Christine-GIS

Public interface for raster data plug-ins

Specification of public interface version 1.3

Christine-GIS

Released: August 2012
Revised: October 2018

Content

1. Introduction

Document describes interface for dynamic linked libraries, which can be used for reading and drawing spatial raster data. The interface allows you create your own library and in this way you can extent Christine's list of known raster formats. If you create your own library, it is strictly recommended to create an installation program for installing the library. See *Installation of your own library* chapter for some hints.
Document describes version 1.3 of the interface. The interface is supported by Christine-GIS version 4.01, build 4.120831 and higher. Christine-GIS Viewer does not support this interface. The material in this document reflects the information available at the time of publication and is essentially identical to the information contained in the Christine's help file. In some cases, more up-to-date information may be available at Christine's web site (www.christine-gis.com).

2. Used data types

Most of used data types are common data types used by Microsoft Windows API.

BITMAP* - A pointer to bitmap structure which defines the type, width, height, color format, and bit values of a bitmap

```
typedef struct tagBITMAP {
    LONG   bmType;        // Specifies the bitmap type (always zero)
    LONG   bmWidth;       // Specifies the width, in pixels, of the bitmap
    LONG   bmHeight;      // Specifies the height, in pixels, of the bitmap
    LONG   bmWidthBytes;  // Specifies the number of bytes in each scan line
    WORD   bmPlanes;      // Specifies the count of color planes (always 1)
    WORD   bmBitsPixel;   // Specifies the number of bits required
                          // to indicate the color of a pixel (always 24)
    LPVOID bmBits;        // Pointer to the array of the values for the bitmap
} BITMAP;
```

Each 3-byte triplet in the array of values for the bitmap represents the relative intensities of blue, green, and red, respectively, for a pixel. Bitmap data are in bottom-up order. It means that the first scan line in the bitmap data is the last scan line to be displayed.

BOOL - A boolean type (should be TRUE or FALSE), stored as signed integer

COLORREF - A 32-bit value used to specify a RGB color. The low-order byte contains a value for the relative intensity of red; the second byte contains a value for green; and the third byte contains a value for blue. The high-order byte must be zero. The maximum value for a single byte is 0xFF.

CRS2CRS – A pointer to function for conversion between coordinate systems. Pointer to the function is declared in following way:

```
typedef BOOL (__cdecl *CRS2CRS)(int, int, double*, double*, double*)
```

Function returns TRUE if conversion is successful done, otherwise returns FALSE. First parameter of the function is count of points for conversion. Second parameter is offset between coordinates in array of coordinates. Third, fourth and fifth parameters are pointers to arrays of x, y and z coordinates.

double - A 64-bit signed floating point number

double* - A pointer to 64-bit signed floating point number

DWORD - A 32-bit unsigned integer

EXTENT - Structure that defines ranges of X, Y, Z coordinates and M values

```
typedef struct _EXTENT {
    double xMin;       // specifies minimum value of X coordinate
    double yMin;       // specifies minimum value of Y coordinate
    double xMax;       // specifies maximum value of X coordinate
    double yMax;       // specifies maximum value of Y coordinate
    double zMin;       // specifies minimum value of Z coordinate (ignored)
    double zMax;       // specifies maximum value of Z coordinate (ignored)
    double mMin;       // specifies minimum of M value (ignored)
    double mMax;       // specifies maximum of M value (ignored)
} EXTENT;
```

EXTENT* - A pointer to extent structure

FALSE - Boolean value, stored as signed integer with zero value

HDC - Handle to a device context

HWND - Handle to a window

int - A 32-bit signed integer

LONG - A 32-bit signed integer

NULL - Zero value

RECT - Structure that defines the coordinates of the upper-left and lower-right corners of a rectangle

```
typedef struct _RECT {
    LONG left;     // x-coordinate of the upper-left corner of the rectangle
    LONG top;      // y-coordinate of the upper-left corner of the rectangle
    LONG right;    // x-coordinate of the lower-right corner of the rectangle
    LONG bottom;   // y-coordinate of the lower-right corner of the rectangle
} RECT;
```

RECT* - A pointer to rectangle structure

TCHAR* - A pointer to array of chars (string)

TRUE - Boolean value, stored as non zero signed integer

UINT - A 32-bit unsigned integer

void - Used as a function‘s return type, specifies that the function does not return a value

void* - A pointer to unspecified data type or structure ("universal pointer")

WORD - A 16-bit unsigned integer

3. How it works

Christine calls functions of the public interface described in chapter *Public interface of library* when it needs. Library is released after each function call, except calling `void` **`SetColorChanging`**`(DWORD nCount, COLORREF* colors)`, `void` **`SetTransparency`**`(DWORD transparencyPercentage)`, `void` **`SetTransparentColor`**`(COLORREF color)` and `void` **`UseTransparentColor`**`(BOOL bUse)` functions before calling function `void` **`Draw`**`(TCHAR* fileName, HWND hWndMainFrame, HDC hDC, EXTENT* imgExtent, EXTENT* imgExtentUnrotated, double rotation, EXTENT* drawExtent, RECT* clipRect, CRS2CRS Crs2Crs, DWORD* pId1, volatile DWORD* pId2).`

3.1 Adding raster data to a view

Christine looks through Windows registry for available raster formats. Christine uses values of *FormatName*, *FormatExtensions* and *FormatID* keys in order to complete list of known formats and fills combo box in open dialog. If user chooses a raster format that uses the interface, Christine will look for value of *LibFileName* key in Windows registry.

List of called functions during the operation:

```
EXTENT* GetExtent(TCHAR* fileName)
EXTENT* GetUnrotatedExtent(TCHAR* fileName)
double GetRotation (TCHAR* fileName)
```

3.2 Drawing and printing raster data

Christine loads raster library to memory and sets transparency of raster and color that will be treat as transparent. After this it calls Draw function. When drawing operation is done, it releases raster library.

List of called functions during the operation:

```
void Draw(TCHAR* fileName, HWND hWndMainFrame, HDC hDC, EXTENT* imgExtent,
EXTENT* imgExtentUnrotated, double rotation, EXTENT* drawExtent, RECT* clipRect,
CRS2CRS Crs2Crs, DWORD* pId1, volatile DWORD* pId2)
void SetColorChanging(DWORD nCount, COLORREF* colors)
void SetTransparency(DWORD transparencyPercentage)
void SetTransparentColor(COLORREF color)
void UseTransparentColor(BOOL bUse)
```

3.3 Showing properties of theme based on raster data

Before properties dialog is shown Christine calls set of function in order to get all necessary informations.

List of called functions during the operation:

```
DWORD GetColorDeep(TCHAR* fileName)
TCHAR* GetCompressionTypeString(TCHAR* fileName)
EXTENT* GetExtent(TCHAR* fileName)
TCHAR* GetFormatName(TCHAR* fileName)
TCHAR* GetFormatVersion(TCHAR* fileName)
DWORD GetHeight(TCHAR* fileName)
DWORD GetInterfaceVersion()
DWORD GetWidth(TCHAR* fileName)
```

```
TCHAR* GetWorldFileName(TCHAR* fileName)
BOOL HasPyramid(TCHAR* fileName)
BOOL SupportsColorChanging()
BOOL SupportsTransparency()
BOOL SupportsTransparentColor()
BOOL SupportsPyramids()
```

3.4 Exporting content of a view to raster file

Christine looks through Windows registry for available raster formats. Christine looks for value of *LibFileName* key in Windows registry and loads dynamic-linked library. After this it calls SupportsExport function and releases library. If the function returns TRUE, Christine will use values of *FormatName*, *FormatExtensions* and *FormatID* keys in order to add the format to export dialog. If user chooses a raster format that uses the interface, Christine will look for value of *LibFileName* key in Windows registry again, loads library and calls Export function. After export process is done, Christine will release library.

List of called functions during the operation:

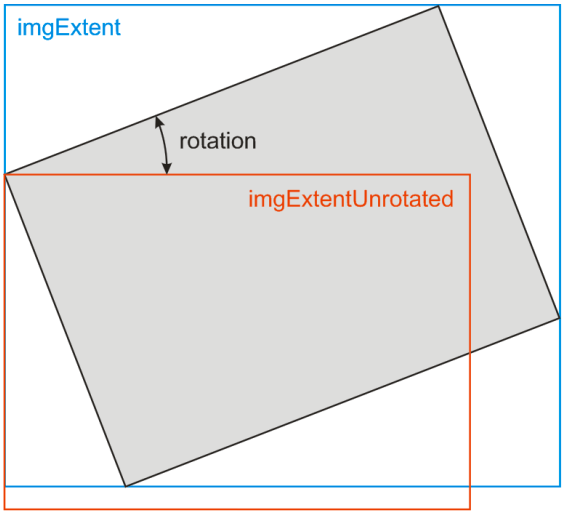
```
EXTENT* GetExtent(TCHAR* fileName)
BOOL Export(TCHAR* outFileName, HWND hWndMainFrame, BITMAP* pBitmap, EXTENT*
exportedExtent, UINT mapUnits, BOOL bCreateWorldFile)
BOOL SupportsExport()
```

4. Public interface of library

This chapter describes public interface of the library. Each function in the public interface has exactly defined behaviour and it is called in situations described in chapter How it works.

`BOOL` **`BuildPyramid`**`(TCHAR* fileName)`
Builds pyramid for raster data. Pyramid is set of small resampled and georeferenced images that allow Christine speeder draw raster data in small scales. Returns TRUE if pyramid was built successfully.

`void` **`Draw`**`(TCHAR* fileName, HWND hWndMainFrame, HDC hDC, EXTENT* imgExtent, EXTENT* imgExtentUnrotated, double rotation, EXTENT* drawExtent, RECT* clipRect, CRS2CRS Crs2Crs, DWORD* pId1, volatile DWORD* pId2)`
Draws raster data on device context with handle hDC. Parameter fileName is full file name of raster data file. Parameter hWndMainFrame is handle to Christine's main window. Parameter imgExtent is extent of whole raster data in map units (in fact it is extent of theme based on the raster data). Parameter imgExtentUnrotated is extent of unrotated raster data in map units. For nonrotated raster data is parameter imgExtentUnrotated equivalent of parameter imgExtent. Parameter rotation is angle of rotation of raster data in radians. Parameter drawExtent is extent for drawing in map units. Parameter clipRect is clipping rectangle in screen units. Parameter Crs2Crs is pointer to function for conversion between coordinate systems. If no conversion is needed parameter is NULL. Parameters pId1 and pId2 are pointers to numeric identificators. If the values of identificators are the same you can continue drawing, but if pId2 is changed you should stop drawing immediately and clean up the memory.



`BOOL` **`Export`**`(TCHAR* outFileName, HWND hWndMainFrame, BITMAP* pBitmap, EXTENT* exportedExtent, UINT mapUnits, BOOL bCreateWorldFile)`
Converts bitmap to desired format and saves it to a file. Parameter outFileName is full file name for the file. Parameter hWndMainFrame is handle to Christine's main window. Parameter pBitmap is pointer to BITMAP structure. Parameter exportedExtent is extent of exported data in map units. You will need to know extent of exported data in order to be able to create world file. Parameter mapUnits specifies map units, it can be one of following values:

Unknown map units	*40244*
Kilometers	*40245*
Meters	*40246*
Decimeters	*40253*
Centimeters	*40247*
Millimeters	*40248*
Miles	*40249*
Feet	*40250*
Inches	*40251*
Yards	*40254*
Nautical Miles	*40255*
Fathoms	*40256*
Chains	*40257*
Links	*40258*
US Surveyor's Inches	*40259*
US Surveyor's Feet	*40260*
US Surveyor's Yards	*40261*
US Surveyor's Chains	*40262*
US Surveyor's Statute Miles	*40263*
Indian Feet	*40264*
Indian Yards	*40265*
Indian Chains	*40266*

Decimal Degrees *40252*
If parameter bCreateWorldFile is TRUE function should create world file or georeference exported image in other way, depend on desired raster format. Returns TRUE if file was written successfully, otherwise returns FALSE.

`DWORD` **`GetColorDeep`**`(TCHAR* fileName)`
Returns color deep of raster data. Parameter fileName is full file name of raster data file.

`TCHAR*` **`GetCompressionTypeString`**`(TCHAR* fileName)`
Returns pointer to string with name of compression metod. If data have no compression, returns “none”. Parameter fileName is full file name of raster data file.

`EXTENT*` **`GetExtent`**`(TCHAR* fileName)`
Returns pointer to unprojected extent of raster. Parameter fileName is full file name of raster data file.

`TCHAR*` **`GetFormatName`**`(TCHAR* fileName)`
Returns pointer to string with name of data format. Parameter fileName is full file name of raster data file.

`TCHAR*` **`GetFormatVersion`**`(TCHAR* fileName)`
Returns pointer to string with version of raster data format. Parameter fileName is full file name of raster data file.

`DWORD` **`GetHeight`**`(TCHAR* fileName)`
Returns height of raster data in pixels. Parameter fileName is full file name of raster data file.

`DWORD` **`GetInterfaceVersion`**`()`
Returns version of the interface. In high word is major version (1) and in low word is minor version (3).

`double` **`GetRotation`**`(TCHAR* fileName)`
Returns rotation angle of raster in radians. Parameter fileName is full file name of raster data file.

`EXTENT*` **`GetUnrotatedExtent`**`(TCHAR* fileName)`
Returns pointer to unprojected and unrotated extent of raster. xMin member of EXTENT structure contains rotation angle in radians. Parameter fileName is full file name of raster data file.

`DWORD` **`GetWidth`**`(TCHAR* fileName)`
Returns width of raster data in pixels. Parameter fileName is full file name of raster data file.

`TCHAR*` **`GetWorldFileName`**`(TCHAR* fileName)`
Returns pointer to string with full name of world file. Returns NULL if raster data have no world file. Parameter fileName is full file name of raster data file.

`BOOL` **`HasPyramid`**`(TCHAR* fileName)`
Returns TRUE if raster data have built pyramid. Parameter fileName is full file name of raster data file.

`void` **`RemovePyramid`**`(TCHAR* fileName)`
Removes pyramid. Parameter fileName is full file name of raster data file.

`void` **`SetColorChanging`**`(DWORD nCount, COLORREF* colors)`
Sets values for color changing. Parameter nCount is number of colors in array colors. Parameter colors is array of pairs of colors for changing. Odd item in pair is original color, even item is new color. It is used before Draw function is called.

`void` **`SetTransparency`**`(DWORD transparencyPercentage)`
Sets transparency to transparencyPercentage. It is used before Draw function is called.

`void` **`SetTransparentColor`**`(COLORREF color)`
Sets transparent color to color. It is used before Draw function is called.

`BOOL` **`SupportsColorChanging`**`()`
Returns TRUE if the library supports color changing, otherwise returns FALSE.

`BOOL` **`SupportsExport`**`()`
Returns TRUE if the library supports export functionality, otherwise returns FALSE.

`BOOL` **`SupportsTransparency`**`()`
Returns TRUE if the library supports transparency, otherwise returns FALSE.

`BOOL` **`SupportsTransparentColor`**`()`
Returns TRUE if the library supports transparent color, otherwise returns FALSE.

`BOOL` **`SupportsPyramids`**`()`
Returns TRUE if the library supports pyramids, otherwise returns FALSE.

`void` **`UseTransparentColor`**`(BOOL bUse)`
If parameter bUse is TRUE, transparent color should be used.

4.1 Minimal functional requirements

Minimal requirement is to draw raster data in a view and to print the raster data to printer. To create functional library that do this you must implement following functions with described behaviour:

`BOOL` **`BuildPyramid`**`(TCHAR* fileName)`
Returns FALSE.

`void` **`Draw`**`(TCHAR* fileName, HWND hWndMainFrame, HDC hDC, EXTENT* imgExtent, EXTENT* imgExtentUnrotated, double rotation, EXTENT* drawExtent, RECT* clipRect, CRS2CRS Crs2Crs, DWORD* pId1, volatile DWORD* pId2)`
Draws raster data on device context with handle hDC.

`BOOL` **`Export`**`(TCHAR* outFileName, HWND hWndMainFrame, BITMAP* pBitmap, EXTENT* exportedExtent, UINT mapUnits, BOOL bCreateWorldFile)`
Returns FALSE.

`DWORD` **`GetColorDeep`**`(TCHAR* fileName)`
Returns color deep of raster data.

`TCHAR*` **`GetCompressionTypeString`**`(TCHAR* fileName)`
Returns pointer to string with name of compression metod. If data have no compression, returns „none“.

`EXTENT*` **`GetExtent`**`(TCHAR* fileName)`
Returns pointer to unprojected extent of raster data.

`TCHAR*` **`GetFormatName`**`(TCHAR* fileName)`
Returns pointer to string with name of data format.

`TCHAR*` **`GetFormatVersion`**`(TCHAR* fileName)`
Returns version of raster data format.

`DWORD` **`GetHeight`**`(TCHAR* fileName)`
Returns height of raster data in pixels.

`DWORD` **`GetInterfaceVersion`**`()`
Returns version of the interface. In high word is major version (1) and in low word is minor version (3).

`double` **`GetRotation`**`(TCHAR* fileName)`
Returns rotation angle of raster data in radians.

`EXTENT*` **`GetUnrotatedExtent`**`(TCHAR* fileName)`
Returns pointer to unprojected and unrotated extent of raster data.

`DWORD` **`GetWidth`**`(TCHAR* fileName)`
Returns width of raster data in pixels.

`TCHAR*` **`GetWorldFileName`**`(TCHAR* fileName)`
Returns pointer to string with full name of world file. Returns NULL if raster data have no world file.

`BOOL` **`HasPyramid`**`(TCHAR* fileName)`
Returns FALSE.

`void` **`RemovePyramid`**`(TCHAR* fileName)`
Does nothing.

`void` **`SetColorChanging`**`(DWORD nCount, COLORREF* colors)`
Does nothing.

`void` **`SetTransparency`**`(DWORD transparencyPercentage)`
Does nothing.

`void` **`SetTransparentColor`**`(COLORREF color)`
Does nothing.

`BOOL` **`SupportsColorChanging`**`()`
Returns FALSE.

`BOOL` **`SupportsExport`**`()`
Returns FALSE.

`BOOL` **`SupportsTransparency`**`()`
Returns FALSE.

`BOOL` **`SupportsTransparentColor`**`()`
Returns FALSE.

`BOOL` **`SupportsPyramids`**`()`
Returns FALSE.

`void` **`UseTransparentColor`**`(BOOL bUse)`
Does nothing.

5. Installation of your own library

It is strictly recommended to create an installation program for installing the library. Installation program must place library file to subfolder dlls in Christine's installation folder. Installation program must also write following informations to Windows registry:

Key:
HKEY_LOCAL_MACHINE\Software\Christine-GIS\5.x\RasterFormats\[ID of format]

Values:

Name	Value	Type	Max. length (byte)	
LibFileName	[library.dll]	REG_SZ	255	Mandatory
FormatExtensions	[*.xxx;*.xxxx]	REG_SZ	63	Mandatory
FormatName	Name of Image File Format	REG_SZ	255	Mandatory
FormatID	[ID of format]	REG_DWORD	4	Mandatory
MineType	[image/*]	REG_SZ	63	Optional

Example for Tag Image File Format:

Key:
HKEY_LOCAL_MACHINE\Software\Christine-GIS\5.x\RasterFormats\3

Values:

Name	Value
LibFileName	tiff.dll
FormatExtensions	*.tif;*.tiff
FormatName	Tag Image File Format
FormatID	3
MineType	image/tiff

Note: Value *FormatID* must be the same as name of the key. Optional value of *MineType* is neccessary if the raster format can be used as output format of an internet map server.

5.1 Format identification number (ID)

If you want to create your own library for reading raster data, look at Christine's web site for free format IDs. After you choose a free format ID, contact us and we will publish your choosen ID on Christine's web site to avoid problems with duplicity. Format ID is used internally by Christine and by these metods in Christine's script language: GetType(Number nType) from class Theme, ExportToBmp(String sFullFileName, Number nType, Rect extent, Number nWidth, Number nHeight, Bool bCreateWorldFile) and AddTheme(String sFullFileName, Number nType) from class View, ExportToVectorFormat(String sFullFileName, Number nFormatID, Bool bUseDefaultSettings) from class FTheme, Export(String sFullFileName, Number nOutputType) from class DSCTheme, ShowBmp(String sFullFileName, Number nDelay, Bool bFrame, Number nType) from class MsgBox.

Recommended ranges of format IDs are following:

0 - 1000 for raster and vector data based on files (typically stored on a file system), range 0 - 100 is reserved for Christine-GIS, 101 - 1000 is available for third-party developers

1001 - 2000 for raster and vector data based on a web service (typically generated by a map server and sent to Christine), range 1001 - 1100 is reserved for Christine-GIS, 1101 - 2000 is available for third-party developers

2001 - 3000 for raster and vector data stored in a geodatabase (for future use, not supported yet), range 2001 - 2100 is reserved for Christine-GIS, 2101 - 3000 is available for third-party developers

6. Libraries that come with Christine

Following libraries come with Christine-GIS version 5.x, that have implemented this interface. All these libraries are placed in subfolder dlls in Christine's installation folder.

Format name:	**Tag Image File Format**
Library file name:	tiff.dll
Format ID:	3
Format extensions:	*.tif;*.tiff
MINE type:	image/tiff
Version of interface:	1.3
Supports color changing:	Yes
Supports export:	Yes
Supports transparency:	Yes
Supports transparent color:	Yes
Supports pyramids:	Yes

Format name:	**Join Photographic Experts Group**
Library file name:	jpeg.dll
Format ID:	5
Format extensions:	*.jpg;*.jpeg
MINE type:	image/jpeg
Version of interface:	1.3
Supports color changing:	Yes
Supports export:	Yes
Supports transparency:	Yes
Supports transparent color:	Yes
Supports pyramids:	Yes

Format name:	**Portable Network Graphics**
Library file name:	png.dll
Format ID:	7
Format extensions:	*.png
MINE type:	image/png
Version of interface:	1.3
Supports color changing:	Yes
Supports export:	Yes
Supports transparency:	Yes
Supports transparent color:	Yes
Supports pyramids:	Yes

Format name:	**ER Mapper ECW**
Library file name:	ecw.dll
Format ID:	8
Format extensions:	*.ecw
Version of interface:	1.3
Supports color changing:	Yes
Supports export:	Yes
Supports transparency:	Yes
Supports transparent color:	Yes
Supports pyramids:	Yes

Format name:	**MrSID**
Library file name:	mrsid.dll
Format ID:	9
Format extensions:	*.sid
Version of interface:	1.3
Supports color changing:	Yes
Supports export:	No

Supports transparency:	Yes
Supports transparent color:	Yes
Supports pyramids:	Yes

Format name:	**JPEG 2000**
Library file name:	jpeg2000.dll
Format ID:	10
Format extensions:	*.jp2
MINE type:	image/jp2
Version of interface:	1.3
Supports color changing:	Yes
Supports export:	No
Supports transparency:	Yes
Supports transparent color:	Yes
Supports pyramids:	Yes

Format name:	**Graphics Interchange Format**
Library file name:	gif.dll
Format ID:	11
Format extensions:	*.gif
MINE type:	image/gif
Version of interface:	1.3
Supports color changing:	Yes
Supports export:	Yes
Supports transparency:	Yes
Supports transparent color:	Yes
Supports pyramids:	Yes

Format name:	**Intergraph Raster File**
Library file name:	ingr.dll
Format ID:	12
Format extensions:	*.cit;*.g3;*.g4
Version of interface:	1.3
Supports color changing:	Yes
Supports export:	No
Supports transparency:	Yes
Supports transparent color:	Yes
Supports pyramids:	Yes

7. Example

In this chapter we offer you C++ source code of an empty framework of dynamic linked library that uses the interface. Code is tested in Microsoft Visual Studio 2005 enviroment.

7.1 How to compile the example

This short chapter describes neccessary steps to compile source code in chapter 7.2. using Microsoft Visual Studio 2005. Simply follow the instructions bellow.

- Run Microsoft Visual Studio 2005.
- From menu File select New and then choose Project. From categories of project types select Win32 in Visual C++ section. From installed templates select Win32 Project. Choose location for new project that will be created and fill project name, then click OK button. In following two steps wizard select „DLL" option and click Finish button.
- A basic main source file projectName.cpp is created.
- Copy source code from chapter 7.2. to created file.
- From Project menu choose projectName Properties (Alt + F7). Select Configuration Properties/General section and set „Character Set" option to „Not Set". Now select Configuration Properties/C/C++/Preprocesor and add to Preprocessor Definitions „_CRT_SECURE_NO_WARNINGS".
- From Build menu choose Build Solution command (F7).

7.2 Source code

```
#include "stdafx.h"
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>

typedef BOOL (__cdecl *CRS2CRS)(int, int, double*, double*, double*);

typedef struct _EXTENT {    // extent of data
   double xMin;
   double yMin;
   double xMax;
   double yMax;
   double zMin;
   double zMax;
   double mMin;
   double mMax;
} EXTENT;

// map units
enum {
    map_units_unknown = 40244,
    map_units_kilometers = 40245,
    map_units_meters = 40246,
    map_units_decimeters = 40253,
    map_units_centimeters = 40247,
    map_units_millimeters = 40248,
    map_units_inches = 40251,
    map_units_feet = 40250,
    map_units_yards = 40254,
    map_units_statute_miles = 40249,
    map_units_nautical_miles = 40255,
    map_units_fathoms = 40256,
    map_units_chains = 40257,
    map_units_links = 40258,
    map_units_us_surveyors_inches = 40259,
    map_units_us_surveyors_feet = 40260,
```

```
    map_units_us_surveyors_yards = 40261,
    map_units_us_surveyors_chains = 40262,
    map_units_us_surveyors_statute_miles = 40263,
    map_units_indian_feet = 40264,
    map_units_indian_yards = 40265,
    map_units_indian_chains = 40266,
    map_units_decimal_degrees = 40252
};

// private variables for each thread
typedef struct _THREADPARAMS {
        TCHAR           m_buffer[_MAX_PATH];
        BOOL            bUseTransparentColor;
        COLORREF        transparentColor;
        int                     transparency;
        COLORREF*       colorChanges;
        DWORD           colorChangesCount;
        EXTENT          extent;
} THREADPARAMS;

HINSTANCE hInstanceDll;
static DWORD   dwTlsIndex; // address of shared memory

BOOL WINAPI DllMain(HINSTANCE hInstDll, DWORD fdwReason, LPVOID lpReserved)
{
    LPVOID lpvData;
        // Perform actions based on the reason for calling.
        switch(fdwReason)
        {
                case DLL_PROCESS_ATTACH:
                        // Initialize once for each new process.
                        // Return FALSE to fail DLL load.

                        // It is always a good idea to save handle of instance of library
                        hInstanceDll = hInstDll;

                        // Allocate a TLS index.
                        if ((dwTlsIndex = TlsAlloc()) == TLS_OUT_OF_INDEXES)
                                return FALSE;
                        // No break: Initialize the index for first thread.

                case DLL_THREAD_ATTACH:
                        // Do thread-specific initialization.
                        // Initialize the TLS index for this thread.
                        lpvData = (LPVOID)new THREADPARAMS;
                        if (lpvData == NULL) return FALSE;
                        if (!TlsSetValue(dwTlsIndex, lpvData)) {
                                delete (THREADPARAMS*)lpvData;
                                return FALSE;
                        }
                        break;

                case DLL_THREAD_DETACH:
                        // Do thread-specific cleanup.
                        // Release the allocated memory for this thread.
                        lpvData = TlsGetValue(dwTlsIndex);
                        if (lpvData != NULL) {
                                THREADPARAMS* threadParams = (THREADPARAMS*)lpvData;
                                if (threadParams) delete threadParams;
                        }
                        break;

                case DLL_PROCESS_DETACH:
                        // Perform any necessary cleanup.
                        lpvData = TlsGetValue(dwTlsIndex);
                        if (lpvData != NULL) {
                                THREADPARAMS* threadParams = (THREADPARAMS*)lpvData;
                                if (threadParams) delete threadParams;
                        }
                        // Release the TLS index.
                        TlsFree(dwTlsIndex);
                        break;
        }
        return TRUE;
}

// Function returns pointer to parameters for current thread
THREADPARAMS* GetThreadParams()
```

```
{
	THREADPARAMS* threadParams = (THREADPARAMS*)TlsGetValue(dwTlsIndex);
	if (!threadParams) {
		threadParams = new THREADPARAMS;
		if (!threadParams) return NULL;
		ZeroMemory(threadParams, sizeof(THREADPARAMS));
		if (!TlsSetValue(dwTlsIndex, (LPVOID)threadParams)) {
			delete threadParams;
			return NULL;
		}
	}
	return threadParams;
}

extern "C" __declspec( dllexport ) BOOL __cdecl BuildPyramid(TCHAR* fileName)
{

	// If not supported or function failed
	return FALSE;

	// Build pyramid here
	// ...
	// insert your code here
	// ...
	// pyramid was successfully built
	// return TRUE;
}

extern "C" __declspec( dllexport ) void __cdecl Draw(TCHAR* fileName, HWND hWndMainFrame, HDC hDC,
EXTENT* imgExtent, EXTENT* imgExtentUnrotated, double rotation, EXTENT* drawExtent, RECT* clipRect,
CRS2CRS Crs2Crs, DWORD* pId1, volatile DWORD* pId2)
{
	// Work until *pId1 is equal to *pId2
	if (*pId1 != *pId2) return;

	SetStretchBltMode(hDC, COLORONCOLOR);

	THREADPARAMS* threadParams = GetThreadParams();
	if (!threadParams) return;

	// ...
	// insert your code here
	// ...
	// Please don't forget about value of pId2. Check the value as often as possible
	// and reasonable. If the value is changed, clean up the memory and leave
	// the function immediately.
}

extern "C" __declspec( dllexport ) BOOL __cdecl Export(TCHAR* outFileName, HWND hWndMainFrame,
BITMAP* pBitmap, EXTENT* exportedExtent, UINT mapUnits, BOOL bCreateWorldFile)
{

	// If not implemented or function failed return FALSE, otherwise return TRUE


	// Convert bitmap to your format and save it to file
	// ...
	// insert your code here
	// ...

	// create world file
	if (bCreateWorldFile) {
		// construct world file name
		TCHAR worldFN[_MAX_PATH];
		lstrcpy(worldFN, outFileName);
		if (TCHAR* pDot = strrchr(worldFN, '.')) pDot[0] = '\0';
		lstrcat(worldFN, ".xxw");
		// create file
		if (FILE* worldFile = fopen(worldFN, "wt")) {
			TCHAR buffer[256];
			sprintf(buffer, "%.14f\n",
(exportedExtent->xMax -exportedExtent->xMin)/(double)pBitmap->bmWidth);
			fwrite(buffer, lstrlen(buffer), 1, worldFile);
			sprintf(buffer, "%.14f\n", 0.0);
			fwrite(buffer, lstrlen(buffer), 1, worldFile);
			fwrite(buffer, lstrlen(buffer), 1, worldFile);
			sprintf(buffer, "%.14f\n",
(-1)*((exportedExtent->yMax - exportedExtent->yMin)/(double)pBitmap->bmHeight));
```

```
                        fwrite(buffer, lstrlen(buffer), 1, worldFile);
                        sprintf(buffer, "%.14f\n", exportedExtent->xMin);
                        fwrite(buffer, lstrlen(buffer), 1, worldFile);
                        sprintf(buffer, "%.14f\n", exportedExtent->yMax);
                        fwrite(buffer, lstrlen(buffer), 1, worldFile);
                        fclose(worldFile);
                }
        }
        return TRUE;
}

extern "C" __declspec( dllexport ) DWORD __cdecl GetColorDeep(TCHAR* fileName)
{
        DWORD colorDeep = 0;

        // Calculate color deep
        // ...
        // insert your code here
        // ...

        return colorDeep;
}

extern "C" __declspec( dllexport ) TCHAR* __cdecl GetCompressionTypeString(TCHAR* fileName)
{
        THREADPARAMS* threadParams = GetThreadParams();
        if (!threadParams) return NULL;
        lstrcpy(threadParams->m_buffer, "none");

        // If raster data are compressed fill the threadParams->m_buffer
        // with name of used type of compression
        // ...
        // insert your code here
        // ...

        return threadParams->m_buffer;
}

extern "C" __declspec( dllexport ) EXTENT* __cdecl GetExtent(TCHAR* fileName)
{
        THREADPARAMS* threadParams = GetThreadParams();
        if (!threadParams) return NULL;
        memset(&(threadParams->extent), 0, sizeof(EXTENT));

        // Fill xMin, yMin, xMax and yMax members of threadParams->extent structure
        // ...
        // insert your code here
        // ...

        return &(threadParams->extent);
}

extern "C" __declspec( dllexport ) TCHAR* __cdecl GetFormatName(TCHAR* fileName)
{
        THREADPARAMS* threadParams = GetThreadParams();
        if (!threadParams) return NULL;
        threadParams->m_buffer[0] = '\0';

        // Fill the threadParams->m_buffer with name of raster format
        // ...
        // insert your code here
        // ...

        return threadParams->m_buffer;
}

extern "C" __declspec( dllexport ) TCHAR* __cdecl GetFormatVersion(TCHAR* fileName)
{
        THREADPARAMS* threadParams = GetThreadParams();
        if (!threadParams) return NULL;
        threadParams->m_buffer[0] = '\0';

        // Fill the threadParams->m_buffer with version of raster format
        // ...
        // insert your code here
        // ...

        return threadParams->m_buffer;
```

```
}

extern "C" __declspec( dllexport ) DWORD __cdecl GetHeight(TCHAR* fileName)
{
	DWORD height = 0;

	// Set height variable to height of raster data in pixels
	// ...
	// insert your code here
	// ...

	return height;
}

extern "C" __declspec( dllexport ) DWORD __cdecl GetInterfaceVersion()
{
	// return interface version (1.2)
	return MAKELONG(2, 1);
}

extern "C" __declspec( dllexport ) DWORD __cdecl GetWidth(TCHAR* fileName)
{
	DWORD width = 0;

	// Set width variable to width of raster data in pixels
	// ...
	// insert your code here
	// ...

	return width;
}

extern "C" __declspec( dllexport ) TCHAR* __cdecl GetWorldFileName(TCHAR* fileName)
{
	// If raster data have no world file return NULL
	// return NULL;

	THREADPARAMS* threadParams = GetThreadParams();
	if (!threadParams) return NULL;
	threadParams->m_buffer[0] = '\0';

	// If raster data have world file fill the threadParams->m_buffer
	// with the full file name of world file
	// ...
	// insert your code here
	// ...

	return threadParams->m_buffer;
}

extern "C" __declspec( dllexport ) BOOL __cdecl HasPyramid(TCHAR* fileName)
{
	// If not supported, or pyramid is not built return FALSE.
	// If pyramid is built return TRUE.
	// ...
	// insert your code here
	// ...

	return FALSE;
}

extern "C" __declspec( dllexport ) void __cdecl RemovePyramid(TCHAR* fileName)
{
	// ...
	// insert your code here
	// ...
}

extern "C" __declspec( dllexport ) void __cdecl SetColorChanging(DWORD nCount, COLORREF* colors)
{
	THREADPARAMS* threadParams = GetThreadParams();
	if (!threadParams) return;
	// Set color changing
	threadParams->colorChanges = colors;
	threadParams->colorChangesCount = nCount;
}

extern "C" __declspec( dllexport ) void __cdecl SetTransparency(DWORD transparencyPercentage)
```

```
{
	THREADPARAMS* threadParams = GetThreadParams();
	if (!threadParams) return;
	// Set transparency percentage.
	if (transparencyPercentage < 0) transparencyPercentage = 0;
	if (transparencyPercentage > 100) transparencyPercentage = 100;
	threadParams->transparency = transparencyPercentage;
}

extern "C" __declspec( dllexport ) void __cdecl SetTransparentColor(COLORREF color)
{
	THREADPARAMS* threadParams = GetThreadParams();
	if (!threadParams) return;
	// Set transparent color.
	threadParams->transparentColor = color;
}

extern "C" __declspec( dllexport ) BOOL __cdecl SupportsColorChanging()
{
	// if not supported return FALSE, otherwise return TRUE
	return FALSE;
}

extern "C" __declspec( dllexport ) BOOL __cdecl SupportsExport()
{
	// if not supported return FALSE, otherwise return TRUE
	return FALSE;
}

extern "C" __declspec( dllexport ) BOOL __cdecl SupportsTransparency()
{
	// if not supported return FALSE, otherwise return TRUE
	return FALSE;
}

extern "C" __declspec( dllexport ) BOOL __cdecl SupportsTransparentColor()
{
	// if not supported return FALSE, otherwise return TRUE
	return FALSE;
}

extern "C" __declspec( dllexport ) BOOL __cdecl SupportsPyramids()
{
	// if not supported return FALSE, otherwise return TRUE
	return FALSE;
}

extern "C" __declspec( dllexport ) void __cdecl UseTransparentColor(BOOL bUse)
{
	THREADPARAMS* threadParams = GetThreadParams();
	if (!threadParams) return;
	threadParams->bUseTransparentColor = bUse;
}
```