

CompeGPS Plug-IN specifications

COMPE GPS

Index

1	INTRODUCTION	3
1.1	HOW TO COMPILE A COMPEGPS PLUG-IN.....	3
2	CREATING THE VERSION AND INIT FUNCTIONS	6
2.1	INT COMPEPLUGIN_VERSION()	6
2.2	BOOL COMPEPLUGIN_INIT (INT Hwnd)	6
2.3	CONST CHAR* COMPEPLUGIN_VERSIONSTR().....	6
2.4	CONST CHAR* COMPEPLUGIN_GETTIPOARCHIVOS()	6
2.5	VOID COMPEPLUGIN_SETTEMPFOLDER (....).....	7
3	BITMAP FUNCTIONS	8
3.1	BOOL COMPEPLUGIN_ABRE_RASTER (.....);	8
3.2	CLASS TDATOSDIBPLUGIN.....	8
3.2.1	<i>AdvancedData</i>	9
3.3	BOOL COMPEPLUGIN_ABRE_ZONA (...)	9
3.4	BOOL COMPEPLUGIN_CIERRA_RASTER (...).....	10
4	MAP FUNCTIONS	11
5	BITMAP SAVING FUNCTIONS	12
5.1	BOOL COMPEPLUGIN_SAVE_RASTER_CREATE (.....);	12
5.2	TDATOSAVEDIBPLUGIN	12
5.3	BOOL COMPEPLUGIN_SAVE_RASTER_ADDLINES (.....);	13
5.4	BOOL COMPEPLUGIN_SAVE_RASTER_CLOSE (.....);	13
6	DEM FUNCTIONS	15
6.1	BOOL __EXPORT COMPEPLUGIN_LEE_DEM (.....);	15
6.2	CLASS TDEMPPLUGIN	15
6.3	EXAMPLE.....	16
7	VECTOR MAPS FUNCTIONS	18
7.1	BOOL COMPEPLUGIN_OPEN_VECTOR_MAP (...).....	18
7.2	CLASS TVECTORMAPPLUGIN	18
7.3	BOOL COMPEPLUGIN_READ_VECTOR_MAP3 (...).....	19
7.4	BOOL COMPEPLUGIN_CLOSE_VECTOR_MAP (...)	19
7.5	BOOL COMPEPLUGIN_ADD_LAYER (...).....	20
8	TRACK FUNCTIONS	22
9	WAYPOINTS FUNCTIONS	23

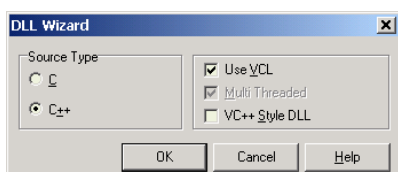
1 Introduction

1.1 How to compile a CompeGPS Plug-IN for PC

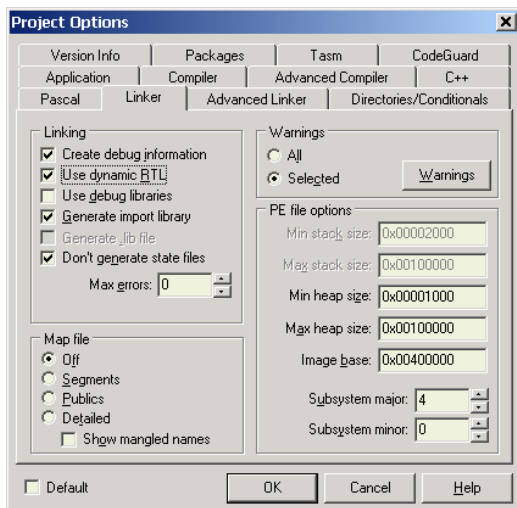
We recommend to make the Plug-ins using Borland C++ Builder version 5.0, 6.0 or 2006, or with Microsoft Visual C++.

To create this DLL using Borland C++ Version 5, follow this procedure:

1. Open Borland C++ Builder.
2. Select the 'File > New' command
3. Click on the 'DLL Wizard' window



4. Click on the menu option 'Project > Options', and **Deactivate** the 'Use dynamic RTL' option.

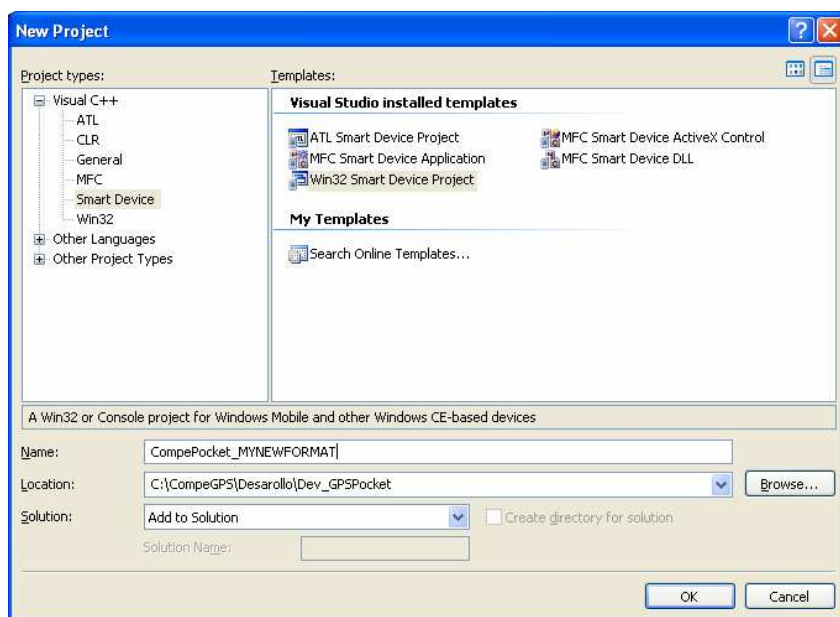


5. Finally, create a new C++ module, and write the DLL functions.

1.2 How to compile a CompeGPS Plug-IN for PocketPC

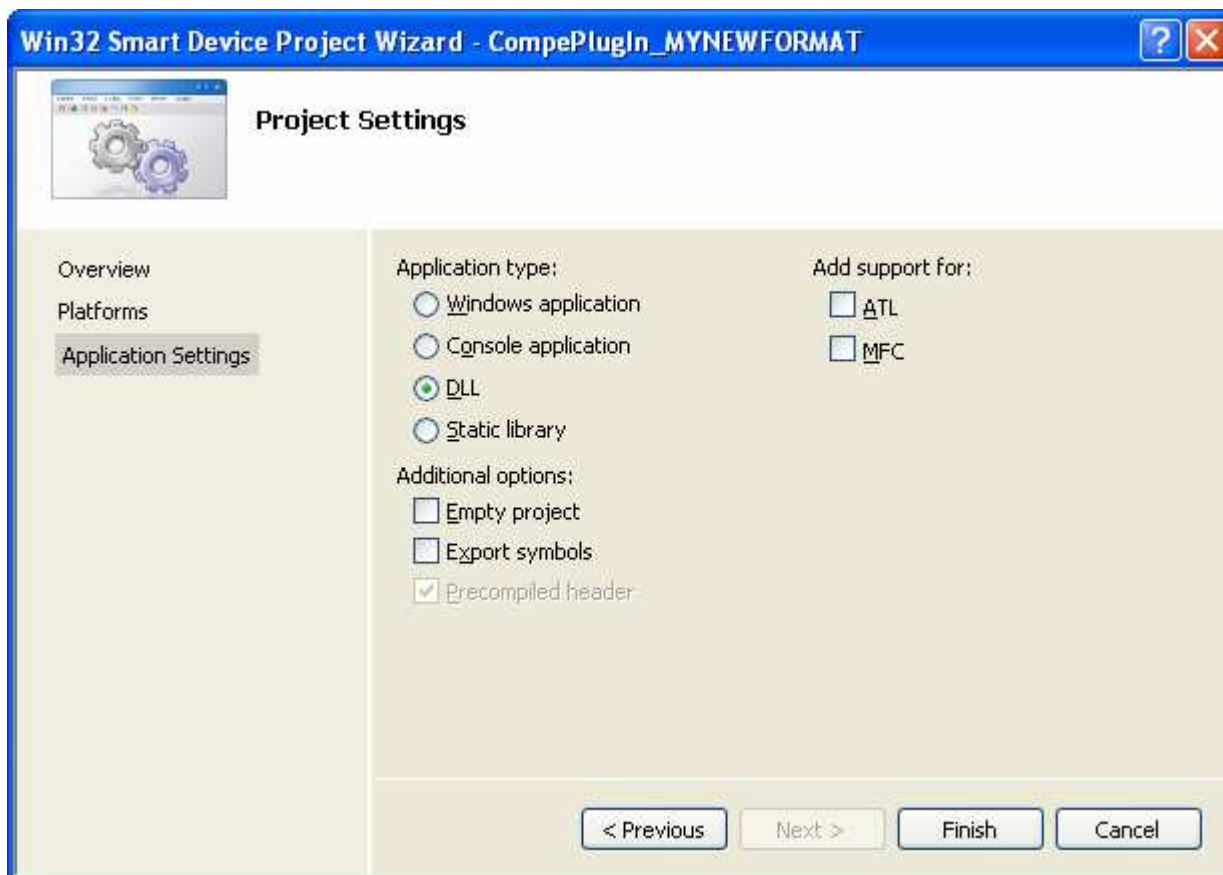
For PocketPC, we recommend to use Microsoft Visual Studio 2005.

1. File > New Project
2. Create a project caeled CompePocket_ *YOURFORMAT*
3. In project type, Select Visual C++ > Smart Device
4. Select “Win32 Smart Device Project”



5. Click on 'OK'

6. Select 'Application Setting', and select 'DLL'



7. Select Finish.

8. Finally, add a c++ module, and write the functions specified in this document.

to make the Plug-ins using Borland C++ Builder version 5.0, 6.0 or 2006, or with Microsoft Visual C++.

2 Creating the version and Init functions

2.1 int CompePlugIn_Version()

This function has to return VERSION_PLUGIN_COMPE.

This way, CompeGPS can now the version of the plugin protocol you are using.

```
int __export CompePlugIn_Version(){
    return VERSION_PLUGIN_COMPE;
};
```

2.2 bool CompePlugIn_Init (int hWnd)

CompeGPS will call this function when loading the DLL. You can make some initialization here if you want.

```
bool __export CompePlugIn_Init (int hWnd){
    return true;
}
```

2.3 const char* CompePlugIn_VersionStr()

In this function, you can write the version, and a description of your plugin.

```
const char* __export CompePlugIn_VersionStr(){
    return "Example BMP plugin reader, version 1.0. "
           "Ivan Twose, July-2003";
};
```

2.4 const char* CompePlugIn_GetTipoArchivos()

This function should return a string with the list of all the formats and features of your plugin. The string must be separated by the | symbol. Each available format is

n1|Name of the format 1|extension 1|

n2|Name of the format 2|extension 2|

.....

n Type of the format. May be one of this

1 : Raster Image Read Only

1001: Raster Images Read & Write

2 : Raster Calibrated map.

3 : DEMs

4 : Internet address for downloading raster maps

5 : Internet address for downloading DEMs

- 6: Vector Maps
- 7: Competition tasks.
- 8: Tracks
- 9: Waypoints

This is an example for a standard plug-in for the CompeGPS maps, with BMP and HDR dems

```
"1|Bitmap (*.BMP)|BMP|2|CompeGPS Maps (*.imp)|imp|3|DEMS (*.hdr)|hdr"
```

For Internet map downloads, use the third line to indicate the rectangle coordinates in which this plug-in works.

```
"4|Nima Ortho 10m/pixel|35N,10W,45N,3E|5|Nima DEM 1Km|"
```

2.5 void CompePlugIn_SetTempFolder (....)

```
void EXPORTW CompePlugIn_SetTempFolder (const char *nom_folder);
```

CompeGPS will call this function just after loading the DLL. Some internet maps, save temporal files on disk. Here CompeGPS suggests a folder to save this maps.

3 Bitmap Functions

3.1 bool CompePlugIn_Abre_Raster (.....);

```
bool __export CompePlugIn_Abre_Raster (
    const char *mapa_name,
    TDatosDibPlugIn& datos,
    TStringPlugIn& err);
```

CompeGPS calls this function when it has to open a raster file.

const char *mapa_name: File Image Name.

TDatosDibPlugIn &datos: Object where you have to store the bitmap header data. In this function, you must fill this variables:

`datos.px` Total pixel width of the image

`datos.py` Total pixel height of the image

`datos.read_all_at_once` Set it to true if the image should be readed all at once.

`datos.handle` You can use this int variable as you want. CompeGPS will ignore this value. Here you will probably put a pointer to your private information.

`datos.AdvancedData` Here you must store advanced data, for example the size of the blocks, if the map should be opened with a cache, etc.

TStringPlugIn& err In case of error, you can return here a string with the error message, and return false. In case of no error, just ignore this parameter.

The function should return `true` if ok, or `false` if there is an error.

3.2 Class TDatosDibPlugIn

This class is used in the three raster functions.

```
class TDatosDibPlugIn{
public:
    TDatosDibPlugIn() {handle=0;nplugin=-1;};
    // Only used by the PlugIns:
    int handle;
    // Use by both
    int px,py;
    bool read_all_at_once;
    // Only used by CompeGPS: plugins should ignore this vaule
    int nplugin;
    // Used by both:
    int bpp;
    TStringPlugIn *AdvancedData;
```

```

    int    *trace; // For debug. do not use.
};

```

3.2.1 AdvancedData

Here, the plugin can give extra information. This string must be separated by lines, and each line, must have the format A=B. This are the possible values of A:

cache_it= default 0. put =1, if you want CompeGPS to use the map cache memory system. It is recommended to use it if the CompePlugIn_Abre_Zona is not very fast.

cache_dx= Width in pixels of the cache cells.

cache_dy= Height in pixels of the cache cells.

cache_multithread= 1 if the plugin CompePlugIn_Abre_Zona function can be called at the same time by several threads at the same time.

constant_palette=1 if all the images returned by have the same palette.

cache_disk= put =1, if you want CompeGPS to use the map cache DISK system. It is recommended if the CompePlugIn_Abre_Zona function gets the map from internet.

concurrent= 1,2, ... 10 The cache system can download more than one section at the same time. To do this, specify the maximum number of threads that can be downloading parts of the map at the same time.

barritas= if =1, compegps will show a progress bar on the cells which are still not downloaded. Use this if you want in slow internet maps.

3.3 bool CompePlugIn_Abre_Zona (...)

```

bool CompePlugIn_Abre_Zona (
    TDatosDibPlugIn &datos,
    int left,int right,int top,int bottom,
    int &sizeX,int &sizeY,
    TDatosZonaPlugIn &dib,
    TStringPlugIn &err,
    TMonitorProcessPlugIn &monitor);

```

CompeGPS calls this function when it needs a portion or the whole bitmap image.

TDatosDibPlugIn &datos: Raster image object, created by a previous call to the function CompePlugIn_Abre_Raster function.

int left,int right,int top,int bottom: Rectangle in bitmap coordinates of the zone of interest.

int &sizeX,int &sizeY: Size of the desired bitmap image.

TDatosZonaPlugIn &dib: Bitmap object. You must write the bitmap here.

TStringPlugIn& err In case of error, you can return here a string with the error message, and return false. In case of no error, just ignore this parameter.

`TMonitorProcessPlugIn &monitor` If the reading process is very slow, you can use this object to tell CompeGPS how this slow process is going... CompeGPS uses this information to show a progress bar.

To write the bits into the `TDatosZonaPlugIn` structure, there are two ways:

1. Writing bits directly.
2. Specifying a path, and saving the image into this path. This image should be in JPG, GIF, TIF or PNG, but it could also be in any other bitmap format which other CompeGPS plugins could read.

3.4 `bool CompePlugIn_Cierra_Raster (...)`

```
void EXPORTW CompePlugIn_Cierra_Raster (TDatosDibPlugIn &datos);
```

This function is called by CompeGPS when the map has to be closed.

4 Map Functions

CompeGPS calls to this function to read the calibration of a map:

```
bool __export CompePlugIn_Lee_Calibracion(  
    const char *nom_mapa,  
    char **fichero_IMP2,  
    TStringPlugIn& err);
```

const char *nom_mapa: File Map Name.

char **fichero_IMP2: char pointer to a string where you must write the calibration of the map in the IMP2 CompeGPS format.

TStringPlugIn& err: In case of error, you can return here a string with the error message, and return false. In case of no error, just ignore this parameter.

The function should return true if ok, or false if there is an error.

5 Bitmap Saving Functions

If the plugin can also save new maps into it's bitmap format, CompeGPS will call this functions to save the raster map:

5.1 bool CompePlugIn_Save_Raster_Create (.....);

```
bool EXPORTW CompePlugIn_Save_Raster_Create (const char* bitmap_name,
                                             TDatosSaveDibPlugIn &dib,
                                             TStringPlugIn &err);
```

CompeGPS will call this function to start saving a raster map.

const char* bitmap_name The name of the new bitmap.

TDatosSaveDibPlugIn dib A structure, with the general properties of the new bitmap, like width, height, etc. The plugin can modify some if this properties, for example the bits per pixel.

TStringPlugIn& err In case of error, you can return here a string with the error message, and return false. In case of no error, just ignore this parameter.

5.2 TDatosSaveDibPlugIn

This class stores the information of the new bitmap:

```
class TDatosSaveDibPlugIn{
public:
    TDatosSaveDibPlugIn() {size=sizeof(*this);str_imp2=NULL;};
    int size; // Size of this structure
    // Internal used only by CompeGPS
    int nplugin;
    // Input values
    int w,h;
    int bpp_in;
    int num_colors;
    int *palette;
    // in/out Values
    // Block size: The number of lines must be proportional to it!.
    int xbl;
    int ybl;
    int bpp_out;
    int dir_y; //+1 y goes upwards.(zero at bottom) -1 y goes down.
    (zero at top)
    int cal_ok;
    float compresion;
```

```

    const char *str_imp2; // Calibration to fit into the image.
    // Internal plug-in values (ignored by CompeGPS)
    int handle;
};

```

CompeGPS will fill this structure with this data:

size, w,h,bpp_in, num_colors, paleta

The Plugin, can modify this values:

xbl, ybl, bpp_out, dir_y

xbl, **ybl** are the cell size, in pixels.

bpp_out is output bits per pixel.

dir_y is the direction of the y axis. It should be -1. (-1 means that the 0 is in the top. +1 means that the zero is in the bottom)

5.3 bool CompePlugIn_Save_Raster_AddLines (.....);

```

bool EXPORTW CompePlugIn_Save_Raster_AddLines (TDatosSaveDibPlugIn& dib,
                                                TSaveLinesDibPlugIn& SomeLines,
                                                TMonitorProcessPlugIn &monitor,
                                                TStringPlugIn &err);

```

CompeGPS will call one time or more to this function, to add the bits to the image. If the image is very little, it will be called only one time.

```

TDatosSaveDibPlugIn& dib,
TSaveLinesDibPlugIn& SomeLines,
TMonitorProcessPlugIn &monitor,

```



TStringPlugIn& err In case of error, you can return here a string with the error message, and return false. In case of no error, just ignore this parameter.

5.4 bool CompePlugIn_Save_Raster_Close (.....);

```

bool EXPORTW CompePlugIn_Save_Raster_Close (TDatosSaveDibPlugIn &dib,
                                             bool canceled,
                                             TMonitorProcessPlugIn &monitor,

```

```
TStringPlugIn &err);
```

6 DEM Functions

6.1 bool __export CompePlugIn_Lee_DEM (.....);

CompeGPS calls to this function to read a DEM (Digital Elevation Model)

```
bool __export CompePlugIn_Lee_DEM (
    const char * dem_name,
    TDemPlugIn &dem,
    TStringPlugIn &err,
    TMonitorProcessPlugIn *monitor,
    bool read_altitudes);
```

const char * dem_name: Name of the DEM file to open

TDemPlugIn &dem: Object with the information of all the DEM.

TStringPlugIn& err :In case of error, you can return here a string with the error message, and return false. In case of no error, just ignore this parameter.

TMonitorProcessPlugIn *monitor

bool read_altitudes: Boolean value: True: This function must read all the elevation data.

False: Do not read the elevation data, this fuction only has to return the HDR and PRJ values of the dem structure.

6.2 class TDemPlugIn

```
class TDemPlugIn{
public:
    TStringPlugIn *file_HDR;
    TStringPlugIn *file_PRJ;
    char* alts;
    void CreateAlts(int len);
};
```

The CompePlugIn_Lee_DEM function must:

- Fill the file_HDR element, by calling dem.file_HDR->Set (....
- Fill the file_PRJ element, by calling dem.file_PRJ->Set (....
- If (read_altitudes), then:
Call CreateAlts, to allocate space in alts for the dem.
Fill in the alts vector with the altitudes.

6.3 Example

```

bool __export CompePlugIn_Lee_DEM (
    const char * dem_name,
    TDemPlugIn &dem,
    TStringPlugIn &err,
    TMonitorProcessPlugIn *monitor,
    bool read_altitudes){
    if (strncmpi(nomf,"ALPS",4)!=0 ){
        err.Set("This plugin only reads 'ALPS' DEM files");
        return false;
    };
    if (nomf.strlen()!=8){
        err.Set ("Wrong DEM file name");
        return false;
    };
    if (read_altitudes){
        dem.CreateAlts(2*258*258);
        short int *pdem = (short int*) (dem.alts);
        FILE *fp = fopen (nomd.str,"rb");
        // Read all the elevation data, and store it in pdem..
        //....
        fclose(fp);
    };

    double res = 75;
    if (Dstrncmpi(nomf,"ALP3",4)==0 ){
        res = 30;
    };
    // Finally, create the two files: HDR - PRJ
    double xlam = 438715;
    double ylam = 1774477;
    DString str_HDR;
    str_HDR = "BYTEORDER      I\r\n"
              "LAYOUT          BIL\r\n";
    str_HDR << "NROWS          " << 258 << "\r\n";
    str_HDR << "NCOLS           " << 258 << "\r\n";
    str_HDR << "NBANDS          1\r\n";
    str_HDR << "NBITS           16\r\n";
    str_HDR << "NODATA          -9999\r\n";
    str_HDR << "ULXMAP          " << xlam << "\r\n";
    str_HDR << "ULYMAP          " << ylam << "\r\n";
    str_HDR << "XDIM            " << res << "\r\n";
    str_HDR << "YDIM            " << -res << "\r\n";
    int len = str_HDR.strlen();
    dem.file_HDR->Set(str_HDR);
    // Rellenar el PRJ
    DString str_PRJ;
    ///!!str_PRJ = "Projection\t France Lambert II étendu\r\n"
    str_PRJ = ("Projection\t Lambert ALPS\r\n"
              "Datum\t          NTF\r\n");

```

```
        "Zunits\t      METERS\r\n"  
        "Zone\t        30\r\n"  
        "Units\t      METERS\r\n");  
dem.file_PRJ->Set(str_PRJ);  
return true;  
};
```

7 Vector Maps Functions

CompeGPS uses plugin dlls to read vector maps since version 6.3. Version 6.3 includes two vector maps plugins: SHP and a E00.

A VectorMap plugin, should implement this functions:

7.1 bool CompePlugIn_Open_Vector_Map (...)

```
bool EXPORTW CompePlugIn_Open_Vector_Map (
    const char*          map_name,
    TVectorMapPlugIn&   vectormap ,
    TStringPlugIn&      err );
```

This function is called when a vector map is going to be opened

const char* map_name The file name of the vector map.

TVectorMapPlugIn& vectormap A structure, with the general properties of the vector map. Here be carefull with this variables: read, write, and create. This tree values can be 0 (false) or 1 (true). If this vector map can be saved by random access, set the 'random_saving' to 1. Else, let it as zero. A map with random saving, can call the saving functions in any order.

TStringPlugIn& err In case of error, you can return here a string with the error message, and return false. In case of no error, just ignore this parameter.

7.2 class TVectorMapPlugIn

```
class TVectorMapPlugIn{
public:
    TVectorMapPlugIn();
    int size;
    // Data used by both
    const char *path_exe; // from CompeGPS to plug-in
    int read; // from CompeGPS to plug-in
    int write; // from CompeGPS to plug-in
    int create; // from CompeGPS to plug-in
    int random_saving; // from plug-in to CompeGPS
    // Data used by CompeGPS
    int nplugin;
    // Data used by the PlugIn:
    int handle;
};
```

7.3 bool CompePlugIn_Read_Vector_Map3 (...)

```
bool EXPORTW CompePlugIn_Read_Vector_Map3 (
    TVectorMapPlugIn&          vectormap ,
    TReadVectorMapPlugIn&    readvectormap ,
    TStringPlugIn&           err,
    TMonitorProcessPlugIn &monitor);
```

This function reads the whole map. The **readvectormap** structure, has three functions, that must be called by the plugin, to tell CompeGPS there are new layers, new polilines, and new points for a polyline.

```
class TReadVectorMapPlugIn {
public:
    TReadVectorMapPlugIn ();
    int size; // Size of this structure.
    // Use by both
    TStringPlugIn *file_PRJ;
    int NewLayer (TLineAttributesPlugIn &line_attr);
    int NewLine (TLineAttributesPlugIn &line_attr);
    int NewPoint (int lineid, TVectorPointPlugIn &point,
        bool last_point);
    // Only used by CompeGPS
    . . . . .
    // Extra opening params
    const char* search_str;
};
```

Take a look to our SHP plugin example to see how this function should work.

7.4 bool CompePlugIn_Close_Vector_Map (...)

```
bool EXPORTW CompePlugIn_Close_Vector_Map(TVectorMapPlugIn&          vectormap,
    TStringPlugIn& _err);
```

This function is called by CompeGPS to close a vector map.

7.5 `bool CompePlugIn_Add_Layer (...)`

SORRY, THIS FUNCTION ARE STILL NOT DOCUMENTED!!!!!!!!!!!!!!1

```
bool EXPORTW CompePlugIn_Add_Layer (  
    TVectorMapPlugIn& vectormap ,  
    TLineAttributesPlugIn &Line,  
    TStringPlugIn& err );
```

7.6 `bool CompePlugIn_Add_Poliline_Vector_Map (...)`

SORRY, THIS FUNCTION ARE STILL NOT DOCUMENTED!!!!!!!!!!!!!!1

```
bool EXPORTW CompePlugIn_Add_Poliline_Vector_Map(  
    TVectorMapPlugIn& vectormap ,  
    TLineAttributesPlugIn &Line,  
    int npoints,  
    TVectorPointPlugIn *points,  
    TStringPlugIn& err );
```

7.7 `bool CompePlugIn_Delete_Poliline_Vector_Map (...)`

SORRY, THIS FUNCTION ARE STILL NOT DOCUMENTED!!!!!!!!!!!!!!1

```
bool EXPORTW CompePlugIn_Delete_Poliline_Vector_Map(  
    TVectorMapPlugIn& vectormap,  
    int dababaseid, TStringPlugIn& err );
```


8 Track Functions

```
bool EXPORTW CompePlugIn_Read_Track (const char *trackname,  
                                     TStringPlugIn& igcout,  
                                     TStringPlugIn &err);
```

This function is called to read tracks. The plugin, must translate the track from it's original format to IGC format, and write it into the **igcout** string.

const char *trackname The name of the track to open.

TStringPlugIn& igcout The function must store here the translated to IGC track.

TStringPlugIn &err In case of error, you can return here a string with the error message, and return false. In case of no error, just ignore this parameter.

9 Waypoints Functions

```
bool EXPORTW CompePlugIn_Read_Wpts (const char *wptsname,  
                                     TStringPlugIn& wptout,  
                                     TStringPlugIn &err);
```

This function is called to read waypoints. The plugin, must translate the waypoints from it's original format to the CompeGPS WPT format, and write it into the **wptout** string.

const char *wptsname The name of the waypoints file to open.

TStringPlugIn& wptout The function must store here the waypoints, translated to WPT format.

TStringPlugIn &err In case of error, you can return here a string with the error message, and return false. In case of no error, just ignore this parameter.