie (double L, double T, double R, double B, double Rot, double StRad, double EdRad);
long PlotPie (DblRect Rect, double Rot, double StRad, double EdRad);
```

Delphi

function PlotPie(L, T, R, B, Rot, StRad, EdRad: Double): Longint; overload;
function PlotPie(Rect: TDblRect; Rot, StRad, EdRad: Double): Longint; overload;

L, T, R, B,Rect　L,T:中心座標、R,B XY 方向半径を指定します。

Rot　回転角を指定します。

StRad,EdRad　描画開始角、描画終了角を指定します。

 コマンドおよび設定値の詳細については描画命令ガイドを参照

PlotChord

弓型を描く

構文

C++

```
long PlotChord (double L, double T,double R, double B, double Rot, double StRad, double EdRad);
long PlotChord (DblRect Rect, double Rot, double StRad, double EdRad);
```

Delphi

```
function PlotChord(L, T, R, B, Rot, StRad, EdRad: Double): Longint; overload;
function PlotChord(Rect: TDblRect; Rot, StRad, EdRad: Double): Longint; overload;
```

L, T, R, B,Rect　L,T:中心座標、R,B XY 方向半径を指定します。

Rot　回転角を指定します。

StRad,EdRad　描画開始角、描画終了角を指定します。

 コマンドおよび設定値の詳細については描画命令ガイドを参照

PlotShape

シェープ（基本図形）を描く

構文

C++

```
long PlotShape (double L, double T,double R, double B, double Rot, double W, double H, long Typ);
long PlotShape (DblRect Rect, double Rot, double W, double H, long Typ);
```

Delphi

```
function PlotShape(L, T, R, B, Rot, W, H: Double; Typ: Integer): Longint; overload;
function PlotShape(Rect: TDblRect; Rot, W, H: Double; Typ: Integer): Longint; overload;
```

L, T, R, B,Rect　L,T:中心座標、R,B XY 方向半径を指定します。

Rot　回転角を指定します。

W,H　　Type が角丸矩形の時、角の半径 W(X 方向)、H(Y 方向)を指定します。
円弧、扇形、弓型の時、描画開始角、描画終了角を指定します。

Typ　　描画する図形の種別を指定します。

 コマンドおよび設定値の詳細については描画命令ガイドを参照

PlotPolyline

ポリライン（多点線分）を描く

構文

C++

```
Long PlotPolyline(double* x, double* y, long num);
```

Delphi

```
function PlotPolyline(x, y: Array of Double): Longint;
```

x, y　　座標点の配列を指定します。

num　　配列数を指定します。

 コマンドおよび設定値の詳細については描画命令ガイドを参照

PlotPolygon

ポリゴン（多角形）を描く

構文

C++

```
long PlotPolygon (double* x, double* y, long num);
long PlotPolygon (DblPoint* data, long num);
```

Delphi

function PlotPolygon(x, y: Array of Double): Longint; overload;

function PlotPolygon(data: Array of TDblPoint): Longint; overload;

x, y, data　　座標点の配列を指定します。

num　　配列数を指定します。


コマンドおよび設定値の詳細については描画命令ガイドを参照

PlotPath

パス（自由図形）を描く

構文

C++

```
Long PlotPath (double* x, double* y, long* Node, long num);
```

Delphi

function PlotPath(x, y: Array of Double; Node: Array of Integer): Longint; overload;

x, y　　座標点の配列を指定します。

Node　　パスを構成するノード情報を指定します。

num　　配列数を指定します。


コマンドおよび設定値の詳細については描画命令ガイドを参照

PlotMesh

メッシュ（3D 図形）を描く

構文

C++

```
Long PlotMesh (double* x, double* y, double* z, long posNum);
```

```
long PlotMesh (double* x, double* y, double* z, long posNum, dpPlugin_DataPlane*
Plane, long plnNum);
```

Delphi

```
function PlotMesh(x, y, z: Array of Double): Longint; overload;
function PlotMesh(x, y, z: Array of Double; Plane : Array of TDelPlotPlugin_DataPlane):
Longint; overload;
```

x, y, z	座標点の配列を指定します。
posNum	座標配列数を指定します。
Plane	座標点で構成する面情報を指定します。
plnNum	面情報配列数を指定します。

 コマンドおよび設定値の詳細については描画命令ガイドを参照

PlotImage

画像ファイルを描画する

構文

C++

```
Long PlotImage (double x, double y, double w, double h, double Rot, double rx, double
ry, long typ, long lbl, wchar_t* Image);
```

Delphi

```
function PlotImage(x, y, w, h, Rot, rx, ry: Double; typ, lbl: Integer; Image: String): Longint;
```

x, y	描画左下座標を指定します。
w, h	描画画像のw(幅)、h(高さ)を指定します。
Rot	回転角を指定します。
rx, ry	回転中心座標を指定します。

typ　　画像の描画方法を指定します。

lbl　　透過色ペン番号を指定します。

Image　　画像ファイルパスを指定します。

 コマンドおよび設定値の詳細については描画命令ガイドを参照

PlotText

文字列を描く

構文

C++

```
Long PlotText (double x, double y, double Rot, double h, wchar_t* Text);
```

Delphi

```
function PlotText(x, y, Rot, h: Double; Text: String): Longint;
```

x, y　　文字列の描画座標を指定します。

Rot　　回転角を指定します。

h　　文字高さを指定します。

Text　　描画する文字列を指定します。

 コマンドおよび設定値の詳細については描画命令ガイドを参照

PlotExText

グラフ文字列を描く

構文

C++

```
Long PlotExText (double x, double y, double Rot, double w, double h, long eff, long lbl, long fcs, wchar_t* Text);
```

Delphi

```
function PlotExText(x, y, Rot, w, h: Double; eff, lbl, fcs: Integer; Text: String): Longint;
```

x, y	文字列の描画座標を指定します。
Rot	回転角を指定します。
w, h	描画する文字列のw(幅)、h(高さ)を指定します。
eff	変形方法を指定します。
lbl	配置指定を行う図形のラベル No を指定します。
fcs	キャラクタセットを指定します。
Text	描画する文字列を指定します。

 コマンドおよび設定値の詳細については描画命令ガイドを参照

PlotLabel

ラベル指定アイテムを描く
ラベルで指定された描画命令、レイヤブロックを座標、拡大率、回転角を指定して描く

構文

C++

```
long PlotLabel (long Index, double x, double y, double w, double h, double Rot);
```

Delphi

```
function PlotLabel(Index: Integer; x,y,w,h, Rot: Double): Longint;
```

x, y	描画座標を指定します。
w, h	w(幅)、h(高さ)方向の拡大率を指定します。
Rot	回転角を指定します。

 コマンドおよび設定値の詳細については描画命令ガイドを参照

PlotMaskOn

マスクを使用する

構文

C++

```
long PlotMaskOn();
```

Delphi

```
function PlotMaskOn(): Longint;
```

 コマンドおよび設定値の詳細については描画命令ガイドを参照

PlotMaskOff

マスクを終了する

構文

C++

```
long PlotMaskOff();
```

Delphi

```
function PlotMaskOff(): Longint;
```

 コマンドおよび設定値の詳細については描画命令ガイドを参照

PlotTransOn

トランス（変形）を使用する

構文

C++

```
long PlotTransOn(long n, long m, double* sx, double* sy, double* tx, double* ty);
```

Delphi

```
function PlotTransOn(n, m: Integer; sx, sy, tx, ty: Array of Double): Longint;
```

n, m	変形対象に設定する n(X 方向)、m(Y 方向)格子数を指定します。
sx, sy	変形前の図形にあてる格子座標点(%)の配列を指定します。
tx, ty	変形後の sx,sy に対応する図形位置を示す座標点(%)の配列を指定します。

 コマンドおよび設定値の詳細については描画命令ガイドを参照

PlotTransOff

トランス（変形）を終了する

構文

C++

```
longPlotTransOff();
```

Delphi

```
function PlotTransOff(): Longint;
```

 コマンドおよび設定値の詳細については描画命令ガイドを参照

DelPlot　ver.1.9.0 プラグイン SDK ガイド

2012/06/30.　　第 1 版発行

Copyright ©2003-2012 Table Soft.

発　行　　*テーブルソフト*

Email : tes@hello.jp
http : //www.hello.ne.jp/tes/

発行者　　平 田　治

ご注意

- 本製品の一部または全部を弊社の許可無く複写・複製することは禁じます。
- 本製品の内容・仕様は改正・改善のため予告なく変更することがあります。
- 本マニュアル記載のソフトウェア使用許諾書に基づいて個人で使用する場合を除いて、弊社の許諾なしにこのソフトウェアおよびマニュアルを使用することを固くお断りします。
- このソフトウェアおよびマニュアルを運用した結果の影響については責任を負いかねますので、ご了承ください。
- 本製品の内容につきましては万全を期しておりますが、万一ご不審な点やお気づきの点がございましたら、弊社までご連絡ください。