DFL Software

Light Lib Multimedia 1.0

DLL Functions

We strongly suggest that you use the extensive support classes and functions provided for the individual languages instead of calling the DLLs directly. The following DLL functions are provided for reference and should not be called directly unless you have a thorough understanding of how to use them.

Light Lib Multimedia

mGet()
mPut()

Light Lib Objects

oAccess()
oAssign()
oNew()
oDel()

Constants

DLL Constants
Light Lib Objects Constants

System Constants

LLM_CLASS_APPLICATION
LLM_CLASS_MULTIMEDIA

LLM_APPLICATION_IDLE := 0x03000001 // LLM_CLASS_APPLICATION + 0x0001
LLM_APPLICATION_VERSION := 0x03000002 // LLM_CLASS_APPLICATION + 0x0002

Constants

```
LLM_AVI_COORD_BOTTOM    := 0x03020004 // LLM_CLASS_AVI + 0x0004
LLM_AVI_COORD_HEIGHT    := 0x03020007 //LLM_CLASS_AVI + 0x0007
LLM_AVI_COORD_LEFT      := 0x03020002 //LLM_CLASS_AVI + 0x0002
LLM_AVI_COORD_MAXIMIZE  := 0x03020008 //LLM_CLASS_AVI + 0x0008
LLM_AVI_COORD_RESIZE    := 0x03020009 //LLM_CLASS_AVI + 0x0009
LLM_AVI_COORD_RIGHT     := 0x03020005 //LLM_CLASS_AVI + 0x0005
LLM_AVI_COORD_TOP       := 0x03020003 //LLM_CLASS_AVI + 0x0003
LLM_AVI_COORD_WIDTH     := 0x03020006 //LLM_CLASS_AVI + 0x0006
LLM_AVI_NAME            := 0x03020001// LLM_CLASS_AVI + 0x0001
LLM_AVI_SIZE          := 0x03020010 //LLM_CLASS_AVI + 0x001
```

Introduction

Welcome to Light Lib Multimedia!

Light Lib Multimedia was designed to be the easiest multimedia library available for Windows application developers. Light Lib Multimedia is highly optimized for very fast processing.

Light Lib Multimedia's amazing speed is based on excellent internal image processing.

Light Lib Multimedia was developed with the following goals in mind:

Ease of Use — It is very easy to integrate Light Lib Multimedia into existing applications. There exists less than a dozen core functions which provide the power needed to manage multimedia applications.

Execution speed — Execution speed is excellent.

Language Support — Providing DLLs is not enough. All Light Lib products for Windows come with extensive language support which make it effortless to introduce Light Lib libraries using established and familiar syntax. Light Lib products are the best solution if you need to develop applications with various Windows languages. There is no need to learn different libraries because Light Lib products provide support for the following WIndows application development languages,

Borland C++
Microsoft C/C++
Microsoft Visual Basic
CA-Visual Objects

Light Lib Multimedia is comprised of a small set of core functions which provide all of the necessary services Above the core is the language support layer which provides simple interface to the core functions in the desired language. Language support is provided MS-Visual Basic, CA-Visual Objects, and C/C++ Windows Development Systems..

No help available for this section.

Using Light Lib Multimedia

The easiest way to learn any new concept is by example. Each installed language has the source code to its own set of demonstration programs. Please reference them to gain a good understanding of how to use Light Lib Multimedia.

If you are familiar with object-oriented programming, you will find it useful to implement OOP concepts.

How to use this Help

This help system was designed to provide quick access to information. Help is provided for the extensive language support and for the supplied Light Lib Multimedia and Light Lib Objects DLLs.

We strongly suggest that you use the individual language support with your applications!

When help for a language is selected, you will be prompted with an overview of all support classes and/or functions. There is also a "How Do I?" section which provides step-by-step instructions on various common tasks.

A secondary window will open containing details and descriptions when any of these items are selected. This window is set to always stay on top. That way once a help topic is selected, you can continue working without losing focus on this window. To close it, simply select the window's system menu and select Close.

Quick Start

See the sample application are provided for each supported language. You should execute the sample application and experiment with the features in order to gain a good understanding of how Light Lib Multimedia works.

Overview

Light Lib Multimedia is an easy to implement multimedia library.

Readme

Compatibility

Windows Screen Drivers

Light Lib Multimedia is compatible with all installed Windows screen drivers.

Windows Printer Drivers

Light Lib Multimedia is compatible with all installed Windows printer drivers.

Callback Functions

Light Lib Multimedia uses "Callback" functions to provide your applications with the ability to do something while data is processed.

Referencing an object from a Callback Function

In general, multimedia data is attached to objects (for example a window), but when Light Lib Multimedia executes a callback function which is not a method, the reference to "self" is lost. In this case, we suggest that "self" be passed to one of the user defined parameters. This gives your callback function a reference to "self" which provides access to all the methods and instance variables. This is very useful object oriented programming.

BLOBs

Light Lib Multimedia supports BLOB (Binary Large OBjects) data formats in that you are able to convert multi media data into a BLOB and vice versa. This allows you to store multi media data to files which support BLOBs.

BLOB's are supported internally at this time. In the near future, the API will be published.

What's New...

Appendices

Dithering Techniques
Stripping Algorithm

CA-Visual Objects Functions and Classes

The CA-Visual Objects support AEF supplied with Light Lib Multimedia should not be modified directly since this support layer calls the Light Lib Objects DLL directly. Functions which should never be modified are explicitly labeled in the AEF itself.

Classes
LightLibAvi
LightLibMIDI
LightLibWav
SoundPlayWindowControl
SoundRecordWindowControl
SoundWindow
VideoPlayWindowControl
VideoWindow

Functions
dwLightLibApp()
dwLightLibAppRegister()
dwLightLibAppUnRegister()

Samples
Simple Sound In Window

How Do I? CA-Visual Objects

General
Add sound to my application
Add video to my application

System
Register and unregister an application

How Do I? CA-Visual Objects

Add sound to my application

How Do I? CA-Visual Objects

Add video to my application

How Do I? CA-Visual Objects

How Do I? CA-Visual Objects

LightLibAVI Class

Purpose

Properties
CoordinateBottom Access/Assign
CoordinateLeft Access/Assign
CoordinateMaximize Access/Assign
CoordinateRight Access/Assign
CoordinateTop Access/Assign
FileName Access/Assign
IsPlaying Access
Modal Access/Assign
Name Access/Assign
OnError Access/Assign
Size Access/Assign

Methods
CoordinateResize()
Destroy()
Information()
Init()
Pause()
Play()
SaveAs()
Stop()

System Properties
*These properties are used internally. The are provided as reference only and should **NEVER** be accessed directly in your applications.*
cbOnError Export
oWindowOwner Export

Inherits From

(No ancestors)

Inherited By

(No descendants)

LightLibAVI:CoordinateBottom Access/Assign

Description

Type

LightLibAVI:CoordinateLeft Access/Assign

Description

Type

LightLibAVI:CoordinateMaximize Access/Assign

Description

Type

LightLibAVI:CoordinateRight Access/Assign

Description

Type

LightLibAVI:CoordinateTop Access/Assign

Description

Type

LightLibAVI:FileName Access/Assign

Description

Type

LightLibAVI:IsPlaying Access

Description

Type

LightLibAVI:Modal Access/Assign

Description

Type

LightLibAVI:Name Access/Assign

Description

Type

LightLibAVI:OnError Access/Assign

Description

Type

LightLibAVI:Size Access/Assign

Description

Type

LightLibAVI:cbOnError Export

Description

Type

LightLibAVI:dwSelf Export

Description

Type

LightLibAVI:oWindowOwner Export

Description

Type

LightLibAVI:CoordinateReSize() Method

Purpose

Maximize the coordinates

Syntax

<oLightLibAVI>:CoordinateReSize() ---> NIL

Arguments

None

Returns

NIL

LightLibAVI:Destroy() Method

Purpose

Destroy the LigthLibAVI object.

Syntax

<oLightLibAVI>:Destroy() ---> NIL

Arguments

None

Returns

NIL

LightLibAVI:Information() Method

Purpose

Display a dialog containing information about the AVI data.

Syntax

<oLightLibAVI>:Information() ---> NIL

Arguments

None

Returns

NIL

LightLibAVI:Init() Method

Purpose

Create a new LightLibAVI object

Syntax

LightLibAVI{ <*oWindowOwner*>, <*sFileName*> } ---> SELF

Arguments

<*oWindowOwner*>	Reference to the owner window
<*sFileName*>	AVI file to load

Returns

SELF

LightLibAVI:Pause() Method

Purpose

Pause playing of the AVI

Syntax

<oLightLibAVI>:Pause() ---> NIL

Arguments

None

Returns

NIL

LightLibAVI:Play() Method

Purpose

Play the AVI

Syntax

<oLightLibAVI>:Play() ---> NIL

Arguments

None

Returns

NIL

LightLibAVI:SaveAs() Method

Purpose

Save the loaded AVI

Syntax

<oLightLibAVI>:SaveAs(<*sFileName*>) ---> NIL

Arguments

<*sFileName*> File name.

Returns

NIL

LightLibAVI:Stop() Method

Purpose

Stop the playing of the AVI

Syntax

<oLightLibAVI>:Stop() ---> NIL

Arguments

None

Returns

NIL

LightLibMIDI Class

Purpose

Properties
FileName Access
IsPlaying Access
IsRecording Access
Modal Access/Assign
Name Access/Assign
OnError Access/Assign
Size Access

Methods
Destroy()
Information()
Init()
Pause()
Play()
SaveAs()
Stop()

System Properties
*These properties are used internally. The are provided as reference only and should **NEVER** be accessed directly in your applications.*
cbOnError Export

Inherits From

(No ancestors)

Inherited By

(No descendants)

LightLibMIDI:FileName Access

Description

Name of the MIDI file

Type

STRING

LightLibMIDI:IsPlaying Access

Description

Type

LOGICAL

LightLibMIDI:IsRecording Access

Description

Type

LOGICAL

LightLibMIDI:Modal Access/Assign

Description

Type

LightLibMIDI:Name Access/Assign

Description

Name of the MIDI file

Type

STRING

LightLibMIDI:OnError Access/Assign

Description

Type

LightLibMIDI:Size Access

Description

Size of the loaded MIDI file.

Type

INTEGER

LightLibMIDI:cbOnError Export

Description

Type

LightLibMIDI:Destroy() Method

Purpose

Destroy the LightLibMIDI object

Syntax

<oLightLibMIDI>:Destroy() ---> NIL

Arguments

None

Returns

NIL

LightLibMIDI:Information() Method

Purpose

Display a dialog containing information about the loaded MIDI file.

Syntax

<oLightLibMIDI>:Information() ---> NIL

Arguments

None

Returns

NIL

LightLibMIDI:Init() Method

Purpose

Create a new LightLibMIDI object

Syntax

LightLibMIDI{ <*sFileName*> } ---> SELF

Arguments

<*sFileName*>	File name

Returns

SELF

LightLibMIDI:Pause() Method

Purpose

Pause

Syntax

<oLightLibMIDI>:Pause() ---> NIL

Arguments

None

Returns

NIL

LightLibMIDI:Play() Method

Purpose

Play the loaded MIDI file

Syntax

<oLightLibMIDI>:Play() ---> NIL

Arguments

None

Returns

NIL

LightLibMIDI:SaveAs() Method

Purpose

Save the loaded MIDI file to disk.

Syntax

<oLightLibMIDI>:SaveAs(<*sFileName*>) ---> NIL

Arguments

<*sFileName*> File name

Returns

NIL

LightLibMIDI:Stop() Method

Purpose

Syntax

<oLightLibMIDI>:Stop() ---> NIL

Arguments

None

Returns

NIL

LightLibWAV Class

Purpose

Properties
FileName Access
IsPlaying Access
IsRecording Access
Name Access/Assign
OnError Access/Assign
Size Access

Methods
Destroy()
Information()
Init()
Pause()
Play()
Record()
SaveAs()
Stop()

System Properties
*These properties are used internally. The are provided as reference only and should **NEVER** be accessed directly in your applications.*
cbOnError Export

Inherits From

(No ancestors)

Inherited By

(No descendants)

LightLibWAV:FileName Access

Description

Name of the WAV file

Type

STRING

LightLibWAV:IsPlaying Access

Description

Type

LOGICAL

LightLibWAV:IsRecording Access

Description

Type

LOGICAL

LightLibWAV:Name Access/Assign

Description

Name of the WAV file

Type

STRING

LightLibWAV:OnError Access/Assign

Description

Type

LightLibWAV:Size Access

Description

Size of the loaded WAV file.

Type

INTEGER

LightLibWAV:cbOnError Export

Description

Type

LightLibWAV:dwSelf Export

Description

Type

LightLibWAV:Destroy() Method

Purpose

Destroy the LightLibWAV object

Syntax

<oLightLibWAV>:Destroy() ---> NIL

Arguments

None

Returns

NIL

LightLibWAV:Information() Method

Purpose

Display a dialog containing information about the loaded WAV file.

Syntax

<oLightLibWAV>:Information() ---> NIL

Arguments

None

Returns

NIL

LightLibWAV:Init() Method

Purpose

Create a new LightLibWAV object

Syntax

LightLibWAV{ <*sFileName*> } ---> SELF

Arguments

<*sFileName*>	File name

Returns

SELF

LightLibWAV:Pause() Method

Purpose

Syntax

<oLightLibWAV>:Pause() ---> NIL

Arguments

None

Returns

NIL

LightLibWAV:Play() Method

Purpose

Play the loaded WAV file

Syntax

<oLightLibWAV>:Play() ---> NIL

Arguments

None

Returns

NIL

LightLibWAV:Record() Method

Purpose

Syntax

<oLightLibWAV>:Record() ---> NIL

Arguments

None

Returns

NIL

LightLibWAV:SaveAs() Method

Purpose

Save the loaded WAV file to disk.

Syntax

<oLightLibWAV>:SaveAs(<*sFileName*>) ---> NIL

Arguments

<*sFileName*>	File name

Returns

NIL

LightLibWAV:Stop() Method

Purpose

Syntax

<oLightLibWAV>:Stop() ---> NIL

Arguments

None

Returns

NIL

SoundWindow Class

Purpose

Properties
None

Methods
Close()
Destroy()
FileExit()
Information()
Init()
Load()
Open()
Pause()
Play()
Record()
SaveAs()
Stop()

System Properties
*These properties are used internally. The are provided as reference only and should **NEVER** be accessed directly in your applications.*
oSound Export

Inherits From

(No ancestors)

Inherited By

(No descendants)

SoundWindow:oSound Export

Description

Type

SoundWindow:Close() Method

Purpose

Close the window

Syntax

<oSoundWindow>:Close(<*oEvent*>) ---> NIL

Arguments

<*oEvent*>	Event object

Returns

NIL

SoundWindow:Destroy() Method

Purpose

Destroy the SoundWindow object

Syntax

<oSoundWindow>:Destroy() ---> NIL

Arguments

None

Returns

NIL

SoundWindow:FileExit() Method

Purpose

Post a message to close the application

Syntax

<oSoundWindow>:FileExit() ---> NIL

Arguments

None

Returns

NIL

SoundWindow:Information() Method

Purpose

Display a dialog containing information about the loaded WAV file.

Syntax

<oSoundWindow>:Information() ---> NIL

Arguments

Returns

Description

SoundWindow:Init() Method

Purpose

Create a SoundWindow object.

Syntax

<oSoundWindow>:Init(<*oParentWindow*>, <*sFileName*>) ---> SELF

Arguments

<*oParentWindow*>	Parent window
<*sFileName*>	WAV file name

Returns

SELF

SoundWindow:Load() Method

Purpose

Load a WAV file

Syntax

<oSoundWindow>:Load(<*sFileName*>) ---> NIL

Arguments

<*sFileName*> WAV file name to load

Returns

NIL

SoundWindow:Open() Method

Purpose

Display the Open File Dialog for WAV files

Syntax

<oSoundWindow>:Open() ---> NIL

Arguments

None

Returns

NIL

Description

If the Open File Dialog returns a file name, it is loaded.

SoundWindow:Pause() Method

Purpose

Syntax

<oSoundWindow>:Pause() ---> NIL

Arguments

None

Returns

NIL

SoundWindow:Play() Method

Purpose

Play the WAV file

Syntax

<oSoundWindow>:Play() ---> NIL

Arguments

None

Returns

NIL

SoundWindow:Record() Method

Purpose

Syntax

<oSoundWindow>:Record() ---> NIL

Arguments

None

Returns

NIL

SoundWindow:SaveAs() Method

Purpose

Save the loaded WAV file to disk

Syntax

<oSoundWindow>:SaveAs() ---> NIL

Arguments

None

Returns

NIL

SoundWindow:Stop() Method

Purpose

Syntax

<oSoundWindow>:Stop() ---> NIL

Arguments

None

Returns

NIL

SoundPlayWindowControl Class

Purpose

Properties
None

Methods
Destroy()
Init()
Load()
MouseButtonDown()
MouseButtonUp()
Play()
RegisterLightLibDataWindow()
Stop()

System Properties
*These properties are used internally. The are provided as reference only and should **NEVER** be accessed directly in your applications.*
oOwner Export
oSound Export

Inherits From

(No ancestors)

Inherited By

(No descendants)

SoundPlayWindowControl:oOwner Export

Description

Type

SoundPlayWindowControl:oSound Export

Description

Type

SoundPlayWindowControl:Destroy() Method

Purpose

Destroy a SoundPlayWindowControl object.

Syntax

<oSoundPlayWindowControl>:Destroy() ---> NIL

Arguments

None

Returns

NIL

SoundPlayWindowControl:Init() Method

Purpose

Create a SoundPlayWindowControl object.

Syntax

SoundPlayWindowControl{ <*oParentWindow*>, <*iCtrlID*> } ---> SELF

Arguments

<*oParentWindow*>	Parent window
<*iCtrlID*>	Control ID

Returns

SELF

Description

This will create a PushButton object capable of playing sound.

SoundPlayWindowControl:Load() Method

Purpose

Load a WAV file

Syntax

<oSoundPlayWindowControl>:Load(<*sFileName*>) ---> NIL

Arguments

<*sFileName*> File name to load

Returns

NIL

SoundPlayWindowControl:MouseButtonDown() Method

Purpose

Syntax

<oSoundPlayWindowControl>:MouseButtonDown(<*oMouseEvent*>) ---> NIL

Arguments

<*oMouseEvent*>

Returns

NIL

SoundPlayWindowControl:MouseButtonUp() Method

Purpose

Syntax

<oSoundPlayWindowControl>:MouseButtonUp(<*oMouseEvent*>) ---> NIL

Arguments

<*oMouseEvent*>

Returns

NIL

SoundPlayWindowControl:Play() Method

Purpose

Play the loaded WAV file

Syntax

<oSoundPlayWindowControl>:Play() ---> NIL

Arguments

None

Returns

NIL

SoundPlayWindowControl:RegisterLightLibDataWindowClient() Method

Purpose

Register the SoundPlayWindowControl control with the LightLibDataWindow object

Syntax

<oSoundPlayWindowControl>:RegisterLightLibDataWindow(<*oDataFieldName*>) ---> NIL

Arguments

<*oDataFieldName*> Field name containing the WAV data.

Returns

NIL

SoundPlayWindowControl:Stop() Method

Purpose

Syntax

<oSoundPlayWindowControl>:Stop() ---> NIL

Arguments

None

Returns

NIL

SoundRecordWindowControl Class

Purpose

Properties
None

Methods
Destroy()
Init()
Load()
MouseButtonDown()
MouseButtonUp()
Play()
Record()
RegisterLightLibDataWindow()
Stop()

System Properties
*These properties are used internally. The are provided as reference only and should **NEVER** be accessed directly in your applications.*
oOwner Export
oSound Export

Inherits From

(No ancestors)

Inherited By

(No descendants)

SoundRecordWindowControl:oOwner Export

Description

Type

OBJECT

SoundRecordWindowControl:oSound Export

Description

Type

OBJECT

SoundRecordWindowControl:Destroy() Method

Purpose

Syntax

<oSoundRecordWindowControl>:Destroy() ---> NIL

Arguments

None

Returns

NIL

SoundRecordWindowControl:Init() Method

Purpose

Syntax

SoundRecordWindowControl{ <*oParentWindow*>, <*iCtrlID*> } ---> SELF

Arguments

<*oParentWindow*>	Parent window
<*iCtrlID*>	Control ID number

Returns

SELF

SoundRecordWindowControl:Load() Method

Purpose

Syntax

<oSoundRecordWindowControl>:Load(<*sFileName*>) ---> NIL

Arguments

<*sFileName*> File name to load

Returns

NIL

SoundRecordWindowControl:MouseButtonDown() Method

Purpose

Syntax

<oSoundRecordWindowControl>:MouseButtonDown(<*oMouseEvent*>) ---> NIL

Arguments

<*oMouseEvent*> Mouse event object

Returns

NIL

SoundRecordWindowControl:MouseButtonUp() Method

Purpose

Syntax

<oSoundRecordWindowControl>:MouseButtonUp(<*oMouseEvent*>) ---> NIL

Arguments

<*oMouseEvent*>	Mouse event object

Returns

NIL

SoundRecordWindowControl:Play() Method

Purpose

Syntax

<oSoundRecordWindowControl>:Play() ---> NIL

Arguments

None

Returns

NIL

SoundRecordWindowControl:Record() Method

Purpose

Syntax

<oSoundRecordWindowControl>:Record() ---> NIL

Arguments

None

Returns

NIL

SoundRecordWindowControl:RegisterLightLibDataWindowClient() Method

Purpose

Syntax

<oSoundRecordWindowControl>:RegisterLightLibDataWindow(<*cDataFieldName*>) ---> NIL

Arguments

<*cDataFieldName*> Name of the data field.

Returns

NIL

SoundRecordWindowControl:Stop() Method

Purpose

Syntax

<oSoundRecordWindowControl>:Stop() ---> NIL

Arguments

None

Returns

NIL

VideoPlayWindowControl Class

Purpose

Properties
BottomLeft Export
TopRight Export

Methods
Destroy()
DisplayBorder()
Expose()
Init()
Load()
MouseButtonDown()
MouseButtonUp()
Play()
RegisterLightLibDataWindowClient()
Stop()

System Properties
*These properties are used internally. The are provided as reference only and should **NEVER** be accessed directly in your applications.*
oOwner Export
oVideo Export

Inherits From

(No ancestors)

Inherited By

(No descendants)

VideoPlayWindowControl:BottomLeft Export

Description

Type

VideoPlayWindowControl:TopRight Export

Description

Type

VideoPlayWindowControl:oOwner Export

Description

Type

VideoPlayWindowControl:oVideo Export

Description

Type

VideoPlayWindowControl:Destroy() Method

Purpose

Syntax

<oVideoPlayWindowControl>:Destroy() ---> NIL

Arguments

None

Returns

NIL

VideoPlayWindowControl:DisplayBorder() Method

Purpose

Syntax

<oVideoPlayWindowControl>:DisplayBorder( ) ---> NIL

Arguments

None

Returns

NIL

VideoPlayWindowControl:Expose() Method

Purpose

Syntax

<oVideoPlayWindowControl>:Expose() ---> NIL

Arguments

None

Returns

NIL

VideoPlayWindowControl:Init() Method

Purpose

Syntax

VideoPlayWindowControl{ } ---> SELF

Arguments

None

Returns

SELF

VideoPlayWindowControl:Load() Method

Purpose

Syntax

<oVideoPlayWindowControl>:Load( ) ---> NIL

Arguments

None

Returns

NIL

VideoPlayWindowControl:MouseButtonDown() Method

Purpose

Syntax

<oVideoPlayWindowControl>:MouseButtonDown(<*oMouseEvent*>) ---> NIL

Arguments

<*oMouseEvent*>	Mouse event object

Returns

NIL

VideoPlayWindowControl:MouseButtonUp() Method

Purpose

Syntax

<oVideoPlayWindowControl>:MouseButtonUp(<*oMouseEvent*>) ---> NIL

Arguments

<*oMouseEvent*> Mouse event object

Returns

NIL

VideoPlayWindowControl:Play() Method

Purpose

Syntax

<oVideoPlayWindowControl>:Play() ---> NIL

Arguments

None

Returns

NIL

VideoPlayWindowControl:RegisterLightLibDataWindowClient() Method

Purpose

Syntax

<oVideoPlayWindowControl>:RegisterLightLibDataWindow(<*cDataFieldName*>) ---> NIL

Arguments

<*cDataFieldName*> Data Field to be registered

Returns

NIL

VideoPlayWindowControl:Stop() Method

Purpose

Syntax

<oVideoPlayWindowControl>:Stop() ---> NIL

Arguments

None

Returns

NIL

VideoWindow Class

Purpose

Properties
None

Methods
Close()
Destroy()
FileExit()
Information()
Init()
Load()
Open()
Play()
SaveAs()

System Properties
*These properties are used internally. The are provided as reference only and should **NEVER** be accessed directly in your applications.*
oVideo Export

Inherits From

(No ancestors)

Inherited By

(No descendants)

VideoWindow:oVideo Export

Description

Type

VideoWindow:Close() Method

Purpose

Close the window

Syntax

<oVideoWindow>:Close(<*oEvent*>) ---> NIL

Arguments

<*oEvent*>	Close event.

Returns

NIL

VideoWindow:Destroy() Method

Purpose

Destroy the VideoWindow object

Syntax

<oVideoWindow>:Destroy() ---> NIL

Arguments

None

Returns

NIL

VideoWindow:FileExit() Method

Purpose

Post a message to close the application

Syntax

<oVideoWindow>:FileExit() ---> NIL

Arguments

None

Returns

NIL

VideoWindow:Information() Method

Purpose

Display a dialog containing information about the AVI data.

Syntax

<oVideoWindow>:Information() ---> NIL

Arguments

None

Returns

NIL

VideoWindow:Init() Method

Purpose

Create a VideoWindow object

Syntax

VideoWindow{ <*oParentWindow*>, <*sFileName*> } ---> SELF

Arguments

<*oParentWindow*>	Parent window
<*sFileName*>	AVI file name to load

Returns

SELF

VideoWindow:Load() Method

Purpose

Load an AVI file

Syntax

<oVideoWindow>:Load(<*sFileName*>) ---> NIL

Arguments

<*sFileName*> File name to load

Returns

NIL

VideoWindow:Open() Method

Purpose

Open the window

Syntax

<oVideoWindow>:Open() ---> NIL

Arguments

None

Returns

NIL

VideoWindow:Play() Method

Purpose

Play the AVI file

Syntax

<oVideoWindow>:Play() ---> NIL

Arguments

None

Returns

NIL

VideoWindow:SaveAs() Method

Purpose

Save the loaded AVI file to disk.

Syntax

<oVideoWindow>:SaveAs() ---> NIL

Arguments

None

Returns

NIL

mGet() CA-Visual Objects

Purpose

Load a multimedia file.

Syntax

```
mGet(       dwLLOwner   AS DWORD,
            dwDevice    AS DWORD,
            dwFormat    AS DWORD,
            dwParam1    AS DWORD,
            dwParam2    AS DWORD,
            dwParam3    AS DWORD,
            dwParam4    AS DWORD,
            dwParam5    AS DWORD ) --> dwData CallBack
```

Arguments

dwLLOwner	Reference to the Light Lib Multimedia owner object
dwDevice	Device being used. The following are valid LLM_DISK
dwFormat	Format of the operation. The following are valid LLM_DISK_AVI LLM_DISK_WAV
dwParam1	Not used
dwParam2	Not used
dwParam3	Not used
dwParam4	Not used
dwParam5	Not used

Returns

dwData	Multimedia data

mPut() CA-Visual Objects

Purpose

Play or display a multimedia file.

Syntax

```
mPut(       dwLLMMedia    AS DWORD,
            liStart       AS LONGINT,
            liLength AS LONGINT,
            dwDevice      AS DWORD,
            dwFormat      AS DWORD,
            dwParam1      AS DWORD,
            dwParam2      AS DWORD,
            dwParam3      AS DWORD,
            dwParam4      AS DWORD,
            dwParam5      AS DWORD ) --> dwData CallBack
```

Arguments

dwLLOwner	Refernce to the Light Lib Multimedia owner object
liStart	Starting offset.
liLength	Length of the data
dwDevice	Device being used. The following are valid LLM_DISK
dwFormat	Format of the operation. The following are valid LLM_DISK_AVI LLM_DISK_WAV
dwParam1	Not used
dwParam2	Not used
dwParam3	Not used
dwParam4	Not used
dwParam5	Not used

Returns

dwData	Multimedia data

Sample not available yet.

MS-Visual Basic

Functions & Classes

Introduction

Light Lib Multimedia provides several support files for use with MS-Visual Basic.

Files

MS-Visual Basic Functions & Classes

iVBStruct()
iVBString2Num()
iVBNum2String()

C/C++

Functions & Classes

Introduction

Light Lib Multimedia provides several support files for use with C/C++.

Files

C/C++ Functions & Classes

Common Problems and Questions

Tips & Techniques

Editions

Light Lib Multimedia comes in two editions. Light Lib Multimedia and Light Lib Multimedia Pro.

Light Lib Multimedia

Light Lib Multimedia Pro

In addition to the standard features, the Pro edition provides support for more advanced and paperful capabilities.

iBlob2MM() DLL Functions

Purpose

Convert a BLOB structure pointer to a multi media structure pointer.

Syntax

iBlob2Img(*Blob*) -> *ptrMM*

Arguments

Blob	Pointer to a BLOB data structure. It must be the result of //\\

Returns

ptrMM	Pointer to mulit media structure

Description

Light Lib Multimedia stores data in a special format which has minimal memory requirements and is optimized for speed.

Therefore, if you try to save *ptrMM* directly to a BLOB field in a database such as Oracle or equivalent such system, all of the elements will be saved except the element containing the multi media itself. This is because the data is represented in a format known only to Light Lib Multimedia. The solution is to convert *ptrMM* multi media element to a character string and then process the character string as ifit were the multi media data.

Notes

Check the features of the product supporting the BLOBs. Some limitations may apply like 64Kb maximum size. This limitation is often reached when dealing with multi media. If this is the case, simply store the data in separate files on disk and store a reference (file name) to the data.

iPack() DLL Functions

Purpose

Compress a Character string

Syntax

iPack(*cToBeCompressed*) ->*cCompressed*

Arguments

cToBeCompressed	Character string to be compressed

Returns

cCompressed	The compressed character string

Description

This function is used in conjunction with iUnPack()to compress character strings.

There is no direct connetion between compressing data using iPack() and iUnPack() and image manipulation. Butsince many languages need the ability to work with character strings instead of pointers, Light Lib Images includes the ability to compress and uncompress character data which in turn can represent images. These functions are very useful in dealing withlarge images. iPack() uses a LZW algorithm which is efficient on strings with sizes greater than 256 Characters.

iUnPack() DLL Functions

Purpose

Uncompress a compressed Character string.

Syntax

iUnPack(*cCompressed*) -> *cUnCompress*

Arguments

lcCompress	Character string previously compressed using iPack()

Returns

lcUnCompress	An uncompressed Character string of a previously compressed string.

Description

This function is used in conjunction with iPack() to restore compressed Character strings to its original value.

There is no direct connetion between compressing data using iPack() and iUnPack() and image manipulation. But since many languages need the ability to work with character strings instead of pointers, Light Lib Images includes the ability to compress and uncompress character data which in turn can represent images. These functions are very useful in dealing with large images. iPack() uses a LZW algorithm which is efficient on strings with sizes greater than 256 Characters.

Who is DFL? With corporate offices in Toronto, Paris and a research and development center in the south of France, DFL is fast becoming a leading developer of advanced add-on products for Windows and DOS. We are committed to providing the very best tools for serious software development. Let DFL's *Light Lib* family of products help you develop better applications.

Thank you for your support,
The DFL Team.

Light Lib Library User License

This document is a contract between you (the licensee) and DFL.

DFL supplies the software and grants a license for its use. You are totally responsible for choosing to use the software for the desired purpose, as well as for installing and using it, and for the results obtained.

DFL grants no rights for the software other than those explicitly stipulated in this agreement, and reserves all rights not explicitly granted to the licensee.

License

DFL grants to the licensee a non-exclusive and non-transferable right to use the software for an unlimited duration. The licensee may make ONE copy of the software in readable form on a computer or other support for backup purposes.

You have the right to load the software into RAM and use it on a single computer, to install it on the hard disk of the computer, to produce executable files (.EXE) containing the Light Lib software.

You are not required to pay royalties to DFL on the sale of any applications using a Light Lib product. If you expect to sell or distribute more than 500 copies of a commercial product which uses one or more Light Lib products, it is required that you inform DFL and that you grant DFL permission to publicize the fact that your product uses Light Lib technology.

Forbidden

The following are forbidden: sharing of the software on a network: each programmer that uses Light Lib software in part or in whole,must possess his or her own registered license, distribution of the software, decompilation of supplied files, disassembly of supplied files, rewriting and any form of reverse engineering of the software, and any other action or activity involving the software that is not explicitly authorized.

Term

The license shall remain in force until it is cancelled. The licensee may cancel it at any time by destroying the software and all copies. It is also cancelled without notification if the licensee violates any terms or conditions of the present agreement. The licensee agrees to destroy the software and all copies if the agreement is cancelled.

Limited warranty

The software is furnished as is, with no warranty of any sort, explicit or implicit, including, but not restricted to, implicit warranties as to its fitness or sale for any particular use. Any risks stemming fromthe quality and performance of the program are entirely bourne by the licensee. If there is any defect in the program, the licensee (and not DFL or any intermediary) will assume any expenses for help, repair, or correction.

DFL does not guarantee that the functions included in the programs correspond to the needs of the licensee or that the program will function without interruptions or errors.

Nevertheless, DFL warrants that the diskettes on which the program is supplied are without defects in material and packaging for a period of ninety (90) days, to be counted from the date of delivery to the licensee as indicated on the bill of sale.

The only responsibility of DFL and the only compensation due to the licensee are:

1) Replacement of any diskette not in conformity with the limited warranty of DFL and which has been returned to DFL or an intermediary authorized by DFL, with a copy of the licensee's receipt, or,

2) If DFL or an authorized intermediary is unable to supply a diskette free of material or packaging defects the licensee may cancel this agreement by returning the software, and the licensee will be reimbursed.

In no case will DFL be responsible to the licensee for damages of any kind, including loss of profits or savings or direct or indirect loss resulting from use of or inability to use the program, even if DFL or an authorized intermediary of DFL was informed of the possibility of such damages, or of any claim forany third party.

General considerations

You certify that you have read and understood the present agreement, and that you agree to be bound by its terms and conditions. You also recognize that it constitutes the sole and exclusive basis for our contract, replacing any earlier proposal or contract, verbal or written, and any other communication between us relating to the object of the present agreement.

Registered trademarks

All trademarks are the property of their registered owners.

Technical Support

The three primary means of technical support are via Compuserve, FAX and on our BBS. If it is an emergency, you can call or fax us.

North America

DFL Software Inc.
1712 Avenue Road
Box 54616
Toronto, ON, M5M 4N5
CANADA

Voice (416) 789-2223
Fax (416) 789-0204
BBS (416) 784-9712
Compuserve 74723,3321
Internet 74723.3321@compuserve.com

Europe

DFL Europe
39-41, rue de la Saussière
Boulogne
FRANCE

Voice (33 1) 46 05 20 66
Fax (33 1) 46 04 10 39
BBS (33 1) 46 05 26 88
Compuserve 100067,652
Internet 100067.652@compuserve.com

products by DFL

All Light Lib products have been designed and developed to be implemented easily and execute quickly .

Windows
Light Lib Business
Light Lib Images
Light Lib Multimedia

DOS
Light Lib Business
Light Lib Images
Light Lib Graphics

Light Lib Business is a revolutionary graphing library. It provides the unprecedented power to present users with "live" graphs. Your users will now be able to dynamically scroll and interact with graph data as if they were scrolling text data. The days of static graphs are over!

Light Lib Images is the most comprehensive image and document managing library available. Scanning, loading, saving, printing images or documents has never been easier.

Light Lib Multimedia is the easiest-to-use multimedia library for Windows. Adding the ability to play or record sound and display video, will bring your applications to new heights.

Light Lib Graphics for CA-Clipper is the first Replaceable Terminal Driver (RTD) for CA-Clipper. It will immediately transforms your text mode applications into graphic mode.

All Light Lib products for DOS are upward compatible with their Windows counterpart. Each product comes with complete help files and source code to the extensive support functions and classes.

All Light Lib products for CA-Clipper are fully compatible with Real and Protected mode linkers (Exospace, Blinker and Causeway) and each product is fully integrated with CA-Clippper's VMM system.

Light Lib Objects Functions

DLL Functions

oAccess()
oAssign()
oNew()
oDel()

Constants

Constants

Light Lib Objects (LLO)

Light Lib Objects is not another Light Lib product. LLO manages memory allocation and the proper creation and deletion of all objects within the Light Lib DLLs themselves. Every Light Lib product for Windows relies on this support DLL. Please review the specific language implementation carefully because the usage of LLO differs slightly from language to language.

LLO provides object oriented technology to languages that do not support object oriented programming and provides enhanced features to languages that support OOP. In addition to standard OOP features such as inheritance, polymorphism, and encapsulation, LLO implements advanced OOP concepts such as inheriting from an owner class which is not the immediate parent, dynamic class creation, BLOB aggregation and much more. The following is an example:

ABSTRACT Class - GRAPH Class

ABSTRACT Class - COLUMN Class

There is no relationship between the GRAPH Class and the COLUMN Class. However, if a method or property is not available in an instance of the COLUMN Class, LLO will not use the ABSTRACT parent class definition, which is how OOP systems work today. Instead, LLO is able to use the class Owner's definition which could, for example, be a GRAPH.

Register and unregister an application

You need to call dwLightLibAppRegister() at the start of your program. This allows the Light Lib DLLs to be properly initialized. If this registration is not executed, you will receive errors.

At the end of execution, you will need to unregister your application with the Light Lib DLLs by calling dwLightLibAppUnRegister().

Light Lib Objects Constants

Abstract
Application
Class
Error

Class Constants

LLO_CLASS_ABSTRACT	`Abstract Class (Hidden)`
LLO_CLASS_APPLICATION	`Application Class`
LLO_CLASS_CONTEXT	`Context Class (Hidden)`
LLO_CLASS_ERROR	`Error Class`

Abstract Constants

LLO_ABSTRACT_APPLICATION
LLO_ABSTRACT_CARGO
LLO_ABSTRACT_CARGO_COUNT
LLO_ABSTRACT_CLASS_ID
LLO_ABSTRACT_CLASS_NAME
LLO_ABSTRACT_CLASS_VERSION
LLO_ABSTRACT_ERROR
LLO_ABSTRACT_LIBRARY_ID
LLO_ABSTRACT_LIBRARY_NAME
LLO_ABSTRACT_LIBRARY_VERSION
LLO_ABSTRACT_OWNER

Application Constants

LLO_APPLICATION_CARGO_COUNT_DEFAULT
LLO_APPLICATION_CONTEXT
LLO_APPLICATION_HANDLE
LLO_APPLICATION_NAME

Error Constants

Error Class

LLO_ERROR_ACTION	
LLO_ERROR_OBJECT	
LLO_ERROR_MESSAGE	`Error message`
LLO_ERROR_NUMBER	
LLO_ERROR_PARAM	`Extended depending on Error Type`
LLO_ERROR_PROPERTY	`Property define#`
LLO_ERROR_PROPERTY_NAME	`Property name`

LLO_ERROR_NUMBER

LLO_ERROR_CARGO_OUT_OF_LIMIT
LLO_ERROR_INVALID_CLASS_DEFINE
LLO_ERROR_INVALID_OWNER_TYPE
LLO_ERROR_INVALID_PARAMETERS
LLO_ERROR_INVALID_ACCESS_NEW
LLO_ERROR_INVALID_ACCESS_DEL
LLO_ERROR_INVALID_ACCESS_ACCESS
LLO_ERROR_INVALID_ACCESS_ASSIGN
LLO_ERROR_MEMORY_ALLOCATION
LLO_ERROR_NO_ERROR
LLO_ERROR_OBJECT_ACCESS_DENIED
LLO_ERROR_OBJECT_ASSIGN_DENIED
LLO_ERROR_READONLY_PROPERTY
LLO_ERROR_UNDEFINED_PROPERTY

LLO_ERROR_ACTION

LLO_ACTION_ACCESS
LLO_ACTION_ASSIGN
LLO_ACTION_DEL
LLO_ACTION_NEW

User Defined Constants

LLI_UDF_ABORT	User Defined Function Abort return value
LLI_UDF_CONT	User Defined Function Continue return value
LLI_UDF_ERROR	Error append during a Light Lib function execution
LLI_UDF_EXIT	Exit phase for a Light Lib function execution
LLI_UDF_IDLE	Idle phase for a Light Lib function execution
LLI_UDF_INIT	Init phase for a Light Lib function execution

Overview

Light Lib Objects

oAccess() DLL Functions

Purpose

Access an object's instance variable. See also Light Lib Objects

Syntax

```
oAccess(    dwLLObject      AS DWORD,
            dwProperty      AS DWORD,
            dwExtraParam    AS DWORD ) ---> dwData
```

Arguments

dwLLObject	A Light Lib object.
dwProperty	A property belonging to this Light Lib object.
dwExtraParam	Used to access the LLI_IMAGE_CARGO value. For example, if LLI_IMAGE_CARGO is a structure, *dwExtraParam* would represent the byte offset into the structure.

Returns

dwData	The value of the requested object member

Description

dwExtraParam must be cast to DWORD. This allows the Light Lib DLL to pass a POINTER, SHORTINT, LONGINT etc.

Examples

```
// This returns the name of the class to which the object belongs.
oAccess( dwMyObject, LLO_ABSTRACT_CLASS_NAME, 0 )

// This returns the value of the second cargo
// instance variable for this object.
 oAccess( dwMyObject, LLO_ABSTRACT_CARGO, 2 )
```

oAssign() DLL Functions

Purpose

Assign any value to a defined variable of an object.See also Light Lib Objects

Syntax

```
oAssign(    dwLLObject      AS DWORD,
            dwProperty      AS DWORD,
            dwValue         AS DWORD,
            dwExtraParam    AS DWORD ) ---> liError
```

Arguments

dwLLObject	A Light Lib object
dwProperty	The predefined value to change. You can only change or assign to the symbols noted as Assignable. You are not able to modify symbols that are Read Only symbols.
dwValue	The value to be assigned.
dwExtraParam	Used to access the LLI_IMAGE_CARGO value. For example, if LLI_IMAGE_CARGO is a structure, *dwExtraParam* would represent a byte offset into the structure.

Returns

liError	An error code.

Description

The *dwExtraParam* and *dwValue* must be cast to DWORD.This allows the DLL to pass a POINTER, SHORINT, LONGINT etc.

Examples

```
// This sets the cargo size for this object to 4 DWORD.
oAssign(dwMyObject, LLO_ABSTRACT_CARGO_SIZE, 4 )

//This sets the second cargo instance variable to dwMyValue.
oAssign(dwMyObject, LLO_ABSTRACT_CARGO, dwMyValue, 2 )
```

oNew() DLL Functions

Purpose

Used to create a new Application Object. See also Light Lib Objects

Syntax

```
oNew(   dwLLClass      AS DWORD,
        dwLLObject     AS DWORD,
        siSizeOfCargo  AS SHORTINT,
        dwValue        AS DWORD
        dwExtraParam   AS DWORD ) ---> ptrAppHnd
```

Arguments

dwLLClass	Represents the class of the object to be created.
dwLLObject	Represents the object to be created. If the class to which the object belongs is an application, the dwLLObject doesn't need to be defined (pass zero).
siSizeOfCargo	The size or number of DWORD parameters in an object's cargo.
dwValue	This is an optional value containing extra information. For example, when you create a new Column object inside a Graph object, *dwValue* dictates where the column should be inserted. If *dwValue* is 0, the new column becomes the last column. If *dwValue* is an existing Column number, the new Column is inserted before the passed number.
dwExtraParam	An optional parameter.

Returns

dwAppHnd	A pointer to a Light Lib Objects application handle.

Description

This allows you to register a Light Lib application with the Light Lib Objects DLL. This registration allows the Light Lib DLL to be used simultaneously by several applications in a multitasking operating system and to automate memory garbage collection. You must ensure that your applications always terminate with oDel().

When you create an application that uses aLight Lib DLL, you need to register that application with the Light Lib Objects DLL. This needs to be done at the very start of your application by calling oNew() and by passing the proper arguments.

Once registered, Light Lib Objects, automatically keeps track of all objects created within the registered application. This guarantees that all objects are properly connected to the Light Lib DLL.

When terminating an application, you need to unregister it from the Light Lib Images DLL with oDel(). This frees all memory allocated to objects in the application, even if the objects have not been explicitly erased. It is, however, always better to erase images from memory when they are no longer needed using oDel().

This oNew() and oDel() technique needs to be implemented to ensure that Light Lib Objects can properly manage all memory and processes when being called simultaneously from multiple applications. This is

very important in a multitasking operating system.

oDel() DLL Functions

Purpose

Delete any Light Lib object. This frees all memory allocated to objects in a registered Light Lib application. See also Light Lib Objects

Syntax

oDel(*dwLLObject* AS DWORD) ---> dwAppHnd

Arguments

dwLLObject A DWORD representing any Light Lib object.

Returns

dwAppHnd An empty pointer to a Light Lib Images application handle.

Description

You must ensure that your Light Lib applications always terminate with oDel().

Be aware, that deleting a Light Lib object will also delete all of its child objects (if any) as well. As an example, deleting the Application object in turn deletes all objects created by that application from memory. It is highly recommended to delete the registered Application object by calling oDel() prior to exiting any Light Lib application.

This oNew() and oDel() technique (registering and unregistering) must be implemented to ensure that Light Lib Objects can properly manage the Light Lib DLLs when being called simultaneously from multiple applications.This is very important in a multitasking operating system.

dwLightLibApp() CA-Visual Objects

Purpose

Get the current Light Lib Application. See also Light Lib Objects

Syntax

dwLightLibApp() ---> *dwLightLibRegisteredApp*

Arguments

None.

Returns

dwLightLibRegisteredApp	The registered application.

dwLightLibAppRegister() CA-Visual Objects

Purpose

Register this instance of application into the LLO.DLL This must be done only once in an application's execution, and prior to any calls to the Light Lib library you are using. See also Light Lib Objects

Syntax

dwLightLibAppRegister(*oApp* AS OBJECT,
oWindow AS OBJECT) ---> *dwLightLibRegisteredApp*

Arguments

oApp	Application to register.
oWindow	Owner window.

Returns

dwLightLibRegisteredApp	The value of the registered application.

Description

This function is used to register your Light Lib application with Light Lib Objects. If this registration process is not done, your application will not work properly.

dwLightLibAppUnRegister() CA-Visual Objects

Purpose

Unregister a Light Lib application from the Light Lib Objects DLL. See also Light Lib Objects

Syntax

dwLightLibAppUnRegister() ---> *dwLightLibRegisteredApp*

Arguments

None

Returns

dwLightLibRegisteredApp Unregister an application.

Out of Memory

If you are experiencing memory problems in applications using Light Lib DLLs, there is a good chance that you are keeping unnecessary references to objects such as images or graphs in memory. When your application no longer needs an object, you should formally remove or delete it from memory by calling oDel() with the proper parameters.

DLL Crashes

This error could occur when multiple applications that use Light Lib DLLs are running simultaneously. In order to prevent conflicts between them, you must ensure that each application is registered with Light Lib Objects.

This involves making a call to oNew() with the proper parameters at the beginning of your program. Also, remember to make a call to oDel() just before your application terminates.

Unable to Load a DLL at Runtime

Make sure that the proper Light Lib DLL is available in your WINDOWS\SYSTEM directory. At installation time, Light Lib DLLs are installed to this directory. If they are not present when your application runs, the applications will cause a LoadError().