DEBENU QUICKPDF LIBRARY

Delphi Edition
ActiveX Edition
DLL Edition

Version 11.15.1.0

Reference Guide

AddArcToPath

Vector graphics, Path definition and drawing, Form fields, Annotations and hotspot links

Description

Adds an arc to the current path.

The arc is drawn around a center point for a specified number of degrees either clockwise or anti-clockwise.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddArcToPath(CenterX, CenterY,
  TotalAngle: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddArcToPath(CenterX As Double,
  CenterY As Double, TotalAngle As Double) As Long
```

DLL

```
int DPLAddArcToPath(int InstanceID, double CenterX, double CenterY,
  double TotalAngle);
```

Parameters

CenterX	The horizontal co-ordinate of the center of the arc
CenterY	The vertical co-ordinate of the center of the arc
TotalAngle	The angular length of the arc. If this value is positive the arc will be drawn in a clockwise direction. A negative value will result in an arc drawn in an anti-clockwise direction. This value must be greater or less than 0. A value of 360 will result in a full circle being drawn.

AddBoxToPath

Vector graphics, Path definition and drawing

Version history

This function was introduced in Quick PDF Library version 9.14.

Description

Adds a rectangle to the current path.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddBoxToPath(Left, Top, Width,
  Height: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddBoxToPath(Left As Double,
  Top As Double, Width As Double, Height As Double) As Long
```

DLL

```
int DPLAddBoxToPath(int InstanceID, double Left, double Top, double Width,
  double Height);
```

Parameters

Left	The horizontal co-ordinate of the left edge of the box
Top	The vertical co-ordinate of the top edge of the box
Width	The width of the box
Height	The height of the box

AddCJKFont

Version history

This function was introduced in Quick PDF Library version 7.11.

Description

Adds a CJK (Chinese Japanese Korean) font to the PDF document.

At present, the only supported CJK fonts are the Japanese "HeiseiKakuGo-W5" font and the Korean "HYGoThic-Medium" font.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddCJKFont(CJKFontID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddCJKFont(
  CJKFontID As Long) As Long
```

DLL

```
int DPLAddCJKFont(int InstanceID, int CJKFontID);
```

Parameters

CJKFontID	1 = HeiseiKakuGo-W5
	2 = HeiseiKakuGo-W5 (Bold)
	3 = HeiseiKakuGo-W5 (Bold Italic)
	4 = HeiseiKakuGo-W5 (Italic)
	5 = HYGoThic-Medium
	6 = HYGoThic-Medium (Bold)
	7 = HYGoThic-Medium (Bold Italic)
	8 = HYGoThic-Medium (Italic)

AddCurveToPath

Vector graphics, Path definition and drawing

Description

Adds a bezier curve to the current path.

The curve is drawn from the last point to the point defined by (EndX, EndY).

(CtAX, CtAY) and (CtBX, CtBY) define the two bezier control points.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddCurveToPath(CtAX, CtAY, CtBX, CtBY,
  EndX, EndY: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddCurveToPath(CtAX As Double,
  CtAY As Double, CtBX As Double, CtBY As Double,
  EndX As Double, EndY As Double) As Long
```

DLL

```
int DPLAddCurveToPath(int InstanceID, double CtAX, double CtAY,
  double CtBX, double CtBY, double EndX, double EndY);
```

Parameters

CtAX	The horizontal co-ordinate of the first control point
CtAY	The vertical co-ordinate of the first control point
CtBX	The horizontal co-ordinate of the second control point
CtBY	The vertical co-ordinate of the second control point
EndX	The horizontal co-ordinate of the end point of the bezier curve
EndY	The vertical co-ordinate of the end point of the bezier curve

AddEmbeddedFile

Version history

This function was introduced in Quick PDF Library version 7.11.

Description

Embeds a file into the PDF but does not link it to any part of the document.

The **AddFileAttachment** function can be used to make the embedded file available as an attachment in the PDF viewer. The PDF viewer must support this functionality (Adobe Reader 7 and later). This process can be done in one step using the **EmbedFile** function.

The **AddLinkToEmbeddedFile** function can be used to create a hotspot on a page that links to the embedded file.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddEmbeddedFile(FileName,
  MIMEType: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddEmbeddedFile(
  FileName As String, MIMEType As String) As Long
```

DLL

```
int DPLAddEmbeddedFile(int InstanceID, wchar_t * FileName,
  wchar_t * MIMEType);
```

Parameters

FileName	The path and filename of the file to embed into the PDF.
MIMEType	The MIME type of the embedded file. For example "image/jpeg" for a JPEG image.

Return values

0	The file could not be found or there was an error embedding the file into the PDF
Non-zero	An EmbeddedFileID that can be used with the **AddFileAttachment** function

AddFileAttachment

Version history

This function was introduced in Quick PDF Library version 7.11.

Description

Makes an embedded file available as an attachment in the PDF viewer, if it supports this functionality. Adobe Reader 7 and later allow the user to work with file attachments.

First use the **AddEmbeddedFile** function to embed the file into the PDF.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddFileAttachment(Title: WideString;
  EmbeddedFileID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddFileAttachment(
  Title As String, EmbeddedFileID As Long) As Long
```

DLL

```
int DPLAddFileAttachment(int InstanceID, wchar_t * Title,
  int EmbeddedFileID);
```

Parameters

Title	The title of the attachment that should appear in the PDF viewer
EmbeddedFileID	The value returned from the **AddEmbeddedFile** function

Return values

0	The EmbeddedFileID parameter was invalid, or the Title was blank
1	The embedded file was made available as an attachment successfully

AddFormFieldChoiceSub

Version history

This function was introduced in Quick PDF Library version 9.11.

Description

Similar to the **AddFormFieldSub** function but allows a choice field item's export value and display value to be set.

The function returns a temporary form field Index which can be used with the **SetFormFieldBounds**, **SetFormFieldCheckStyle** and other functions.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddFormFieldChoiceSub(Index: Integer;
  SubName, DisplayName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddFormFieldChoiceSub(
  Index As Long, SubName As String,
  DisplayName As String) As Long
```

DLL

```
int DPLAddFormFieldChoiceSub(int InstanceID, int Index, wchar_t * SubName,
  wchar_t * DisplayName);
```

Parameters

Index	The index of the form field to work with. The first form field has an index of 1. The form field must be a choice field.
SubName	The export value of the new sub-field. The value of the form field could be set to this name using the **SetFormFieldValue** function.
DisplayName	The display name of the new sub-field.

Return values

0	The sub-field was not added. The specified form field may not have been a choice form field.
Non-zero	A temporary field Index

AddFormFieldSub

Description

Adds a sub-field to the specified radio-button or choice form field.

The function returns a temporary form field Index which can be used with the **SetFormFieldBounds**, **SetFormFieldCheckStyle** and other functions.

To set a choice item's export value and display value use the **AddFormFieldChoiceSub** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddFormFieldSub(Index: Integer;
  SubName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddFormFieldSub(Index As Long,
  SubName As String) As Long
```

DLL

```
int DPLAddFormFieldSub(int InstanceID, int Index, wchar_t * SubName);
```

Parameters

Index	The index of the form field to work with. The first form field has an index of 1.
SubName	The name of the new sub-field. The value of the form field could be set to this name using the **SetFormFieldValue** function.

Return values

0	The sub-field was not added. The specified form field may not have been a radio-button or choice form field.
Non-zero	A temporary field Index

AddFormFont

Description

Adds a font to the form.

The font must have been added using one of the Add*Font functions.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddFormFont(FontID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddFormFont(
  FontID As Long) As Long
```

DLL

```
int DPLAddFormFont(int InstanceID, int FontID);
```

Parameters

FontID	The FontID returned by one of the Add*Font functions

Return values

0	Invalid FontID
Non-zero	The font was added successfully, the value returned is the number of fonts available for use by form fields

AddFreeTextAnnotation

Text, Annotations and hotspot links

Version history

This function was introduced in Quick PDF Library version 10.11.

Description

Adds a free text annotation to the selected page. If a border and/or fill is specified using the Options parameter then the settings are retrieved from the current line color, fill color, line width, pen dash settings.

SetLineColor does not affect border color. Border color is currently set to the same color as the text color due to the way Acrobat works. SelectFont does not affect font style. Currently the font is hardcoded to the standard Helvetica font due to the way Acrobat works.

SetTextSize will affect text size correctly
SetTextColor will affect text color correctly
SetLineWidth will adjust border width correctly
SetTextAlign will adjust text alignment correctly

The **SetTransparency** function will *not* change the transparency of annotations. Use the new **AddFreeTextAnnotationEx** function if you want to adjust transparency settings.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddFreeTextAnnotation(Left, Top, Width,
  Height: Double; Text: WideString; Angle, Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddFreeTextAnnotation(
  Left As Double, Top As Double, Width As Double,
  Height As Double, Text As String, Angle As Long,
  Options As Long) As Long
```

DLL

```
int DPLAddFreeTextAnnotation(int InstanceID, double Left, double Top,
  double Width, double Height, wchar_t * Text, int Angle,
  int Options);
```

Parameters

Left	The horizontal coordinate of the left edge of the annotation rectangle
Top	The vertical coordinate of the left edge of the annotation rectangle
Width	The width of the annotation rectangle
Height	The height of the annotation rectangle
Text	The text content of the annotation
Angle	The angle of the drawn text. Can be 0, 90, 180 or 270.
Options	0 = Outline 1 = Fill 2 = Fill and Outline

AddFreeTextAnnotationEx

Version history

This function was introduced in Quick PDF Library version 11.14.

Description

Adds a free text annotation to the selected page. If a border and/or fill is specified using the Options parameter then the settings are retrieved from the current line color, fill color, line width, pen dash settings. Use the Transparency parameter to adjust the transparency settings for free text annotations. The SetTransparency function does not work with annotations.

SetLineColor does not affect border color. Border color is currently set to the same color as the text color due to the way Acrobat works. SelectFont does not affect font style. Currently the font is hardcoded to standard Helvetica font due to the way Acrobat works.

SetTextSize will affect text size correctly
SetTextColor will affect text color correctly
SetLineWidth will adjust border width correctly
SetTextAlign will adjust text alignment correctly

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddFreeTextAnnotationEx(Left, Top, Width,
  Height: Double; Text: WideString; Angle, Options,
  Transparency: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddFreeTextAnnotationEx(
  Left As Double, Top As Double, Width As Double,
  Height As Double, Text As String, Angle As Long,
  Options As Long, Transparency As Long) As Long
```

DLL

```
int DPLAddFreeTextAnnotationEx(int InstanceID, double Left, double Top,
  double Width, double Height, wchar_t * Text, int Angle,
  int Options, int Transparency);
```

Parameters

Left	The horizontal coordinate of the left edge of the annotation rectangle
Top	The vertical coordinate of the left edge of the annotation rectangle
Width	The width of the annotation rectangle
Height	The height of the annotation rectangle
Text	The text content of the annotation
Angle	The angle of the drawn text. Can be 0, 90, 180 or 270.
Options	0 = Outline 1 = Fill 2 = Fill and Outline
Transparency	The amount of transparency to apply 0 = No transparency 50 = 50% transparency 100 = Invisible

AddGlobalJavaScript

Description

Adds JavaScript to a global location in the document.

For example, this allows functions to be defined which can then be called from JavaScript attached to events.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddGlobalJavaScript(PackageName,
  JavaScript: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddGlobalJavaScript(
  PackageName As String, JavaScript As String) As Long
```

DLL

```
int DPLAddGlobalJavaScript(int InstanceID, wchar_t * PackageName,
  wchar_t * JavaScript);
```

Parameters

PackageName	The name to store the JavaScript under. If any JavaScript is already stored under this name it will be removed and the new JavaScript will be stored in its place.
JavaScript	The JavaScript to store globally under the specified package name.

Return values

0	The PackageName was empty
1	The JavaScript was stored successfully

AddImageFromFile

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Adds an image from a file to the selected document.

Once an image has been added to the document it can be drawn on any page multiple times without further increasing the size of the PDF file.

Supported image file types are: BMP, TIFF, JPEG, PNG, GIF, WMF and EMF.

For BMP and TIFF images, the **CompressImages** function can called before calling this function to compress the image data. Other image types are automatically compressed.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddImageFromFile(FileName: WideString;
  Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddImageFromFile(
  FileName As String, Options As Long) As Long
```

DLL

```
int DPLAddImageFromFile(int InstanceID, wchar_t * FileName, int Options);
```

Parameters

FileName	The file name of the image to add.
Options	For multi-page TIFF images this parameter specifies the page number to load.
	For PNG images:
	0 = Load the image as usual
	1 = Load the alpha channel as a greyscale image
	2 = Load the image and alpha channel (limit alpha to 8-bit)
	3 = Load the image (limit image 8-bit/channel)
	4 = Load the alpha channel (limit to 8-bit/channel)
	5 = Load the image with alpha channel (limit both to 8-bit/channel)
	6 = Load the image and alpha channel
	7 = Load the image and ICC color profile
	For other image types this parameter should be set to 0.
	Setting Options to -1 forces TIFF, EMF and WMF images to be loaded using the GDI+ graphics library. Multipage TIFF images can also be loaded using GDI+ by setting the Options parameter to -PageNumber (for example -3 for page 3).

Return values

0	The image could not be added. Either it could not be found or it is in an unsupported format.
Non-zero	The image was added successfully. The ImageID is returned which can be passed to functions like **SelectImage** and **DrawImage**.

AddImageFromFileOffset

Description

Adds an image from a part of a file to the selected document.

For example, if many images have been concatenated into one file this function will allow the individual images to be extracted and added to the document.

Once an image has been added to the document it can be drawn on any page multiple times without further increasing the size of the PDF file.

Supported image file types are: BMP, TIFF, JPEG, PNG, GIF, WMF and EMF.

For BMP and TIFF images, the **CompressImages** function can called before calling this function to compress the image data. Other image types are automatically compressed.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddImageFromFileOffset(
   FileName: WideString; Offset, DataLength, Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddImageFromFileOffset(
   FileName As String, Offset As Long, DataLength As Long,
   Options As Long) As Long
```

DLL

```
int DPLAddImageFromFileOffset(int InstanceID, wchar_t * FileName,
   int Offset, int DataLength, int Options);
```

Parameters

FileName	The name of the file containing the images.
Offset	The offset into the file where the required image starts. The first byte in the file has an offset of 0.
DataLength	The length of the image data in bytes
Options	For multi-page TIFF images this parameter specifies the page number to load.
	For PNG images:
	0 = Load the image as usual
	1 = Load the alpha channel as a greyscale image
	2 = Load the image and alpha channel (limit alpha to 8-bit)
	3 = Load the image (limit image 8-bit/channel)
	4 = Load the alpha channel (limit to 8-bit/channel)
	5 = Load the image with alpha channel (limit both to 8-bit/channel)
	6 = Load the image and alpha channel
	7 = Load the image and ICC color profile
	For other image types this parameter should be set to 0.
	Setting Options to -1 forces TIFF, EMF and WMF images to be loaded using the GDI+ graphics library. Multipage TIFF images can also be loaded using GDI+ by setting the Options parameter to -PageNumber (for example -3 for page 3).

Return values

0	The image could not be read from the file. This could indicate invalid image data or the file could not be found.
Non-zero	The image was read from the file and successfully added to the document. The value returned is the ID of the image which can be used with the image drawing functions such as **DrawImage**.

AddImageFromStream

Description

Adds an image from a TStream to the selected document.

Once an image has been added to the document it can be drawn on any page multiple times without further increasing the size of the PDF file.

Supported image file types are: BMP, TIFF, JPEG, PNG, GIF, WMF and EMF.

For BMP and TIFF images, the **CompressImages** function can called before calling this function to compress the image data. Other image types are automatically compressed.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddImageFromStream(InStream: TStream;
  Options: Integer): Integer;
```

Parameters

InStream	The TStream object containing the image data. The current position in the stream will be ignored, image data will be read from position 0 in the stream.
Options	For multi-page TIFF images this parameter specifies the page number to load.
	For PNG images:
	0 = Load the image as usual
	1 = Load the alpha channel as a greyscale image
	2 = Load the image and alpha channel (limit alpha to 8-bit)
	3 = Load the image (limit image 8-bit/channel)
	4 = Load the alpha channel (limit to 8-bit/channel)
	5 = Load the image with alpha channel (limit both to 8-bit/channel)
	6 = Load the image and alpha channel
	7 = Load the image and ICC color profile
	For other image types this parameter should be set to 0.
	Setting Options to -1 forces TIFF, EMF and WMF images to be loaded using the GDI+ graphics library. Multipage TIFF images can also be loaded using GDI+ by setting the Options parameter to -PageNumber (for example -3 for page 3).

Return values

0	There was an error reading valid image data from the stream
Non-zero	The image was successfully added to the document. The value returned is the ID of the image which can be used with the image drawing functions such as **DrawImage**.

AddImageFromString

Description

Adds an image from memory to the selected document.

Once an image has been added to the document it can be drawn on any page multiple times without further increasing the size of the PDF file.

Supported image file types are: BMP, TIFF, JPEG, PNG, GIF, WMF and EMF.

For BMP and TIFF images, the **CompressImages** function can called before calling this function to compress the image data. Other image types are automatically compressed.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddImageFromString(
  const Source: AnsiString; Options: Integer): Integer;
```

DLL

```
int DPLAddImageFromString(int InstanceID, char * Source, int Options);
```

Parameters

Source	A string containing the image data. In the ActiveX version of the library this string must contain 16-bit characters, only the lower 8-bits of each character will be used.
Options	For multi-page TIFF images this parameter specifies the page number to load. For PNG images: 0 = Load the image as usual 1 = Load the alpha channel as a greyscale image 2 = Load the image and alpha channel (limit alpha to 8-bit) 3 = Load the image (limit image 8-bit/channel) 4 = Load the alpha channel (limit to 8-bit/channel) 5 = Load the image with alpha channel (limit both to 8-bit/channel) 6 = Load the image and alpha channel 7 = Load the image and ICC color profile For other image types this parameter should be set to 0. Setting Options to -1 forces TIFF, EMF and WMF images to be loaded using the GDI+ graphics library. Multipage TIFF images can also be loaded using GDI+ by setting the Options parameter to -PageNumber (for example -3 for page 3).

Return values

0	The image data was invalid or the image was in an unsupported format
1	The image was added successfully. The value returned is the ImageID which can be used with functions like **SelectImage** and **DrawImage**.

AddImageFromVariant

Description

Adds an image from a variant byte array to the selected document.

Once an image has been added to the document it can be drawn on any page multiple times without further increasing the size of the PDF file.

Supported image file types are: BMP, TIFF, JPEG, PNG, GIF, WMF and EMF.

For BMP and TIFF images, the **CompressImages** function can called before calling this function to compress the image data. Other image types are automatically compressed.

Syntax

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddImageFromVariant(
    SourceData As Variant, Options As Long) As Long
```

Parameters

SourceData	A variant containing the image data
Options	For multi-page TIFF images this parameter specifies the page number to load.
	For PNG images:
	0 = Load the image as usual
	1 = Load the alpha channel as a greyscale image
	2 = Load the image and alpha channel (limit alpha to 8-bit)
	3 = Load the image (limit image 8-bit/channel)
	4 = Load the alpha channel (limit to 8-bit/channel)
	5 = Load the image with alpha channel (limit both to 8-bit/channel)
	6 = Load the image and alpha channel
	7 = Load the image and ICC color profile
	For other image types this parameter should be set to 0.
	Setting Options to -1 forces TIFF, EMF and WMF images to be loaded using the GDI+ graphics library. Multipage TIFF images can also be loaded using GDI+ by setting the Options parameter to -PageNumber (for example -3 for page 3).

Return values

0	The image could not be added
Non-zero	The image was added successfully. This is the ID of the new image.

AddLGIDictToPage

Version history

This function was introduced in Quick PDF Library version 7.15.

Description

Adds a new LGIDict object to the selected page.
This is used with the GeoPDF system as defined in Open Geospatial Consortium Inc.'s OGC 08-139r2 specification.
More than one dictionary can be added to the page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddLGIDictToPage(
  DictContent: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddLGIDictToPage(
  DictContent As String) As Long
```

DLL

```
int DPLAddLGIDictToPage(int InstanceID, wchar_t * DictContent);
```

Parameters

DictContent	The LGIDict dictionary content to add to the page.

Return values

0	The LGI dictionary could not be added to the page. Check that the dictionary content string is a valid PDF dictionary.
1	The LGI dictionary was added successfully.

AddLineToPath

Vector graphics, Path definition and drawing

Description

Adds a line to the current path.

The line is drawn from the last point to the point defined by (EndX, EndY).

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddLineToPath(EndX, EndY: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddLineToPath(EndX As Double,
  EndY As Double) As Long
```

DLL

```
int DPLAddLineToPath(int InstanceID, double EndX, double EndY);
```

Parameters

EndX	The horizontal co-ordinate of the end point of the line to add to the path
EndY	The vertical co-ordinate of the end point of the line to add to the path

AddLinkToDestination

Version history

> This function was introduced in Quick PDF Library version 7.22.

Description

> Adds a clickable hotspot rectangle to the selected page which links to another page in the same document. The target page, position and zoom level are specified by a destination object which can be created with the **NewDestination** function.
>
> Use the **SetAnnotBorderColor** function to change the color of the hotspot border.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddLinkToDestination(Left, Top, Width,
  Height: Double; DestID, Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddLinkToDestination(
  Left As Double, Top As Double, Width As Double,
  Height As Double, DestID As Long, Options As Long) As Long
```

DLL

```
int DPLAddLinkToDestination(int InstanceID, double Left, double Top,
  double Width, double Height, int DestID, int Options);
```

Parameters

Left	The left edge of the hotspot rectangle
Top	The top edge of the hotspot rectangle
Width	The width of the hotspot rectangle
Height	The height of the hotspot rectangle
DestID	The DestID of a destination object
Options	Specifies the appearance of the link: 0 = No border 1 = Draw a border

Return values

0	The DestID property was invalid
1	The link annotation was created successfully

AddLinkToEmbeddedFile

Document properties, Annotations and hotspot links

Version history

This function was introduced in Quick PDF Library version 7.11.

Description

Adds a clickable hotspot rectangle to the selected page which links to an embedded file.

Files can be embedded into the PDF using the **AddEmbeddedFile** function.

The function definition was changed in version 9.11 to provide separate parameters for the title/contents and transparency.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddLinkToEmbeddedFile(Left, Top, Width,
  Height: Double; EmbeddedFileID: Integer; Title, Contents: WideString;
  IconType, Transpareny: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddLinkToEmbeddedFile(
  Left As Double, Top As Double, Width As Double,
  Height As Double, EmbeddedFileID As Long, Title As String,
  Contents As String, IconType As Long,
  Transpareny As Long) As Long
```

DLL

```
int DPLAddLinkToEmbeddedFile(int InstanceID, double Left, double Top,
  double Width, double Height, int EmbeddedFileID,
  wchar_t * Title, wchar_t * Contents, int IconType,
  int Transpareny);
```

Parameters

Left	The horizontal co-ordinate of the left edge of the hotspot rectangle
Top	The vertical co-ordinate of the top of the hotspot rectangle
Width	The width of the hotspot rectangle
Height	The height of the hotspot rectangle
EmbeddedFileID	The value returned from the **AddEmbeddedFile** function
Title	The title of the attachment that should appear in the PDF viewer.
Contents	The text to use for the contents of the popup
IconType	0 = Standard icon (PushPin) 1 = 28x28 disk image 2 = No icon 3 = Graph 4 = Paperclip 5 = Tag 6 = Solid white rectangle
Transpareny	The transparency percentage to apply ranging from 0 to 100. A value of 0 indicates 0% transparency which is fully opaque (no transparency). A value of 100 indicates 100% transparency which would make the icon invisible.

Return values

0	The EmbeddedFileID parameter was invalid
1	The link was created successfully

AddLinkToFile

Annotations and hotspot links

Description

Adds a clickable hotspot rectangle to the selected page which links to a specific page and position in another PDF document.

Use the **SetAnnotBorderColor** function to change the color of the hotspot border.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddLinkToFile(Left, Top, Width,
  Height: Double; FileName: WideString; Page: Integer; Position: Double;
  NewWindow, Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddLinkToFile(Left As Double,
  Top As Double, Width As Double, Height As Double,
  FileName As String, Page As Long, Position As Double,
  NewWindow As Long, Options As Long) As Long
```

DLL

```
int DPLAddLinkToFile(int InstanceID, double Left, double Top,
  double Width, double Height, wchar_t * FileName, int Page,
  double Position, int NewWindow, int Options);
```

Parameters

Left	The horizontal co-ordinate of the left edge of the hotspot rectangle
Top	The vertical co-ordinate of the top edge of the hotspot rectangle
Width	The width of the hotspot rectangle
Height	The height of the hotspot rectangle
FileName	The path and file name of the PDF document to link to.
Page	The page in the destination document to link to
Position	The vertical co-ordinate on the destination page to link to
NewWindow	0 = Close the current document and then open the new document 1 = Open the current document in a new window
Options	Specifies the appearance of the link: 0 = No border 1 = Draw a border

AddLinkToFileDest

Annotations and hotspot links

Version history

This function was introduced in Quick PDF Library version 10.13.

Description

Adds a clickable hotspot rectangle to the selected named destination which links to a specific page and position in another PDF document.

Use the **SetAnnotBorderColor** function to change the color of the hotspot border.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddLinkToFileDest(Left, Top, Width,
  Height: Double; FileName, NamedDest: WideString; Position: Double;
  NewWindow, Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddLinkToFileDest(
  Left As Double, Top As Double, Width As Double,
  Height As Double, FileName As String, NamedDest As String,
  Position As Double, NewWindow As Long, Options As Long) As Long
```

DLL

```
int DPLAddLinkToFileDest(int InstanceID, double Left, double Top,
  double Width, double Height, wchar_t * FileName,
  wchar_t * NamedDest, double Position, int NewWindow,
  int Options);
```

Parameters

Left	The horizontal co-ordinate of the left edge of the hotspot rectangle
Top	The vertical co-ordinate of the top edge of the hotspot rectangle
Width	The width of the hotspot rectangle
Height	The height of the hotspot rectangle
FileName	The path and file name of the PDF document to link to.
NamedDest	The Named Destination string in the destination document to link to
Position	The vertical co-ordinate on the destination page to link to
NewWindow	0 = Close the current document and then open the new document 1 = Open the current document in a new window
Options	Specifies the appearance of the link: 0 = No border 1 = Draw a border

AddLinkToFileEx

Annotations and hotspot links

Version history

This function was introduced in Quick PDF Library version 9.16.

Description

Adds a clickable hotspot rectangle to the selected page which links to a specific page and position in another PDF document.

Use the **SetAnnotBorderColor** function to change the color of the hotspot border.

The link to the target document is only via the file name. This means the page dimensions of the target document are not known so the DestLeft, DestTop, DestRight and DestBottom parameters are always specified in points measured from the bottom left corner of the destination page's MediaBox.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddLinkToFileEx(Left, Top, Width,
  Height: Double; FileName: WideString; DestPage, NewWindow, Options,
  Zoom, DestType: Integer; DestLeft, DestTop, DestRight,
  DestBottom: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddLinkToFileEx(Left As Double,
  Top As Double, Width As Double, Height As Double,
  FileName As String, DestPage As Long, NewWindow As Long,
  Options As Long, Zoom As Long, DestType As Long,
  DestLeft As Double, DestTop As Double, DestRight As Double,
  DestBottom As Double) As Long
```

DLL

```
int DPLAddLinkToFileEx(int InstanceID, double Left, double Top,
  double Width, double Height, wchar_t * FileName, int DestPage,
  int NewWindow, int Options, int Zoom, int DestType,
  double DestLeft, double DestTop, double DestRight,
  double DestBottom);
```

Parameters

Left	The horizontal co-ordinate of the left edge of the hotspot rectangle
Top	The vertical co-ordinate of the top edge of the hotspot rectangle
Width	The width of the hotspot rectangle
Height	The height of the hotspot rectangle
FileName	The path and file name of the PDF document to link to.
DestPage	The page in the destination document to link to
NewWindow	0 = Close the current document and then open the new document 1 = Open the current document in a new window
Options	Specifies the appearance of the link: 0 = No border 1 = Draw a border
Zoom	The zoom percentage to use for the destination object, valid values from 0 to 6400. Only used for DestType = 1, should be set to 0 for other DestTypes.
DestType	1 = "XYZ" - the target page is positioned at the point specified by the Left and Top parameters. The Zoom parameter specifies the zoom percentage. 2 = "Fit" - the entire page is zoomed to fit the window. None of the other parameters are used and should be set to zero. 3 = "FitH" - the page is zoomed so that the entire width of the page is visible. The height of the page may be greater or less than the height of the window. The page is positioned at the vertical position specified by the Top parameter. 4 = "FitV" - the page is zoomed so that the entire height of the page can be seen. The width of the page may be greater or less than the width of the window. The page is positioned at the horizontal position specified by the Left parameter. 5 = "FitR" - the page is zoomed so that a certain rectangle on the page is visible. The Left, Top, Right and Bottom parameters define the rectangular area on the page. 6 = "FitB" - the page is zoomed so that it's bounding box is visible. 7 = "FitBH" - the page is positioned vertically at the position specified by the Top parameter. The page is zoomed so that the entire width of the page's bounding box is visible. 8 = "FitBV" - the page is positioned at the horizontal position specified by the Left parameter. The page is zoomed just enough to fit the entire height of the bounding box into the window.
DestLeft	The horizontal position used by DestType = 1, 4, 5 and 8
DestTop	The vertical position used by DestType = 1, 3, 5 and 7
DestRight	The horizontal position of the righthand edge of the rectangle. Used by DestType = 5
DestBottom	The horizontal position of the bottom of the rectangle. Used by DestType = 5

AddLinkToJavaScript

Description

Adds a clickable hotspot rectangle to the selected page which links to a JavaScript action.

Use the **SetAnnotBorderColor** function to change the color of the hotspot border.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddLinkToJavaScript(Left, Top, Width,
  Height: Double; JavaScript: WideString; Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddLinkToJavaScript(
  Left As Double, Top As Double, Width As Double,
  Height As Double, JavaScript As String,
  Options As Long) As Long
```

DLL

```
int DPLAddLinkToJavaScript(int InstanceID, double Left, double Top,
  double Width, double Height, wchar_t * JavaScript,
  int Options);
```

Parameters

Left	The horizontal co-ordinate of the left edge of the hotspot rectangle
Top	The vertical co-ordinate of the top edge of the hotspot rectangle
Width	The width of the hotspot rectangle
Height	The height of the hotspot rectangle
JavaScript	The JavaScript to execute.
Options	Specifies the appearance of the link: 0 = No border 1 = Draw a border

AddLinkToLocalFile

Annotations and hotspot links

Version history

This function was introduced in Quick PDF Library version 7.19.

Description

Adds a clickable hotspot rectangle to the selected page which links to a local file.

The file doesn't have to exist when the PDF is created but should exist when the PDF is viewed for the link to work.

Use the **SetAnnotBorderColor** function to change the color of the hotspot border.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddLinkToLocalFile(Left, Top, Width,
  Height: Double; FileName: WideString; Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddLinkToLocalFile(
  Left As Double, Top As Double, Width As Double,
  Height As Double, FileName As String, Options As Long) As Long
```

DLL

```
int DPLAddLinkToLocalFile(int InstanceID, double Left, double Top,
  double Width, double Height, wchar_t * FileName, int Options);
```

Parameters

Left	The left edge of the hotspot rectangle
Top	The top edge of the hotspot rectangle
Width	The width of the hotspot rectangle
Height	The height of the hotspot rectangle
FileName	The relative or absolute path to the local file.
Options	Specifies the appearance of the link and whether the target is opened in a new window or the same window: 0 = No border, same window 1 = Draw a border, same window 2 = No border, new window 3 = Draw a border, new window

AddLinkToPage

Description

Adds a clickable hotspot rectangle to the selected page which links to another page in the same document.

Use the **SetAnnotBorderColor** function to change the color of the hotspot border.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddLinkToPage(Left, Top, Width,
  Height: Double; Page: Integer; Position: Double;
  Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddLinkToPage(Left As Double,
  Top As Double, Width As Double, Height As Double,
  Page As Long, Position As Double, Options As Long) As Long
```

DLL

```
int DPLAddLinkToPage(int InstanceID, double Left, double Top,
  double Width, double Height, int Page, double Position,
  int Options);
```

Parameters

Left	The left edge of the hotspot rectangle
Top	The top edge of the hotspot rectangle
Width	The width of the hotspot rectangle
Height	The height of the hotspot rectangle
Page	The destination page number to link to
Position	The vertical position on the destination page to link to
Options	Specifies the appearance of the link: 0 = No border 1 = Draw a border

AddLinkToWeb

Annotations and hotspot links

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Adds a clickable hotspot rectangle to the selected page which links to a URL on the internet.

This can also be used to link to an e-mail address.

Use the **SetAnnotBorderColor** function to change the color of the hotspot border.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddLinkToWeb(Left, Top, Width,
   Height: Double; Link: WideString; Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddLinkToWeb(Left As Double,
   Top As Double, Width As Double, Height As Double,
   Link As String, Options As Long) As Long
```

DLL

```
int DPLAddLinkToWeb(int InstanceID, double Left, double Top, double Width,
   double Height, wchar_t * Link, int Options);
```

Parameters

Left	The left edge of the hotspot rectangle
Top	The top edge of the hotspot rectangle
Width	The width of the hotspot rectangle
Height	The height of the hotspot rectangle
Link	The URL to link to. Some examples: "http://www.example.com/" "mailto:info@example.com"
Options	Specifies the appearance of the link: 0 = No border 1 = Draw a border

AddNoteAnnotation

Annotations and hotspot links

Description

Adds a note annotation to the selected page. The values of the color parameters range from 0 to 1, with 0 indicating 0% and 1 indicating 100% of the color.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddNoteAnnotation(Left, Top: Double;
  AnnotType: Integer; PopupLeft, PopupTop, PopupWidth,
  PopupHeight: Double; Title, Contents: WideString; Red, Green,
  Blue: Double; Open: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddNoteAnnotation(
  Left As Double, Top As Double, AnnotType As Long,
  PopupLeft As Double, PopupTop As Double, PopupWidth As Double,
  PopupHeight As Double, Title As String, Contents As String,
  Red As Double, Green As Double, Blue As Double,
  Open As Long) As Long
```

DLL

```
int DPLAddNoteAnnotation(int InstanceID, double Left, double Top,
  int AnnotType, double PopupLeft, double PopupTop,
  double PopupWidth, double PopupHeight, wchar_t * Title,
  wchar_t * Contents, double Red, double Green, double Blue,
  int Open);
```

Parameters

Left	The horizontal co-ordinate of the anchor for the annotation
Top	The vertical co-ordinate of the anchor for the annotation
AnnotType	The annotation type: 0 = Note 1 = Comment 2 = Help 3 = Insert 4 = Key 5 = New paragraph 6 = Paragraph Add 100 to any of the above values to suppress the date shown in the popup annotation's title
PopupLeft	The horizontal co-ordinate of the left edge of the popup window
PopupTop	The vertical co-ordinate of the left edge of the popup window
PopupWidth	The width of the popup window
PopupHeight	The height of the popup window
Title	The title of the annotation
Contents	The body of the popup annotation
Red	The red component of the color of the annotation
Green	The green component of the color of the annotation
Blue	The blue component of the color of the annotation
Open	Specifies whether to show the annotation when the document is opened: 0 = hide 1 = show

AddOpenTypeFontFromFile

Version history

This function was introduced in Quick PDF Library version 8.12.

Description

This function is identical to **AddTrueTypeFontFromFile**. Both functions allow a TrueType, OpenType/TrueType or OpenType/CFF font to be added from a file.

This version of the function provides an Options parameter which may be expanded in future to support advanced OpenType features.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddOpenTypeFontFromFile(
  FileName: WideString; Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddOpenTypeFontFromFile(
  FileName As String, Options As Long) As Long
```

DLL

```
int DPLAddOpenTypeFontFromFile(int InstanceID, wchar_t * FileName,
  int Options);
```

Parameters

FileName	The font file name.
Options	Should be set to 0.

Return values

0	The font could not be embedded
Non-zero	The ID of the font that was successfully added. This ID can be used with the **SelectFont** function to select the font

AddOpenTypeFontFromString

Text, Fonts

Version history

This function was introduced in Quick PDF Library version 11.15.

Description

This function is identical to **AddTrueTypeFontFromString**. Both functions allow a TrueType, OpenType/TrueType or OpenType/CFF font to be added from a dta string.

This version of the function provides an Options parameter which may be expanded in future to support advanced OpenType features.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddOpenTypeFontFromString(
  const Source: AnsiString; Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddOpenTypeFontFromString(
  Source As String, Options As Long) As Long
```

DLL

```
int DPLAddOpenTypeFontFromString(int InstanceID, char * Source,
  int Options);
```

Parameters

Source	Font program data string.
Options	Should be set to 0.

Return values

0	The font could not be embedded
Non-zero	The ID of the font that was successfully added. This ID can be used with the **SelectFont** function to select the font

AddPageLabels

Description

Adds a range of page labels to the selected document. A range starting from page 1 must be present in the document for the page labels to display correctly.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddPageLabels(Start, Style,
  Offset: Integer; Prefix: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddPageLabels(Start As Long,
  Style As Long, Offset As Long, Prefix As String) As Long
```

DLL

```
int DPLAddPageLabels(int InstanceID, int Start, int Style, int Offset,
  wchar_t * Prefix);
```

Parameters

Start	The starting page for the range of page labels
Style	0 = No numbers 1 = Decimal arabic numerals 2 = Uppercase roman numerals 3 = Lowercase roman numerals 4 = Uppercase letters (A to Z for first 26 pages, AA to ZZ for next 26, etc.) 5 = Lowercase letters (a to z for first 26 pages, aa to zz for next 26, etc.)
Offset	The value of the numeric portion for the first page label in the range. Subsequent values will be numbered sequentially from this value, which much be greater than or equal to 1.
Prefix	The prefix for the page labels in this range.

Return values

0	The Style parameter was out of range
1	The page label range was added successfully

AddPageMatrix

Version history

This function was introduced in Quick PDF Library version 10.15.

Description

Function will scale the page contents in either direction and also move the page up, down, left or right. The parameters are in points where 72 points = 1 inch.

xscale = 1, yscale = 1 is the required for 100% scaling.
scale = 2 scale the width by a factor of 2 or 200%
xoffset = 72 moves the page 1 inch to the right. -72 1 inch to the left
yoffset = 72 moves the page 1 up and -72 moves the page 1 inch down

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddPageMatrix(xscale, yscale, xoffset,
  yoffset: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddPageMatrix(xscale As Double,
  yscale As Double, xoffset As Double, yoffset As Double) As Long
```

DLL

```
int DPLAddPageMatrix(int InstanceID, double xscale, double yscale,
  double xoffset, double yoffset);
```

Parameters

xscale	Horizontal scale
yscale	Vertical scale
xoffset	Horizontal offset
yoffset	Vertical offset

Return values

1	Page matrix added successfuly
0	Failed adding page matrix

AddRelativeLinkToFile

Version history

This function was introduced in Quick PDF Library version 11.11.

Description

Adds a clickable hotspot rectangle to the selected page which links using relative path to a specific page and position in another PDF document.

Use the **SetAnnotBorderColor** function to change the color of the hotspot border.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddRelativeLinkToFile(Left, Top, Width,
  Height: Double; FileName: WideString; Page: Integer; Position: Double;
  NewWindow, Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddRelativeLinkToFile(
  Left As Double, Top As Double, Width As Double,
  Height As Double, FileName As String, Page As Long,
  Position As Double, NewWindow As Long, Options As Long) As Long
```

DLL

```
int DPLAddRelativeLinkToFile(int InstanceID, double Left, double Top,
  double Width, double Height, wchar_t * FileName, int Page,
  double Position, int NewWindow, int Options);
```

Parameters

Left	The horizontal co-ordinate of the left edge of the hotspot rectangle
Top	The vertical co-ordinate of the top edge of the hotspot rectangle
Width	The width of the hotspot rectangle
Height	The height of the hotspot rectangle
FileName	The full absolute path and file name of the PDF document to link to, it will be converted to relative path.
Page	The page in the destination document to link to
Position	The vertical co-ordinate on the destination page to link to
NewWindow	0 = Close the current document and then open the new document 1 = Open the current document in a new window
Options	Specifies the appearance of the link: 0 = No border 1 = Draw a border

AddRelativeLinkToFileDest

Annotations and hotspot links

Version history

This function was introduced in Quick PDF Library version 11.11.

Description

Adds a clickable hotspot rectangle to the selected named destination which links to a specific page and position in another PDF document, using relative path.

Use the **SetAnnotBorderColor** function to change the color of the hotspot border.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddRelativeLinkToFileDest(Left, Top, Width,
  Height: Double; FileName, NamedDest: WideString; Position: Double;
  NewWindow, Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddRelativeLinkToFileDest(
  Left As Double, Top As Double, Width As Double,
  Height As Double, FileName As String, NamedDest As String,
  Position As Double, NewWindow As Long, Options As Long) As Long
```

DLL

```
int DPLAddRelativeLinkToFileDest(int InstanceID, double Left, double Top,
  double Width, double Height, wchar_t * FileName,
  wchar_t * NamedDest, double Position, int NewWindow,
  int Options);
```

Parameters

Left	The horizontal co-ordinate of the left edge of the hotspot rectangle
Top	The vertical co-ordinate of the top edge of the hotspot rectangle
Width	The width of the hotspot rectangle
Height	The height of the hotspot rectangle
FileName	The full absolute path and file name of the PDF document to link to, it will be converted to relative path.
NamedDest	The Named Destination string in the destination document to link to
Position	The vertical co-ordinate on the destination page to link to
NewWindow	0 = Close the current document and then open the new document 1 = Open the current document in a new window
Options	Specifies the appearance of the link: 0 = No border 1 = Draw a border

AddRelativeLinkToFileEx

Annotations and hotspot links

Version history

This function was introduced in Quick PDF Library version 11.11.

Description

Adds a clickable hotspot rectangle to the selected page which links using relative path to a specific page and position in another PDF document.

Use the **SetAnnotBorderColor** function to change the color of the hotspot border.

The link to the target document is only via the file name. This means the page dimensions of the target document are not known so the DestLeft, DestTop, DestRight and DestBottom parameters are always specified in points measured from the bottom left corner of the destination page's MediaBox.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddRelativeLinkToFileEx(Left, Top, Width,
  Height: Double; FileName: WideString; DestPage, NewWindow, Options,
  Zoom, DestType: Integer; DestLeft, DestTop, DestRight,
  DestBottom: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddRelativeLinkToFileEx(
  Left As Double, Top As Double, Width As Double,
  Height As Double, FileName As String, DestPage As Long,
  NewWindow As Long, Options As Long, Zoom As Long,
  DestType As Long, DestLeft As Double, DestTop As Double,
  DestRight As Double, DestBottom As Double) As Long
```

DLL

```
int DPLAddRelativeLinkToFileEx(int InstanceID, double Left, double Top,
  double Width, double Height, wchar_t * FileName, int DestPage,
  int NewWindow, int Options, int Zoom, int DestType,
  double DestLeft, double DestTop, double DestRight,
  double DestBottom);
```

Parameters

Left	The horizontal co-ordinate of the left edge of the hotspot rectangle
Top	The vertical co-ordinate of the top edge of the hotspot rectangle
Width	The width of the hotspot rectangle
Height	The height of the hotspot rectangle
FileName	The full absolute path and file name of the PDF document to link to, it will be converted to relative path.
DestPage	The page in the destination document to link to
NewWindow	0 = Close the current document and then open the new document 1 = Open the current document in a new window
Options	Specifies the appearance of the link: 0 = No border 1 = Draw a border
Zoom	The zoom percentage to use for the destination object, valid values from 0 to 6400. Only used for DestType = 1, should be set to 0 for other DestTypes.
DestType	1 = "XYZ" - the target page is positioned at the point specified by the Left and Top parameters. The Zoom parameter specifies the zoom percentage. 2 = "Fit" - the entire page is zoomed to fit the window. None of the other parameters are used and should be set to zero. 3 = "FitH" - the page is zoomed so that the entire width of the page is visible. The height of the page may be greater or less than the height of the window. The page is positioned at the vertical position specified by the Top parameter. 4 = "FitV" - the page is zoomed so that the entire height of the page can be seen. The width of the page may be greater or less than the width of the window. The page is positioned at the horizontal position specified by the Left parameter. 5 = "FitR" - the page is zoomed so that a certain rectangle on the page is visible. The Left, Top, Right and Bottom parameters define the rectangular area on the page. 6 = "FitB" - the page is zoomed so that it's bounding box is visible. 7 = "FitBH" - the page is positioned vertically at the position specified by the Top parameter. The page is zoomed so that the entire width of the page's bounding box is visible. 8 = "FitBV" - the page is positioned at the horizontal position specified by the Left parameter. The page is zoomed just enough to fit the entire height of the bounding box into the window.
DestLeft	The horizontal position used by DestType = 1, 4, 5 and 8
DestTop	The vertical position used by DestType = 1, 3, 5 and 7
DestRight	The horizontal position of the righthand edge of the rectangle. Used by DestType = 5
DestBottom	The horizontal position of the bottom of the rectangle. Used by DestType = 5

AddRelativeLinkToLocalFile

Annotations and hotspot links

Version history

This function was introduced in Quick PDF Library version 11.11.

Description

Adds a clickable hotspot rectangle to the selected page which links using relative path to a local file.

The file doesn't have to exist when the PDF is created but should exist when the PDF is viewed for the link to work.

Use the **SetAnnotBorderColor** function to change the color of the hotspot border.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddRelativeLinkToLocalFile(Left, Top,
  Width, Height: Double; FileName: WideString; Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddRelativeLinkToLocalFile(
  Left As Double, Top As Double, Width As Double,
  Height As Double, FileName As String, Options As Long) As Long
```

DLL

```
int DPLAddRelativeLinkToLocalFile(int InstanceID, double Left, double Top,
  double Width, double Height, wchar_t * FileName, int Options);
```

Parameters

Left	The horizontal co-ordinate of the left edge of the hotspot rectangle
Top	The vertical co-ordinate of the top edge of the hotspot rectangle
Width	The width of the hotspot rectangle
Height	The height of the hotspot rectangle
FileName	The full absolute path and file name of the PDF document to link to, it will be converted to relative path.
Options	Specifies the appearance of the link: 0 = No border 1 = Draw a border

AddSVGAnnotationFromFile

Description

Adds an SVG file as an annotation to the current page. This is only supported if the PDF is viewed using Adobe Acrobat 6 or Adobe Reader 6. Earlier and later versions will not show the SVG annotation.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddSVGAnnotationFromFile(Left, Top, Width,
  Height: Double; FileName: WideString; Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddSVGAnnotationFromFile(
  Left As Double, Top As Double, Width As Double,
  Height As Double, FileName As String, Options As Long) As Long
```

DLL

```
int DPLAddSVGAnnotationFromFile(int InstanceID, double Left, double Top,
  double Width, double Height, wchar_t * FileName, int Options);
```

Parameters

Left	The horizontal co-ordinate of the left edge of the annotation rectangle
Top	The vertical co-ordinate of the top edge of the annotation rectangle
Width	The width of the annotation rectangle
Height	The height of the annotation rectangle
FileName	The path and name of the file containing the SVG image.
Options	This parameter is ignored and should be set to 0

Return values

0	The SVG file could not be opened
1	The SVG annotation was added successfully

AddSWFAnnotationFromFile

Vector graphics, Image handling, Annotations and hotspot links, Page layout

Version history

This function was introduced in Quick PDF Library version 8.16.

Description

Adds a Flash SWF file as an annotation to the current page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddSWFAnnotationFromFile(Left, Top, Width,
  Height: Double; FileName, Title: WideString; Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddSWFAnnotationFromFile(
  Left As Double, Top As Double, Width As Double,
  Height As Double, FileName As String, Title As String,
  Options As Long) As Long
```

DLL

```
int DPLAddSWFAnnotationFromFile(int InstanceID, double Left, double Top,
  double Width, double Height, wchar_t * FileName,
  wchar_t * Title, int Options);
```

Parameters

Left	The horizontal co-ordinate of the left edge of the annotation rectangle
Top	The vertical co-ordinate of the top edge of the annotation rectangle
Width	The width of the annotation rectangle
Height	The height of the annotation rectangle
FileName	The path and name of the SWF file
Title	The annotation title
Options	Annotation event to activate SWF: 0 = Page visible 1 = Mouse enter 2 = Mouse button click

Return values

0	The specified file could not be found
1	The SWF was successfully added as an annotation

AddSeparationColor

Description

Adds a separation color to the document.

A separation color has a name and an equivalent color in the CMYK color space. If the document is viewed the CMYK color will be used. If the document is printed to an image setter a separation with the specified name will be generated.

The values of the color parameters range from 0 to 1, with 0 indicating 0% and 1 indicating 100% of the color.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddSeparationColor(ColorName: WideString;
  C, M, Y, K: Double; Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddSeparationColor(
  ColorName As String, C As Double, M As Double, Y As Double,
  K As Double, Options As Long) As Long
```

DLL

```
int DPLAddSeparationColor(int InstanceID, wchar_t * ColorName, double C,
  double M, double Y, double K, int Options);
```

Parameters

ColorName	The name of the separation color, for example "PANTONE 403 EC". This can be any name you want, but is usually set to the name of a specific spot color that your printing press will know what to do with.
C	The cyan component of the color equivalent to the spot color
M	The magenta component of the color equivalent to the spot color
Y	The yellow component of the color equivalent to the spot color
K	The black component of the color equivalent to the spot color
Options	This parameter is ignored and should be set to 0

Return values

0	The separation color could not be added. The color name may already have been used.
1	The separation color was added successfully

AddStampAnnotation

Annotations and hotspot links

Version history

This function was introduced in Quick PDF Library version 9.11.

Description

Adds a stamp annotation to the selected page. The values of the color parameters range from 0 to 1, with 0 indicating 0% and 1 indicating 100% of the color. The color only affects the background color of the popup and not the stamp itself.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddStampAnnotation(Left, Top, Width,
  Height: Double; StampType: Integer; Title, Contents: WideString; Red,
  Green, Blue: Double; Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddStampAnnotation(
  Left As Double, Top As Double, Width As Double,
  Height As Double, StampType As Long, Title As String,
  Contents As String, Red As Double, Green As Double,
  Blue As Double, Options As Long) As Long
```

DLL

```
int DPLAddStampAnnotation(int InstanceID, double Left, double Top,
  double Width, double Height, int StampType, wchar_t * Title,
  wchar_t * Contents, double Red, double Green, double Blue,
  int Options);
```

Parameters

Left	The horizontal coordinate of the left edge of the stamp annotation
Top	The vertical coordinate of the top edge of the stamp annotation
Width	The width of the annotation
Height	The height of the annotation
StampType	0 = Approved 1 = Experimental 2 = NotApproved 3 = AsIs 4 = Expired 5 = NotForPublicRelease 6 = Confidential 7 = Final 8 = Sold 9 = Departmental 10 = ForComment 11 = TopSecret 12 = Draft 13 = ForPublicRelease
Title	The title of the popup annotation
Contents	The contents of the popup annotation
Red	The red component of the popup annotation's background color
Green	The green component of the popup annotation's background color
Blue	The blue component of the popup annotation's background color
Options	Reserved for future use. Should always be set to 0.

Return values

0	The stamp annotation could not be added to the page
1	Success

AddStampAnnotationFromImage

Annotations and hotspot links

Version history

This function was introduced in Quick PDF Library version 10.13.

Description

Adds a custom stamp annotation to the selected page. The values of the color parameters range from 0 to 1, with 0 indicating 0% and 1 indicating 100% of the color. The color only affects the background color of the popup and not the stamp itself.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddStampAnnotationFromImage(Left, Top,
  Width, Height: Double; FileName, Title, Contents: WideString; Red,
  Green, Blue: Double; Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddStampAnnotationFromImage(
  Left As Double, Top As Double, Width As Double,
  Height As Double, FileName As String, Title As String,
  Contents As String, Red As Double, Green As Double,
  Blue As Double, Options As Long) As Long
```

DLL

```
int DPLAddStampAnnotationFromImage(int InstanceID, double Left,
  double Top, double Width, double Height, wchar_t * FileName,
  wchar_t * Title, wchar_t * Contents, double Red, double Green,
  double Blue, int Options);
```

Parameters

Left	The horizontal coordinate of the left edge of the stamp annotation
Top	The vertical coordinate of the top edge of the stamp annotation
Width	The width of the annotation
Height	The height of the annotation
FileName	Complete FilePath to the image
Title	The title of the popup annotation
Contents	The contents of the popup annotation
Red	The red component of the popup annotation's background color
Green	The green component of the popup annotation's background color
Blue	The blue component of the popup annotation's background color
Options	Reserved for future use. Should always be set to 0.

Return values

0	The stamp annotation could not be added to the page
1	Success

AddStampAnnotationFromImageID

Annotations and hotspot links

Version history

This function was introduced in Quick PDF Library version 10.14.

Description

Adds a custom stamp annotation to the selected page based on the image ID that is already added to the document. The values of the color parameters range from 0 to 1, with 0 indicating 0% and 1 indicating 100% of the color. The color only affects the background color of the popup and not the stamp itself.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddStampAnnotationFromImageID(Left, Top,
  Width, Height: Double; ImageID: Integer; Title, Contents: WideString;
  Red, Green, Blue: Double; Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddStampAnnotationFromImageID(
  Left As Double, Top As Double, Width As Double,
  Height As Double, ImageID As Long, Title As String,
  Contents As String, Red As Double, Green As Double,
  Blue As Double, Options As Long) As Long
```

DLL

```
int DPLAddStampAnnotationFromImageID(int InstanceID, double Left,
  double Top, double Width, double Height, int ImageID,
  wchar_t * Title, wchar_t * Contents, double Red, double Green,
  double Blue, int Options);
```

Parameters

Left	The horizontal coordinate of the left edge of the stamp annotation
Top	The vertical coordinate of the top edge of the stamp annotation
Width	The width of the annotation
Height	The height of the annotation
ImageID	ID of the image that should be used as stamp
Title	The title of the popup annotation
Contents	The contents of the popup annotation
Red	The red component of the popup annotation's background color
Green	The green component of the popup annotation's background color
Blue	The blue component of the popup annotation's background color
Options	Reserved for future use. Should always be set to 0.

Return values

0	The stamp annotation could not be added to the page
1	Success

AddStandardFont

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Adds a standard font to the document. These standard fonts will always be available on all PDF viewers.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddStandardFont(
  StandardFontID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddStandardFont(
  StandardFontID As Long) As Long
```

DLL

```
int DPLAddStandardFont(int InstanceID, int StandardFontID);
```

Parameters

StandardFontID	The ID of the font to add: 0 = Courier 1 = CourierBold 2 = CourierBoldOblique 3 = CourierOblique 4 = Helvetica 5 = HelveticaBold 6 = HelveticaBoldOblique 7 = HelveticaOblique 8 = TimesRoman 9 = TimesBold 10 = TimesItalic 11 = TimesBoldItalic 12 = Symbol 13 = ZapfDingbats

Return values

0	The font could not be added
Non-zero	The ID of the font that was successfully added

AddSubsettedFont

DEBENU
QUICKPDF
LIBRARY

Description

This function is used to embed a "subset" of a font. This means that only the font information for specified characters is embedded, reducing the size of the document. This function also allows any Unicode character to be embedded which means that characters from Chinese, Japanese, Korean and other languages can be used.

The newer **AddTrueTypeSubsettedFont** function provides more advanced font subsetting functionality.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddSubsettedFont(FontName: WideString;
  CharsetIndex: Integer; SubsetChars: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddSubsettedFont(
  FontName As String, CharsetIndex As Long,
  SubsetChars As String) As Long
```

DLL

```
int DPLAddSubsettedFont(int InstanceID, wchar_t * FontName,
  int CharsetIndex, wchar_t * SubsetChars);
```

Parameters

FontName	The name of the TrueType font to install. This can either be the name of the font as shown in the Windows\Fonts folder (for example "Times New Roman Bold") or it can be the font family name with an optional style specifier in square brackets (for example "Times New Roman [BoldItalic]"). Possible optional specifiers are: [Bold], [Italic] or [BoldItalic].
CharsetIndex	You must specify a character set containing the characters you want to subset: 1 = ANSI 2 = Default 3 = Symbol 4 = Shift JIS 5 = Hangeul 6 = GB2312 7 = Chinese Big 5 8 = OEM 9 = Johab 10 = Hebrew 11 = Arabic 12 = Greek 13 = Turkish 14 = Vietnamese 15 = Thai 16 = East Europe 17 = Russian 18 = Mac 19 = Baltic
SubsetChars	A string containing the characters you would like to subset. Repeated characters are ignored. A maximum of 255 characters can be placed in any font subset. Any Unicode character can be embedded, but you must ensure that the character is available in the specified character set.

Return values

0	The subsetted font could not be added or the CharSet parameter was out of range
Non-zero	The FontID of the added font. This ID can be used with the **SelectFont** function to select the font.

AddTextMarkupAnnotation

Annotations and hotspot links

Version history

This function was introduced in Quick PDF Library version 7.21.

Description

Adds a text markup annotation to the current page.

By default the annotation will consist of a single rectangular area matching the annotation's bounding box. This area can be edited and other areas can be added using the **GetAnnotQuadCount**, **GetAnnotQuadPoints** and **SetAnnotQuadPoints** functions.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddTextMarkupAnnotation(
  MarkupType: Integer; Left, Top, Width, Height: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddTextMarkupAnnotation(
  MarkupType As Long, Left As Double, Top As Double,
  Width As Double, Height As Double) As Long
```

DLL

```
int DPLAddTextMarkupAnnotation(int InstanceID, int MarkupType,
  double Left, double Top, double Width, double Height);
```

Parameters

MarkupType	0 = Highlight 1 = Underline 2 = Squiggly 3 = Strike out
Left	The horizontal co-ordinate of the left edge of the annotation bounding box
Top	The vertical co-ordinate of the top edge of the annotation bounding box
Width	The width of the annotation bounding box
Height	The height of the annotation bounding box

Return values

0	The MarkupType parameter was not between 1 and 4.
1	The text markup annotation was added successfully.

AddToBuffer

Version history

This function was introduced in Quick PDF Library version 7.11.

Description

Adds a block of data to the buffer created with the **CreateBuffer** function.

This function can be called multiple times until the buffer is full. The return value is the number of bytes remaining in the buffer.

Syntax

DLL

```
int DPLAddToBuffer(int InstanceID, char * Buffer, char * Source,
  int SourceLength);
```

Parameters

Buffer	A value returned from the **CreateBuffer** function
Source	A pointer to the first byte of data to add
SourceLength	The total number of bytes to copy from the source

AddToFileList

Description

Adds a file to a named file list. This file list can later be used with functions that will operate on all the files in the list.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddToFileList(ListName,
  FileName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddToFileList(
  ListName As String, FileName As String) As Long
```

DLL

```
int DPLAddToFileList(int InstanceID, wchar_t * ListName,
  wchar_t * FileName);
```

Parameters

ListName	The name of the file list to work with
FileName	The file name to add to the list.

AddTrueTypeFont

Description

Adds a TrueType font to the document. The font must be installed on the system. If the font is not embedded, then the reader of the PDF document must have the font installed on their system too. If the font is embedded, then the reader does not need the font installed on their system. Embedding a font makes the PDF file much larger. Some fonts are not licensed to be embedded.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddTrueTypeFont(FontName: WideString;
  Embed: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddTrueTypeFont(
  FontName As String, Embed As Long) As Long
```

DLL

```
int DPLAddTrueTypeFont(int InstanceID, wchar_t * FontName, int Embed);
```

Parameters

FontName	The name of the TrueType font to install. This can either be the name of the font as shown in the Windows\Fonts folder (for example "Times New Roman") or it can be the font family name with an optional style specifier in square brackets (for example "Times New Roman [BoldItalic]"). Possible optional specifiers are: [Bold], [Italic] or [BoldItalic]. A codepage can also be specified (for example "Arial [Bold] {1250}") which allows other encodings to be used. Possible code pages are: {0} Direct mapping {437} OEM_CHARSET {850} OEM_CHARSET {852} OEM_CHARSET {874} THAI_CHARSET {1250} EASTEUROPE_CHARSET {1251} RUSSIAN_CHARSET {1252} ANSI_CHARSET {1253} GREEK_CHARSET {1254} TURKISH_CHARSET {1255} HEBREW_CHARSET {1256} ARABIC_CHARSET {1257} BALTIC_CHARSET {1258} VIETNAMESE_CHARSET {1361} JOHAB_CHARSET Note: {932}, {936}, {949} and {950} are not supported from version 8.11
Embed	Specifies whether to embed the font or not: 0 = Don't embed the font 1 = Embed the font

Return values

0	The font could not be added. This may mean that the font is not licensed to be embedded, or that the font could not be found.
Non-zero	The ID of the font that was successfully added. This ID can be used with the **SelectFont** function to select the font

AddTrueTypeFontFromFile

Description

Embeds a TrueType, OpenType/TrueType or OpenType/CFF font into the document. The TrueType font is specified by the file name and does not have to be installed as a system font.

This function is functionally identical to **AddOpenTypeFontFromFile**.

For TrueType and OpenType/TrueType fonts, a temporary file must be created during this process, call **SetTempPath** to specify where this temporary file should be created.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddTrueTypeFontFromFile(
    FileName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddTrueTypeFontFromFile(
    FileName As String) As Long
```

DLL

```
int DPLAddTrueTypeFontFromFile(int InstanceID, wchar_t * FileName);
```

Parameters

FileName	The full path and file name of the TrueType font file to embed.

Return values

0	The font could not be embedded
Non-zero	The ID of the font that was successfully added. This ID can be used with the **SelectFont** function to select the font

AddTrueTypeFontFromString

Version history

This function was introduced in Quick PDF Library version 11.15.

Description

Embeds a TrueType, OpenType/TrueType or OpenType/CFF font into the document. The TrueType font is specified by the data string and does not have to be installed as a system font.

This function is functionally identical to **AddOpenTypeFontFromString**.

For TrueType and OpenType/TrueType fonts, a temporary file must be created during this process, call **SetTempPath** to specify where this temporary file should be created.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddTrueTypeFontFromString(
  const Source: AnsiString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddTrueTypeFontFromString(
  Source As String) As Long
```

DLL

```
int DPLAddTrueTypeFontFromString(int InstanceID, char * Source);
```

Parameters

Source	String that contains data of the TrueType font program to embed.

Return values

0	The font could not be embedded
Non-zero	The ID of the font that was successfully added. This ID can be used with the **SelectFont** function to select the font

AddTrueTypeSubsettedFont

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Adds a subsetted TrueType font to the document.

For Options 0 and 1 the font subset is fixed and cannot be changed.

For Options 2 and 3 the font subset can be changed Similar to 0 but subset can be updated using **UpdateTrueTypeSubsettedFont**

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddTrueTypeSubsettedFont(FontName,
  SubsetChars: WideString; Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddTrueTypeSubsettedFont(
  FontName As String, SubsetChars As String,
  Options As Long) As Long
```

DLL

```
int DPLAddTrueTypeSubsettedFont(int InstanceID, wchar_t * FontName,
  wchar_t * SubsetChars, int Options);
```

Parameters

FontName	The name of the TrueType font that must be subsetted.
SubsetChars	A string containing the characters that should be included in the font subset.
Options	0=MS PlatformID, Unicode charset 1=Unicode PlatformID, "don't care" charset 2=Similar to 0 but subset can be updated using **UpdateTrueTypeSubsettedFont** 3=Similar to 1 but subset can be updated using **UpdateTrueTypeSubsettedFont** 4=Similar to 2 but subset is automatically updated 5=Similar to 3 but subset is automatically updated

Return values

0	The subsetted font could not be added or the CharSet parameter was out of range
Non-zero	The ID of the font that was successfully added. This ID can be used with the **SelectFont** function to select the font

AddType1Font

Description

Adds a PostScript Type1 font to the document. The font must be supplied as two files, a .pfm and a .pfb file. The full path to the .pfm file must be supplied. The font is embedded in the document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddType1Font(FileName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddType1Font(
  FileName As String) As Long
```

DLL

```
int DPLAddType1Font(int InstanceID, wchar_t * FileName);
```

Parameters

FileName	The full path to the .pfm file. A .pfb file with the same name should exist in the same directory as the .pfm file.

Return values

0	The font could not be added. Either the font files are in the wrong format, or they cannot be found.
Non-zero	The ID of the font that was successfully added. This ID can be used with the **SelectFont** function to select the font

AddU3DAnnotationFromFile

Vector graphics, Image handling, Annotations and hotspot links, Page layout

Version history

This function was introduced in Quick PDF Library version 7.12.

Description

Adds an SVG file as an annotation to the current page. The SVG annotation will only be visible if the PDF is viewed with Adobe Acrobat 7 or higher.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AddU3DAnnotationFromFile(Left, Top, Width,
  Height: Double; FileName: WideString; Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AddU3DAnnotationFromFile(
  Left As Double, Top As Double, Width As Double,
  Height As Double, FileName As String, Options As Long) As Long
```

DLL

```
int DPLAddU3DAnnotationFromFile(int InstanceID, double Left, double Top,
  double Width, double Height, wchar_t * FileName, int Options);
```

Parameters

Left	The horizontal co-ordinate of the left edge of the annotation rectangle
Top	The vertical co-ordinate of the top edge of the annotation rectangle
Width	The width of the annotation rectangle
Height	The height of the annotation rectangle
FileName	The path and name of the file containing the U3D model.
Options	0 = the 3D annotation is static 1 = the 3D annotation is interactive

AnalyseFile

Description

Analyses a file on disk. The entire file is not loaded into memory so huge files can be examined. Use the **GetAnalysisInfo** function to retrieve the individual analysis results. Call **DeleteAnalysis** to remove the results from memory when you are finished.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AnalyseFile(InputFileName,
  Password: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AnalyseFile(
  InputFileName As String, Password As String) As Long
```

DLL

```
int DPLAnalyseFile(int InstanceID, wchar_t * InputFileName,
  wchar_t * Password);
```

Parameters

InputFileName	The path and name of the file to analyse.
Password	The password to use when opening the file. This can be either the owner or the user password. This parameter can be left blank if the file does not require a password to be opened.

Return values

0	The file could not be analysed. Check the result of the **LastErrorCode** function to determine the reason for the failure.
Non-zero	The analysis results ID. Pass this to the **GetAnalysisInfo** function.

AnnotationCount

Annotations and hotspot links

Description

Returns the number of annotations on the selected page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AnnotationCount: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AnnotationCount As Long
```

DLL

```
int DPLAnnotationCount(int InstanceID);
```

AnsiStringResultLength

Description

Returns the length of the most recent string returned from the library by all functions that return 8-bit strings.

Syntax

DLL

```
int DPLAnsiStringResultLength(int InstanceID);
```

AppendSpace

Description

Moves the current text position horizontally by a percentage of the height of the text.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AppendSpace(RelativeSpace: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AppendSpace(
  RelativeSpace As Double) As Long
```

DLL

```
int DPLAppendSpace(int InstanceID, double RelativeSpace);
```

Parameters

RelativeSpace	A value of 1 moves the horizontal position by a value equal to the height of the text at the present font size, also known as an EM space. A value of 0.5 moves the horizontal position by half the height of the text at the present font size, also known as an EN space.

AppendTableColumns

Version history

This function was introduced in Quick PDF Library version 7.16.

Description

Adds columns to the right of the specified table

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AppendTableColumns(TableID,
  NewColumnCount: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AppendTableColumns(
  TableID As Long, NewColumnCount As Long) As Long
```

DLL

```
int DPLAppendTableColumns(int InstanceID, int TableID, int NewColumnCount);
```

Parameters

TableID	A TableID returned by the **CreateTable** function
NewColumnCount	The number of columns to add to the table

Return values

0	Columns could not be added. Check the TableID parameter and make sure NewColumnCount is greater than or equal to 1.
Non-zero	The total number of columns in the table after adding the new columns.

AppendTableRows

Version history

This function was introduced in Quick PDF Library version 7.16.

Description

Adds rows to the bottom of the specified table.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AppendTableRows(TableID,
  NewRowCount: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AppendTableRows(TableID As Long,
  NewRowCount As Long) As Long
```

DLL

```
int DPLAppendTableRows(int InstanceID, int TableID, int NewRowCount);
```

Parameters

TableID	A TableID returned by the **CreateTable** function
NewRowCount	The number of rows to add to the table

Return values

0	Rows could not be added. Check the TableID parameter and make sure NewRowCount is greater than or equal to 1.
Non-zero	The total number of rows in the table after adding the new rows.

AppendText

Description

Draws text immediately following text previously drawn with **DrawText** or **AppendText**.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AppendText(Text: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AppendText(
  Text As String) As Long
```

DLL

```
int DPLAppendText(int InstanceID, wchar_t * Text);
```

Parameters

Text	The text to append to the previously drawn text

AppendToFile

Version history

This function was introduced in Quick PDF Library version 10.11.

Description

Appends the changed objects to the specified file in an incremental update.

The file name specified should be the same file that was the source of the document in the earlier call to **LoadFromFile**, **LoadFromString** or **LoadFromStream**.

Appending to a different file will result in a corrupt PDF.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AppendToFile(FileName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AppendToFile(
  FileName As String) As Long
```

DLL

```
int DPLAppendToFile(int InstanceID, wchar_t * FileName);
```

Parameters

FileName	The name of the file to create

Return values

0	The incremental update could not be appended to the specified file
1	Success

AppendToString

Version history

This function was introduced in Quick PDF Library version 11.11.

Description

Appends the changed objects to a string in an incremental update.

The update must be made to the same input file from a previous call to **LoadFromFile**, **LoadFromString** or **LoadFromStream**.

The AppendMode parameter can be used to change how the update section is returned. Either form the original input or from the input set by the **SetAppendInputFromString** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AppendToString(
  AppendMode: Integer): AnsiString;
```

DLL

```
char * DPLAppendToString(int InstanceID, int AppendMode);
```

Parameters

AppendMode	0 = Return original source plus the update section
	1 = Return just the update section
	2 = Return input string set with **SetAppendInputFromString** plus the update section

AppendToVariant

Version history

This function was introduced in Quick PDF Library version 11.11.

Description

Appends the changed objects to a variant byte array in an incremental update.

The update must be made to the same input file from a previous call to **LoadFromFile**, **LoadFromVariant** or **LoadFromStream**.

The AppendMode parameter can be used to change how the update section is returned. Either form the original input or from the input set by the **SetAppendInputFromVariant** function.

Syntax

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AppendToVariant(
  AppendMode As Long) As Variant
```

Parameters

AppendMode	0 = Return original source plus the update section 1 = Return just the update section 2 = Return input data set with **SetAppendInputFromVariant** plus the update section

ApplyStyle

Text, Page layout

Description

Applies a style that was previously saved using the **SaveStyle** function. The style name is case sensitive, it must exactly match the style name used with the **SaveStyle** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ApplyStyle(StyleName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ApplyStyle(
  StyleName As String) As Long
```

DLL

```
int DPLApplyStyle(int InstanceID, wchar_t * StyleName);
```

Parameters

StyleName	The name to associate with the style. This name is case sensitive.

Return values

0	The specified StyleName could not be found
1	The style was applied successfully

AttachAnnotToForm

Version history

> This function was introduced in Quick PDF Library version 7.18.

Description

> This functions attaches an annotation to the document form.

> Use the **IsAnnotFormField** function to check if the specified annotation can be attached to the document form and whether it is currently attached or not.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.AttachAnnotToForm(Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::AttachAnnotToForm(
  Index As Long) As Long
```

DLL

```
int DPLAttachAnnotToForm(int InstanceID, int Index);
```

Parameters

Index	The index of the annotation. The first annotation on the page has an index of 1.

Return values

0	The specified annotation could not be attached to the document form.
1	The specified annotation was attached successfully to the document form.

BalanceContentStream

Version history

This function was introduced in Quick PDF Library version 9.11.

Description

This function combines the content stream parts and surrounds the content stream with "save graphics state" and "restore graphics state" operators.

If the page contains unbalanced "save graphics state" and "restore graphics state" commands this function will add extra "restore graphics state" commands at the end of the page to balance the graphics state stack.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.BalanceContentStream: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::BalanceContentStream As Long
```

DLL

```
int DPLBalanceContentStream(int InstanceID);
```

BalancePageTree

Version history

This function was introduced in Quick PDF Library version 9.15.

Description

Arranges the selected document's internal page structure into a balanced tree for faster random access to pages in the document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.BalancePageTree(Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::BalancePageTree(
  Options As Long) As Long
```

DLL

```
int DPLBalancePageTree(int InstanceID, int Options);
```

Parameters

Options	Reserved for future use, should be set to zero.

Return values

0	The page tree could not be balanced
1	Success

BeginPageUpdate

Version history

This function was introduced in Quick PDF Library version 7.12.

Description

For detailed page layouts this function can be called before a group of drawing commands. The page layout commands will then be buffered until a matching call to the **EndPageUpdate** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.BeginPageUpdate: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::BeginPageUpdate As Long
```

DLL

```
int DPLBeginPageUpdate(int InstanceID);
```

CapturePage

Description

This function "captures" a page. Once the page has been captured it can be drawn onto other pages. This is useful for combining different pages or for placing more than one original page onto another page (imposition). Once a page has been captured it is removed from the document. If you would like the page to remain in the document you must create a blank page and draw the captured page onto the blank page.

Also, because a document must have at least one page at all times it is not possible to capture a page if it is the only page in the document. In this case, you must add a new blank page before the existing page can be captured.

You cannot use CapturePage to move pages from one document to another new document so all the required pages must be merged into a single document before calling CapturePage. The CaptureID is just a pointer to a hidden page therefore memory does not need to be released.

The "media box" for the page is used as the bounding rectangle for the captured page. The **CapturePage** function can be used in cases where the "crop box" for the page should be used instead.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.CapturePage(Page: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::CapturePage(Page As Long) As Long
```

DLL

```
int DPLCapturePage(int InstanceID, int Page);
```

Parameters

Page	The page number to capture. The first page in the document is page 1.

Return values

0	The specified page does not exist, or it is the only page in the document
Non-zero	The ID of the capture process. This ID must be supplied to the **DrawCapturedPage** function.

CapturePageEx

Description

This function "captures" a page. Once the page has been captured it can be drawn onto other pages. This is useful for combining different pages or for placing more than one original page onto another page (imposition). Once a page has been captured it is removed from the document. If you would like the page to remain in the document you must create a blank page and draw the captured page onto the blank page.

Also, because a document must have at least one page at all times it is not possible to capture a page if it is the only page in the document. In this case, you must add a new blank page before the existing page can be captured.

You cannot use CapturePage to move pages from one document to another new document so all the required pages must be merged into a single document before calling CapturePage. The CaptureID is just a pointer to a hidden page therefore memory does not need to be released.

The "media box" for the page is used as the bounding rectangle for the captured page. The **CapturePageEx** function can be used in cases where the "crop box" for the page should be used instead.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.CapturePageEx(Page,
  Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::CapturePageEx(Page As Long,
  Options As Long) As Long
```

DLL

```
int DPLCapturePageEx(int InstanceID, int Page, int Options);
```

Parameters

Page	The page number to capture. The first page in the document is page 1.
Options	0 = Use the page's media box for the bounding rectangle 1 = Use the page's crop box for the bounding rectangle if it has one, otherwise use the media box 2 = Use the page's bleed box for the bounding rectangle if it has one, otherwise use the crop box 3 = Use the page's trim box for the bounding rectangle if it has one, otherwise use the crop box 4 = Use the page's art box for the bounding rectangle if it has one, otherwise use the crop box

Return values

0	The specified page does not exist, or it is the only page in the document
Non-zero	The ID of the capture process. This ID must be supplied to the **DrawCapturedPage** function.

CharWidth

Description

Returns the width of a character for the selected font.

This width is returned as a ratio to the text size. For example, if this function returns 750 for a certain character, then the width of the character for a 12 point font will be (750 / 1000) * 12.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.CharWidth(CharCode: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::CharWidth(
  CharCode As Long) As Long
```

DLL

```
int DPLCharWidth(int InstanceID, int CharCode);
```

Parameters

CharCode   The character to determine the width for. For example, 65 is the character A.

Return values

The width of the specified character. Divide this value by 1000, and multiply by the text size in points to get the width of the character.

CheckFileCompliance

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

This function tests a PDF document against various standards to determine compliance with the standard.

This function is currently under development and currently runs only a small subset of possible tests. It should not be used in real world applications.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.CheckFileCompliance(InputFileName,
  Password: WideString; ComplianceTest, Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::CheckFileCompliance(
  InputFileName As String, Password As String,
  ComplianceTest As Long, Options As Long) As Long
```

DLL

```
int DPLCheckFileCompliance(int InstanceID, wchar_t * InputFileName,
  wchar_t * Password, int ComplianceTest, int Options);
```

Parameters

InputFileName	The file to check
Password	The password to open the file. If there is no password an empty string should be used.
ComplianceTest	1 = PDF/A compliance test
Options	For PDF/A compliance test: 0 = Show all errors 1 = Stop after the first error

Return values

0	The file passed the compliance test.
Non-zero	A StringListID that can be used with the **GetStringListCount** and **GetStringListItem** functions.

CheckObjects

Description

Checks the file to ensure all objects are valid. This may take some time with large files and consume large amounts of memory.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.CheckObjects: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::CheckObjects As Long
```

DLL

```
int DPLCheckObjects(int InstanceID);
```

CheckPageAnnots

Description

Checks all the annotations on the selected page and ensures that they are all valid. Invalid annotations are removed from the page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.CheckPageAnnots: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::CheckPageAnnots As Long
```

DLL

```
int DPLCheckPageAnnots(int InstanceID);
```

Return values

0	No annotations were found to be in an incorrect format
1	One or more annotations were not in the correct format and were unlinked from the page

CheckPassword

Version history

This function was introduced in Quick PDF Library version 10.12.

Description

Determines if a password is a valid password for the selected document.

This is useful when the document has been opened with the user password but confirmation should be obtained from the user before changing security settings.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.CheckPassword(
  Password: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::CheckPassword(
  Password As String) As Long
```

DLL

```
int DPLCheckPassword(int InstanceID, wchar_t * Password);
```

Parameters

Password	The password to check

Return values

0	The document is not encrypted or the supplied password is not a valid owner or user password
1	Valid user password
2	Valid owner password
3	Valid owner and user password

ClearFileList

Description

Clears a named file list.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ClearFileList(
  ListName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ClearFileList(
  ListName As String) As Long
```

DLL

```
int DPLClearFileList(int InstanceID, wchar_t * ListName);
```

Parameters

ListName	The name of the file list to clear

Return values

0	The named list could not be found
1	The named list was cleared successfully

ClearImage

Version history

> This function was introduced in Quick PDF Library version 7.21.

Description

> Clears the specified image.

> To prevent the corruption of existing links to the image it will not be deleted from the document. The image will be converted into a 24-bit RGB format consisting of a single transparent pixel.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ClearImage(ImageID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ClearImage(
  ImageID As Long) As Long
```

DLL

```
int DPLClearImage(int InstanceID, int ImageID);
```

Parameters

ImageID	The ImageID of the image to be cleared

Return values

0	The specified ImageID was not valid
1	The image was cleared

ClearPageLabels

Description

Removes all the page labels from the selected document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ClearPageLabels: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ClearPageLabels As Long
```

DLL

```
int DPLClearPageLabels(int InstanceID);
```

ClearTextFormatting

Description

Clears any formatting that has been applied. Subsequently drawn text will be drawn left aligned in black with all highlighting, underlining, character spacing, word spacing, horizontal scaling and vertical spacing removed.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ClearTextFormatting: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ClearTextFormatting As Long
```

DLL

```
int DPLClearTextFormatting(int InstanceID);
```

CloneOutlineAction

Version history

This function was introduced in Quick PDF Library version 7.16.

Description

Calling this function will clone the action dictionary of the specified outline. This is useful when an outline and an annotation share the same action dictionary and the actions must be set individually.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.CloneOutlineAction(
  OutlineID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::CloneOutlineAction(
  OutlineID As Long) As Long
```

DLL

```
int DPLCloneOutlineAction(int InstanceID, int OutlineID);
```

Parameters

OutlineID	The ID of the outline as returned by the **NewOutline** function. Alternatively, use the **GetOutlineID** function to get a valid outline ID.

ClonePages

Description

Copies pages from the document multiple times, with only a negligible increase in file size. Note that only the first "layer" of the page is cloned. Unless you specifically want to take part of the page you should call **CombineContentStreams** for all the pages you want to clone before calling this function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ClonePages(StartPage, EndPage,
  RepeatCount: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ClonePages(StartPage As Long,
  EndPage As Long, RepeatCount As Long) As Long
```

DLL

```
int DPLClonePages(int InstanceID, int StartPage, int EndPage,
  int RepeatCount);
```

Parameters

StartPage	The first page to clone
EndPage	The last page to clone
RepeatCount	The number of times to clone the pages

Return values

0	The parameters were out of range
1	The function was successful

CloseOutline

Description

Collapses an outline item (bookmark).

Syntax

Delphi

```
function TDebenuPDFLibrary1115.CloseOutline(OutlineID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::CloseOutline(
  OutlineID As Long) As Long
```

DLL

```
int DPLCloseOutline(int InstanceID, int OutlineID);
```

Parameters

OutlineID	The ID of the outline as returned by the **NewOutline** function. Alternatively, use the **GetOutlineID** function to get a valid outline ID.

Return values

0	The Outline ID provided was invalid
1	The outline item was collapsed

ClosePath

Description

Closes the path defined by calls to **StartPath**, **AddLineToPath**, and **AddCurveToPath**. A line is drawn from the last point to the first point.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ClosePath: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ClosePath As Long
```

DLL

```
int DPLClosePath(int InstanceID);
```

CombineContentStreams

Version history

This function was renamed in Quick PDF Library version 8.11.
The function name in earlier versions was CombineLayers.

Description

A page in a PDF document has one or more content stream parts that together contain all the PDF page description commands for the page.

This function combines all the content stream parts of the selected page into a single content stream.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.CombineContentStreams: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::CombineContentStreams As Long
```

DLL

```
int DPLCombineContentStreams(int InstanceID);
```

Return values

0	The content stream could not be combined
1	The content stream was combined successfully

CompareOutlines

Version history

This function was introduced in Quick PDF Library version 7.22.

Description

Compares two OutlineID values.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.CompareOutlines(FirstOutlineID,
  SecondOutlineID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::CompareOutlines(
  FirstOutlineID As Long, SecondOutlineID As Long) As Long
```

DLL

```
int DPLCompareOutlines(int InstanceID, int FirstOutlineID,
  int SecondOutlineID);
```

Parameters

FirstOutlineID	The first OutlineID to compare
SecondOutlineID	The second OutlineID to compare

Return values

0	One or both of the OutlineID values were not valid or there is no relationship between the two outlines.
1	The OutlineID values refer to the same outline item.

CompressContent

Description

Compresses the content of the selected document. The Flate algorithm is used to compress the content.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.CompressContent: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::CompressContent As Long
```

DLL

```
int DPLCompressContent(int InstanceID);
```

Return values

0	The content could not be compressed
1	The content was compressed successfully

CompressFonts

Description

Specifies whether or not to compress TrueType, Packaged and Type1 fonts subsequently added to the document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.CompressFonts(Compress: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::CompressFonts(
  Compress As Long) As Long
```

DLL

```
int DPLCompressFonts(int InstanceID, int Compress);
```

Parameters

Compress	0 = Don't compress fonts 1 = Compress all subsequently added fonts

Return values

0	The Compress parameter was out of range
1	The font compression setting was changed successfully

CompressImages

Description

Specifies the compression to use for images added to the document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.CompressImages(Compress: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::CompressImages(
  Compress As Long) As Long
```

DLL

```
int DPLCompressImages(int InstanceID, int Compress);
```

Parameters

Compress	0 = No compression 1 = Flate compression

Return values

0	The Compress parameter was not valid
1	The image compression was set successfully

CompressPage

Description

This function is similar to the **CompressContent** function, however it only compresses the selected page. Looping through all the pages using this function will have the same effect as **CompressContent**, however it will be possible to provide feedback to the user.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.CompressPage: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::CompressPage As Long
```

DLL

```
int DPLCompressPage(int InstanceID);
```

ContentStreamCount

Content Streams and Optional Content Groups

Version history

This function was renamed in Quick PDF Library version 8.11.
The function name in earlier versions was LayerCount.

Description

A page in a PDF document has one or more content stream parts that together contain all the PDF page description commands for the page.

This function returns the total number of content stream parts for the selected page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ContentStreamCount: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ContentStreamCount As Long
```

DLL

```
int DPLContentStreamCount(int InstanceID);
```

Return values

The number of content stream parts on the selected page

ContentStreamSafe

Content Streams and Optional Content Groups

Version history

This function was renamed in Quick PDF Library version 8.11.
The function name in earlier versions was LayerSafe.

Description

A page in a PDF document has one or more content stream parts that together contain all the PDF page description commands for the page.

This function determines if the content stream part that was selected using the **SelectContentStream** function was created by Quick PDF Library or not.

Only content stream parts created by Quick PDF Library should be considered "safe" to drawn on. If a content stream part is not safe it would be best to combine all the content stream parts using the **CombineContentStreams** function before drawing on the page to prevent later errors in the document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ContentStreamSafe: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ContentStreamSafe As Long
```

DLL

```
int DPLContentStreamSafe(int InstanceID);
```

Return values

0	The layer was not created by Quick PDF Library and care should be taken when drawing onto this layer
1	The layer was created by Quick PDF Library and is safe to draw on

CopyPageRanges

Description

Use this function to copy one or more pages from one document to another.

The pages are copied in sequential order and duplicates are not allowed. To extract pages in a different order to the source document or with duplicate pages the **CopyPageRangesEx** function can be used.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.CopyPageRanges(DocumentID: Integer;
  RangeList: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::CopyPageRanges(
  DocumentID As Long, RangeList As String) As Long
```

DLL

```
int DPLCopyPageRanges(int InstanceID, int DocumentID, wchar_t * RangeList);
```

Parameters

DocumentID	The ID of the document to copy the pages from
RangeList	The pages to extract, for example "10,15,18-20,25-35". Invalid characters and duplicate page numbers in the string will be ignored. Reversed page ranges such as "5-1" will be accepted. The list of pages will be sorted resulting in the pages being extracted in numerical order.

Return values

0	The specified DocumentID was not valid or was the same as the selected document, or the RangeList was invalid
1	The pages were successfully copied from the specified document to the selected document

CopyPageRangesEx

Version history

This function was introduced in Quick PDF Library version 9.11.

Description

Use this function to copy one or more pages from one document to another. It is functionality identical to the **CopyPageRanges** function but adds an option to allow the page list to contain duplicate page numbers and a different page order to the original document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.CopyPageRangesEx(DocumentID: Integer;
  RangeList: WideString; Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::CopyPageRangesEx(
  DocumentID As Long, RangeList As String,
  Options As Long) As Long
```

DLL

```
int DPLCopyPageRangesEx(int InstanceID, int DocumentID,
  wchar_t * RangeList, int Options);
```

Parameters

DocumentID	The ID of the document to copy the pages from
RangeList	The pages to extract, for example "10,15,18-20,25-35". Invalid characters in the string will be ignored.
Options	0 = Identical behaviour to the **CopyPageRanges** function. The page list is sorted and duplicate page numbers are ignored. 1 = Do not sort the page list and allow duplicate page numbers

Return values

0	The specified DocumentID was not valid or was the same as the selected document, or the RangeList was invalid
1	The pages were successfully copied from the specified document to the selected document

CreateBuffer

Version history

This function was introduced in Quick PDF Library version 7.11.

Description

Creates a buffer that can be used to send strings to Quick PDF Library DLL containing null characters.

Once the buffer has been created, use the **AddToBuffer** function to add data to the buffer. The data can be added to the buffer in one call, or chunks of data can be sent one at a time until the buffer is full.

When you are finished with the buffer, call the **ReleaseBuffer** function to release it.

Syntax

DLL

```
char * DPLCreateBuffer(int InstanceID, int BufferLength);
```

Parameters

BufferLength	The size in bytes of the buffer that must be created

Return values

0	The BufferLength value was less than 1, or the InstanceID was invalid
Non-zero	A PChar that can be passed as any string parameter to other functions

CreateLibrary

Miscellaneous functions

Version history

>This function was introduced in Quick PDF Library version 7.11.

Description

>Call this function to create an instance of Quick PDF Library in the DLL. The value returned is used as the InstanceID parameter of all the other functions.

>Call the **ReleaseLibrary** function to free the the instance when you are finished with it.

Syntax

DLL

```
int DPLCreateLibrary(int InstanceID);
```

Return values

0	An instance of Quick PDF Library could not be created
Non-zero	An InstanceID value that can be used with other functions

CreateNewObject

Version history

This function was introduced in Quick PDF Library version 7.26.

Description

Adds a new PDF object to the document. The contents of the object can be set using the **SetObjectFromString** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.CreateNewObject: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::CreateNewObject As Long
```

DLL

```
int DPLCreateNewObject(int InstanceID);
```

Return values

Non-zero	The object number of the newly created object

CreateTable

Version history

This function was introduced in Quick PDF Library version 7.14.

Description

Creates a table with the specified number of rows and columns. Use the other table functions to set up the table and then use **DrawTableRows** to draw the table onto the page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.CreateTable(RowCount,
  ColumnCount: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::CreateTable(RowCount As Long,
  ColumnCount As Long) As Long
```

DLL

```
int DPLCreateTable(int InstanceID, int RowCount, int ColumnCount);
```

Parameters

RowCount	The number of rows that the new table should have
ColumnCount	The number of columns that the new table should have.

Return values

0	The table could not be created. Row and column count must be greater or equal to 1.
Non-zero	A TableID that can be used with the other table functions.

DAAppendFile

Description

Appends any changes made to a document originally opened using the **DAOpenFile** function. This is a fast operation because only the changed objects must be added to the end of the original file. The file is closed after this operation and the file handle will no longer be valid.

This function will not work if the source file was opened in read only mode or if the document was loaded from a malformed file for example where whitespace was added to the start of the file. In these cases the **DASaveAsFile** function should be used instead.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAAppendFile(FileHandle: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAAppendFile(
  FileHandle As Long) As Long
```

DLL

```
int DPLDAAppendFile(int InstanceID, int FileHandle);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions

Return values

0	The specified FileHandle was not valid
1	The changes to the file were appended successfully
2	The file was opened in read only mode and the update cannot be written. Use **DASaveAsFile** instead.
3	The document was opened from a malformed file and an append operation is not possible. See the **DAShiftedHeader** function.

DACapturePage

Description

This function "captures" the specified page from a document originally opened with **DAOpenFile**. The captured page can then be drawn onto any other page using the **DADrawCapturedPage** function. This is useful for combining different pages or for placing more than one original page onto another page (imposition).

Once a page has been captured it is removed from the document. If you would like the page to remain in the document you must create a blank page and draw the captured page onto the blank page.

The "media box" for the page is used as the bounding rectangle for the capture page. The **DACapturePageEx** function can be used in cases where the "crop box" for the page should be used instead.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DACapturePage(FileHandle,
  PageRef: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DACapturePage(
  FileHandle As Long, PageRef As Long) As Long
```

DLL

```
int DPLDACapturePage(int InstanceID, int FileHandle, int PageRef);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
PageRef	A page reference returned by the **DAFindPage** or **DANewPage** functions

Return values

0	The specified FileHandle or PageRef were not valid
Non-zero	An ID that can be used with the **DADrawCapturedPage** function

DACapturePageEx

Description

Captures the specified page from a document originally opened with **DAOpenFile**. The captured page is hidden, but can then be drawn onto any other page using the **DADrawCapturedPage** function. The "media box" for the page is used as the bounding rectangle for the capture page. The **DACapturePageEx** function can be used in cases where the "crop box" for the page should be used instead.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DACapturePageEx(FileHandle, PageRef,
  Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DACapturePageEx(
  FileHandle As Long, PageRef As Long, Options As Long) As Long
```

DLL

```
int DPLDACapturePageEx(int InstanceID, int FileHandle, int PageRef,
  int Options);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
PageRef	A page reference returned by the **DAFindPage** or **DANewPage** functions
Options	0 = Use the page's media box for the bounding rectangle 1 = Use the page's crop box for the bounding rectangle if it has one, otherwise use the media box 2 = Use the page's bleed box for the bounding rectangle if it has one, otherwise use the crop box 3 = Use the page's trim box for the bounding rectangle if it has one, otherwise use the crop box 4 = Use the page's art box for the bounding rectangle if it has one, otherwise use the crop box

Return values

0	The specified FileHandle or PageRef were not valid, or the specified page was the only page in the document
Non-zero	An ID that can be used with the **DADrawCapturedPage** function

DACloseFile

Direct access functionality

Description

Closes a file that was originally opened using the **DAOpenFile** function. Any changes made to the file are lost. If you would like to keep your changes you must use either the **DASaveAsFile** function or the **DAAppendFile** function before closing the file.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DACloseFile(FileHandle: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DACloseFile(
  FileHandle As Long) As Long
```

DLL

```
int DPLDACloseFile(int InstanceID, int FileHandle);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions

Return values

0	The specified FileHandle was not valid, the file may already have been closed
1	The file was closed successfully

DADrawCapturedPage

Direct access functionality, Page layout

Description

Draws a page originally captured using the **DrawCapturedPage** function onto the specified page. The original page must have been captured from the same document (having the same FileHandle).

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DADrawCapturedPage(FileHandle, DACaptureID,
  DestPageRef: Integer; PntLeft, PntBottom, PntWidth,
  PntHeight: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DADrawCapturedPage(
  FileHandle As Long, DACaptureID As Long, DestPageRef As Long,
  PntLeft As Double, PntBottom As Double, PntWidth As Double,
  PntHeight As Double) As Long
```

DLL

```
int DPLDADrawCapturedPage(int InstanceID, int FileHandle, int DACaptureID,
  int DestPageRef, double PntLeft, double PntBottom,
  double PntWidth, double PntHeight);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
DACaptureID	A capture ID returned by the **DACapturePage** function
DestPageRef	A page reference returned by the **DAFindPage** or **DANewPage** functions
PntLeft	The horizontal co-ordinate of the left edge of the destination rectangle, measured in points from the left edge of the page
PntBottom	The vertical co-ordinate of the bottom edge of the destination rectangle, measured in points from the bottom edge of the page
PntWidth	The width of the destination rectangle, measured in points
PntHeight	The height of the destination rectangle, measured in points

Return values

0	The specified FileHandle, PageRef or DACaptureID were not valid
1	The captured page was drawn successfully

DADrawRotatedCapturedPage

Direct access functionality, Page layout

Version history

This function was introduced in Quick PDF Library version 7.23.

Description

Similar to the **DADrawCapturedPage** function but allows the captured page to be drawn at any angle.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DADrawRotatedCapturedPage(FileHandle,
  DACaptureID, DestPageRef: Integer; PntLeft, PntBottom, PntWidth,
  PntHeight, Angle: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DADrawRotatedCapturedPage(
  FileHandle As Long, DACaptureID As Long, DestPageRef As Long,
  PntLeft As Double, PntBottom As Double, PntWidth As Double,
  PntHeight As Double, Angle As Double) As Long
```

DLL

```
int DPLDADrawRotatedCapturedPage(int InstanceID, int FileHandle,
  int DACaptureID, int DestPageRef, double PntLeft,
  double PntBottom, double PntWidth, double PntHeight,
  double Angle);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
DACaptureID	A capture ID returned by the **DACapturePage** function
DestPageRef	A page reference returned by the **DAFindPage** or **DANewPage** functions
PntLeft	The horizontal co-ordinate of the left edge of the destination rectangle, measured in points from the left edge of the page
PntBottom	The vertical co-ordinate of the bottom edge of the destination rectangle, measured in points from the bottom edge of the page
PntWidth	The width of the destination rectangle, measured in points
PntHeight	The height of the destination rectangle, measured in points
Angle	The angle to rotate the captured page by, measured anti-clockwise in degrees from the baseline

Return values

0	The specified FileHandle, PageRef or DACaptureID were not valid
1	The captured page was drawn successfully

DAEmbedFileStreams

Document manipulation, Direct access functionality

Version history

This function was introduced in Quick PDF Library version 7.24.

Description

Converts any stream object where the data is stored in an external file into a regular embedded stream object.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAEmbedFileStreams(FileHandle: Integer;
  RootPath: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAEmbedFileStreams(
  FileHandle As Long, RootPath As String) As Long
```

DLL

```
int DPLDAEmbedFileStreams(int InstanceID, int FileHandle,
  wchar_t * RootPath);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
RootPath	The directory to use as the root for relative paths.

DAExtractPageText

Description

This function provides two different methods for extracting text from the selected page, and presents the results in a variety of formats.

The **DASetTextExtractionWordGap**, **DASetTextExtractionOptions** and **DASetTextExtractionArea** functions can be used to adjust the text extraction process.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAExtractPageText(FileHandle, PageRef,
  Options: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAExtractPageText(
  FileHandle As Long, PageRef As Long, Options As Long) As String
```

DLL

```
wchar_t * DPLDAExtractPageText(int InstanceID, int FileHandle,
  int PageRef, int Options);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
PageRef	A page reference returned by the **DAFindPage** or **DANewPage** functions
Options	Using the standard text extraction algorithm: 0 = Extract text in human readable format 1 = Deprecated 2 = Return a CSV string including font, color, size and position of each piece of text on the page Using the more accurate but slower text extraction algorithm: 3 = Return a CSV string for each piece of text on the page with the following format: Font Name, Text Color, Text Size, X1, Y1, X2, Y2, X3, Y3, X4, Y4, Text The co-ordinates are the four points bounding the text, measured using the units set with the **SetMeasurementUnits** function and the origin set with the **SetOrigin** function. Co-ordinate order is anti-clockwise with the bottom left corner first. 4 = Similar to option 3, but individual words are returned, making searching for words easier 5 = Similar to option 3 but character widths are output after each block of text 6 = Similar to option 4 but character widths are output after each line of text 7 = Extract text in human readable format with improved accuracy compared to option 0 8 = Similar output format as option 0 but using the more accurate algorithm. Returns unformatted lines.

DAExtractPageTextBlocks

Version history

> This function was introduced in Quick PDF Library version 9.13.

Description

> Similar to the **DAExtractPageText** function but the results are stored in a text block list rather than returned as a CSV string.
>
> Once the results are in the text block list, functions such as **DAGetTextBlockCount**, **DAGetTextBlockText** and **DAGetTextBlockColor** can be used to retrieve the properties of each block of text.

Syntax

> **Delphi**

```
function TDebenuPDFLibrary1115.DAExtractPageTextBlocks(FileHandle,
  PageRef, ExtractOptions: Integer): Integer;
```

> **ActiveX**

```
Function DebenuPDFLibrary1115.PDFLibrary::DAExtractPageTextBlocks(
  FileHandle As Long, PageRef As Long,
  ExtractOptions As Long) As Long
```

> **DLL**

```
int DPLDAExtractPageTextBlocks(int InstanceID, int FileHandle,
  int PageRef, int ExtractOptions);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
PageRef	A page reference returned by the **DAFindPage** or **DANewPage** functions
ExtractOptions	3 = Normal extraction 4 = Split words

Return values

0	Text could not be extracted from the page
Non-zero	A TextBlockListID value

DAFindPage

Description

Use this function to obtain a page reference for use with other Direct Access functions. This page reference will remain constant even if other pages are added to or removed from the document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAFindPage(FileHandle,
  Page: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAFindPage(FileHandle As Long,
  Page As Long) As Long
```

DLL

```
int DPLDAFindPage(int InstanceID, int FileHandle, int Page);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
Page	The page number. The first page in the document has a page number of 1.

Return values

0	The specified FileHandle was not valid or the Page parameter was out of range
Non-zero	An ID that can be used as the PageRef parameter for any of the direct access functions

DAGetAnnotationCount

Direct access functionality

Version history

This function was introduced in Quick PDF Library version 7.21.

Description

Returns the number of annotations on the specified page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAGetAnnotationCount(FileHandle,
  PageRef: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAGetAnnotationCount(
  FileHandle As Long, PageRef As Long) As Long
```

DLL

```
int DPLDAGetAnnotationCount(int InstanceID, int FileHandle, int PageRef);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
PageRef	A page reference returned by the **DAFindPage** or **DANewPage** functions

DAGetFormFieldCount

Form fields, Direct access functionality

Version history

This function was introduced in Quick PDF Library version 7.16.

Description

Returns the number of form fields in the document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAGetFormFieldCount(
  FileHandle: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAGetFormFieldCount(
  FileHandle As Long) As Long
```

DLL

```
int DPLDAGetFormFieldCount(int InstanceID, int FileHandle);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions

DAGetFormFieldTitle

Version history

This function was introduced in Quick PDF Library version 7.16.

Description

Returns the title of the specified form field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAGetFormFieldTitle(FileHandle,
  FieldIndex: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAGetFormFieldTitle(
  FileHandle As Long, FieldIndex As Long) As String
```

DLL

```
wchar_t * DPLDAGetFormFieldTitle(int InstanceID, int FileHandle,
  int FieldIndex);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
FieldIndex	The index of the form field to work with. The first form field has an index of 1.

DAGetFormFieldValue

Version history

This function was introduced in Quick PDF Library version 7.16.

Description

Returns the value of the specified form field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAGetFormFieldValue(FileHandle,
  FieldIndex: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAGetFormFieldValue(
  FileHandle As Long, FieldIndex As Long) As String
```

DLL

```
wchar_t * DPLDAGetFormFieldValue(int InstanceID, int FileHandle,
  int FieldIndex);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
FieldIndex	The index of the form field to work with. The first form field has an index of 1.

DAGetImageDataToString

Version history

This function was renamed in Quick PDF Library version 7.11.
The function name in earlier versions was DAGetImageDataAsString.

Description

Returns the image data of an image in an image list.

The format of the data depends on the type of the image. The **DAGetImageIntProperty** function can be used to determine the image type.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAGetImageDataToString(FileHandle,
  ImageListID, ImageIndex: Integer): AnsiString;
```

DLL

```
char * DPLDAGetImageDataToString(int InstanceID, int FileHandle,
  int ImageListID, int ImageIndex);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
ImageListID	A value returned by the **DAGetPageImageList** function
ImageIndex	The index of the image. The first image in the list has an index of 1. Use the **DAGetImageListCount** function to determine the number of images in the list.

DAGetImageDataToVariant

Image handling, Direct access functionality

Version history

This function was renamed in Quick PDF Library version 7.11.
The function name in earlier versions was DAGetImageDataAsVariant.

Description

Returns the image data of an image in an image list as a byte array variant.

The format of the data depends on the type of the image. The **DAGetImageIntProperty** function can be used to determine the image type.

Syntax

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAGetImageDataToVariant(
  FileHandle As Long, ImageListID As Long,
  ImageIndex As Long) As Variant
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
ImageListID	A value returned by the **DAGetPageImageList** function
ImageIndex	The index of the image. The first image in the list has an index of 1. Use the **DAGetImageListCount** function to determine the number of images in the list.

DAGetImageDblProperty

Image handling, Direct access functionality

Description

Returns certain properties of an image in an image list.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAGetImageDblProperty(FileHandle,
  ImageListID, ImageIndex, PropertyID: Integer): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAGetImageDblProperty(
  FileHandle As Long, ImageListID As Long, ImageIndex As Long,
  PropertyID As Long) As Double
```

DLL

```
double DPLDAGetImageDblProperty(int InstanceID, int FileHandle,
  int ImageListID, int ImageIndex, int PropertyID);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
ImageListID	A value returned by the **DAGetPageImageList** function
ImageIndex	The index of the image. The first image in the list has an index of 1. Use the **DAGetImageListCount** function to determine the number of images in the list.
PropertyID	501 = Horizontal co-ordinate of top-left corner 502 = Vertical co-ordinate of top-left corner 503 = Horizontal co-ordinate of top-right corner 504 = Vertical co-ordinate of top-right corner 505 = Horizontal co-ordinate of bottom-right corner 506 = Vertical co-ordinate of bottom-right corner 507 = Horizontal co-ordinate of bottom-left corner 508 = Vertical co-ordinate of bottom-left corner

DAGetImageIntProperty

Description

Returns certain properties of an image in an image list.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAGetImageIntProperty(FileHandle,
  ImageListID, ImageIndex, PropertyID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAGetImageIntProperty(
  FileHandle As Long, ImageListID As Long, ImageIndex As Long,
  PropertyID As Long) As Long
```

DLL

```
int DPLDAGetImageIntProperty(int InstanceID, int FileHandle,
  int ImageListID, int ImageIndex, int PropertyID);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
ImageListID	A value returned by the **DAGetPageImageList** function
ImageIndex	The index of the image. The first image in the list has an index of 1. Use the **DAGetImageListCount** function to determine the number of images in the list.
PropertyID	400 = Image type (see **ImageType**) for values 401 = Width in pixels 402 = Height in pixels 403 = Bits per pixel 404 = Color space type 405 = Image ID (will be 0 if it is an Inline image)

DAGetImageListCount

Description

Returns the number of images in an image list.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAGetImageListCount(FileHandle,
  ImageListID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAGetImageListCount(
  FileHandle As Long, ImageListID As Long) As Long
```

DLL

```
int DPLDAGetImageListCount(int InstanceID, int FileHandle,
  int ImageListID);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
ImageListID	A value returned by the **DAGetPageImageList** function

DAGetInformation

Description

Retrieves information from the document information section. This could be standard information such as Author and Subject, or custom information.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAGetInformation(FileHandle: Integer;
  Key: WideString): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAGetInformation(
  FileHandle As Long, Key As String) As String
```

DLL

```
wchar_t * DPLDAGetInformation(int InstanceID, int FileHandle,
  wchar_t * Key);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
Key	For standard information use "Author", "Title", "Subject", "Keywords", "Creator", or "Producer". For custom information any other string can be used.

DAGetObjectCount

Version history

This function was introduced in Quick PDF Library version 7.16.

Description

Returns the number of raw PDF objects in the document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAGetObjectCount(
  FileHandle: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAGetObjectCount(
  FileHandle As Long) As Long
```

DLL

```
int DPLDAGetObjectCount(int InstanceID, int FileHandle);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions

DAGetObjectToString

Version history

This function was renamed in Quick PDF Library version 8.11.
The function name in earlier versions was DAGetObjectSource.

Description

Returns the raw PDF object data for the specified object number. This is for advanced use only.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAGetObjectToString(FileHandle,
  ObjectNumber: Integer): AnsiString;
```

DLL

```
char * DPLDAGetObjectToString(int InstanceID, int FileHandle,
  int ObjectNumber);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
ObjectNumber	The number of the object to retrieve. The first object is numbered 1 and the last object has an object number equal to the result of the **GetObjectCount** function.

DAGetObjectToVariant

Version history

This function was renamed in Quick PDF Library version 8.11.
The function name in earlier versions was DAGetObjectSource.

Description

Returns the raw PDF object data for the specified object number as a variant byte array. This is for advanced use only.

Syntax

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAGetObjectToVariant(
  FileHandle As Long, ObjectNumber As Long) As Variant
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
ObjectNumber	The number of the object to retrieve. The first object is numbered 1 and the last object has an object number equal to the result of the **GetObjectCount** function.

DAGetPageBox

Version history

This function was introduced in Quick PDF Library version 7.23.

Description

Returns a dimension of the specified page boundary rectangle.

Returned values are points measured from the bottom left corner of the page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAGetPageBox(FileHandle, PageRef, BoxIndex,
  Dimension: Integer): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAGetPageBox(FileHandle As Long,
  PageRef As Long, BoxIndex As Long, Dimension As Long) As Double
```

DLL

```
double DPLDAGetPageBox(int InstanceID, int FileHandle, int PageRef,
  int BoxIndex, int Dimension);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
PageRef	A page reference returned by the **DAFindPage** or **DANewPage** functions
BoxIndex	1 = MediaBox 2 = CropBox 3 = BleedBox 4 = TrimBox 5 = ArtBox
Dimension	0 = Left 1 = Top 2 = Width 3 = Height 4 = Right 5 = Bottom

DAGetPageContentToString

Direct access functionality, Page properties

Version history

This function was renamed in Quick PDF Library version 8.11.
The function name in earlier versions was DAGetPageContent.

Description

Retrieves the graphics commands and operators that make up the specified page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAGetPageContentToString(FileHandle,
  PageRef: Integer): AnsiString;
```

DLL

```
char * DPLDAGetPageContentToString(int InstanceID, int FileHandle,
  int PageRef);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
PageRef	A page reference returned by the **DAFindPage** or **DANewPage** functions

DAGetPageContentToVariant

Direct access functionality, Page properties

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Retrieves the graphics commands and operators that make up the specified page as a variant byte array.

Syntax

```
Function DebenuPDFLibrary1115.PDFLibrary::DAGetPageContentToVariant(
    FileHandle As Long, PageRef As Long) As Variant
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
PageRef	A page reference returned by the **DAFindPage** or **DANewPage** functions

DAGetPageCount

Description

Returns the number of pages in a document opened with the **DAOpenFile** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAGetPageCount(
  FileHandle: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAGetPageCount(
  FileHandle As Long) As Long
```

DLL

```
int DPLDAGetPageCount(int InstanceID, int FileHandle);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions

Return values

0	The specified FileHandle was not valid
Non-zero	The number of pages in the document

DAGetPageHeight

Description

Returns the height of the specified page in a document opened with the **DAOpenFile** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAGetPageHeight(FileHandle,
  PageRef: Integer): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAGetPageHeight(
  FileHandle As Long, PageRef As Long) As Double
```

DLL

```
double DPLDAGetPageHeight(int InstanceID, int FileHandle, int PageRef);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
PageRef	A page reference returned by the **DAFindPage** or **DANewPage** functions

DAGetPageImageList

Description

This function finds all the images on the selected page and returns an ImageListID that can be used with the **DAGetImageListCount**, **DAGetImageListItemIntProperty**, **DAGetImageListItemDblProperty**, **DAGetImageListItemDataToString**, **DAGetImageListItemDataToVariant** and **DASaveImageListItemDataToFile** functions.

As of version 10.13 will include Inline images but the ImageID will be 0 for any inline image which means that any inline images cannot used with ReplaceImage or ClearImage functions.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAGetPageImageList(FileHandle,
  PageRef: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAGetPageImageList(
  FileHandle As Long, PageRef As Long) As Long
```

DLL

```
int DPLDAGetPageImageList(int InstanceID, int FileHandle, int PageRef);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
PageRef	A page reference returned by the **DAFindPage** function

Return values

0	The FileHandle or PageRef parameters were invalid
Non-zero	An ImageListID value that can be used with the other direct access image list functions

DAGetPageLayout

Version history

This function was introduced in Quick PDF Library version 11.15.

Description

Returns the page layout of the specified document.

Syntax

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions

Return values

0	Single page
1	One column
2	Two columns, odd-numbered pages on left
3	Two columns, odd-numbered pages on right
4	Two pages, odd-numbered pages on left
5	Two pages, odd-numbered pages on right

DAGetPageMode

Version history

This function was introduced in Quick PDF Library version 11.15.

Description

Returns the page mode of the specified document.

Syntax

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions

Return values

0	Normal view
1	Show the outlines pane
2	Show the thumbnails pane
3	Show the document in full screen mode
4	Optional content group panel visible
5	Attachments panel visible

DAGetPageWidth

Description

Returns the width of the specified page in a document opened with the **DAOpenFile** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAGetPageWidth(FileHandle,
  PageRef: Integer): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAGetPageWidth(
  FileHandle As Long, PageRef As Long) As Double
```

DLL

```
double DPLDAGetPageWidth(int InstanceID, int FileHandle, int PageRef);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
PageRef	A page reference returned by the **DAFindPage** or **DANewPage** functions

DAGetTextBlockAsString

Text, Extraction, Direct access functionality

Version history

This function was introduced in Quick PDF Library version 11.12.

Description

Returns all the text block entries for a single text block as a formatted string delimited by CR/LF

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAGetTextBlockAsString(TextBlockListID,
  Index: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAGetTextBlockAsString(
  TextBlockListID As Long, Index As Long) As String
```

DLL

```
wchar_t * DPLDAGetTextBlockAsString(int InstanceID, int TextBlockListID,
  int Index);
```

Parameters

TextBlockListID	A value returned by the **ExtractPageTextBlocks** function
Index	The index of the text block. The first text block in the list has an index of 1.

Return values

TextBlockAsString	A formatted string of all available text block fields where each line is separate by a CR/LF. Here is a sample output string

CNT:4
FNT:Arial
SIZ:12
CLR:#000000
TX1:20
TY1:769.516
TX2:48.02
TY2:769.516
TX3:48.02
TY3:780.616
TX4:20
TY4:780.616
WID:8.004,6.672,6.672,6.672
TXT:Page

where CNT = char count, FNT = fontname, SIZ = Fontsize, CLR = color, TXx = X value for bounds point x, TYy = Y value for bounds y, WID = comma separated character widths, TXT = extracted text.

DAGetTextBlockBound

Version history

This function was introduced in Quick PDF Library version 9.13.

Description

Returns one of the bounds of the specified text block.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAGetTextBlockBound(TextBlockListID, Index,
  BoundIndex: Integer): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAGetTextBlockBound(
  TextBlockListID As Long, Index As Long,
  BoundIndex As Long) As Double
```

DLL

```
double DPLDAGetTextBlockBound(int InstanceID, int TextBlockListID,
  int Index, int BoundIndex);
```

Parameters

TextBlockListID	A value returned by the **DAExtractPageTextBlocks** or **ExtractFilePageTextBlocks** functions
Index	The index of the text block. The first text block in the list has an index of 1.
BoundIndex	1 = Bottom left horizontal coordinate 2 = Bottom left vertical coordinate 3 = Bottom right horizontal coordinate 4 = Bottom right vertical coordinate 5 = Top right horizontal coordinate 6 = Top right vertical coordinate 7 = Top left horizontal coordinate 8 = Top left vertical coordinate

DAGetTextBlockCharWidth

Version history

This function was introduced in Quick PDF Library version 9.13.

Description

Returns the width of a particular character within the specified text block.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAGetTextBlockCharWidth(TextBlockListID,
  Index, CharIndex: Integer): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAGetTextBlockCharWidth(
  TextBlockListID As Long, Index As Long,
  CharIndex As Long) As Double
```

DLL

```
double DPLDAGetTextBlockCharWidth(int InstanceID, int TextBlockListID,
  int Index, int CharIndex);
```

Parameters

TextBlockListID	A value returned by the **DAExtractPageTextBlocks** or **ExtractFilePageTextBlocks** functions
Index	The index of the text block. The first text block in the list has an index of 1.
CharIndex	The index of the character to retrieve the width of. The first character has an index of 1.

DAGetTextBlockColor

Version history

This function was introduced in Quick PDF Library version 9.13.

Description

Returns one component of the color of the text in the specified text block.

The color component value is returned as a value between 0 and 1.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAGetTextBlockColor(TextBlockListID, Index,
   ColorComponent: Integer): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAGetTextBlockColor(
   TextBlockListID As Long, Index As Long,
   ColorComponent As Long) As Double
```

DLL

```
double DPLDAGetTextBlockColor(int InstanceID, int TextBlockListID,
   int Index, int ColorComponent);
```

Parameters

TextBlockListID	A value returned by the **DAExtractPageTextBlocks** or **ExtractFilePageTextBlocks** functions
Index	The index of the text block. The first text block in the list has an index of 1.
ColorComponent	For RGB: 1 = Red 2 = Green 3 = Blue For CMYK: 1 = Cyan 2 = Magenta 3 = Yellow 4 = Black

DAGetTextBlockColorType

Version history

This function was introduced in Quick PDF Library version 9.13.

Description

Returns the type of color of the text in the specified text block.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAGetTextBlockColorType(TextBlockListID,
  Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAGetTextBlockColorType(
  TextBlockListID As Long, Index As Long) As Long
```

DLL

```
int DPLDAGetTextBlockColorType(int InstanceID, int TextBlockListID,
  int Index);
```

Parameters

TextBlockListID	A value returned by the **DAExtractPageTextBlocks** or **ExtractFilePageTextBlocks** functions
Index	The index of the text block. The first text block in the list has an index of 1.

Return values

3	RGB
4	CMYK

DAGetTextBlockCount

Version history

This function was introduced in Quick PDF Library version 9.13.

Description

Returns the number of text blocks in the specified text block list.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAGetTextBlockCount(
  TextBlockListID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAGetTextBlockCount(
  TextBlockListID As Long) As Long
```

DLL

```
int DPLDAGetTextBlockCount(int InstanceID, int TextBlockListID);
```

Parameters

TextBlockListID	A value returned by the **DAExtractPageTextBlocks** or **ExtractFilePageTextBlocks** functions

DAGetTextBlockFontName

Version history

This function was introduced in Quick PDF Library version 9.13.

Description

Returns the font name of the text in the specified text block.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAGetTextBlockFontName(TextBlockListID,
  Index: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAGetTextBlockFontName(
  TextBlockListID As Long, Index As Long) As String
```

DLL

```
wchar_t * DPLDAGetTextBlockFontName(int InstanceID, int TextBlockListID,
  int Index);
```

Parameters

TextBlockListID	A value returned by the **DAExtractPageTextBlocks** or **ExtractFilePageTextBlocks** functions
Index	The index of the text block. The first text block in the list has an index of 1.

DAGetTextBlockFontSize

Version history

This function was introduced in Quick PDF Library version 9.13.

Description

Returns the font size of the text in the specified text block.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAGetTextBlockFontSize(TextBlockListID,
  Index: Integer): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAGetTextBlockFontSize(
  TextBlockListID As Long, Index As Long) As Double
```

DLL

```
double DPLDAGetTextBlockFontSize(int InstanceID, int TextBlockListID,
  int Index);
```

Parameters

TextBlockListID	A value returned by the **DAExtractPageTextBlocks** or **ExtractFilePageTextBlocks** functions
Index	The index of the text block. The first text block in the list has an index of 1.

DAGetTextBlockText

Version history

This function was introduced in Quick PDF Library version 9.13.

Description

Returns the text in the specified text block.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAGetTextBlockText(TextBlockListID,
  Index: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAGetTextBlockText(
  TextBlockListID As Long, Index As Long) As String
```

DLL

```
wchar_t * DPLDAGetTextBlockText(int InstanceID, int TextBlockListID,
  int Index);
```

Parameters

TextBlockListID	A value returned by the **DAExtractPageTextBlocks** or **ExtractFilePageTextBlocks** functions
Index	The index of the text block. The first text block in the list has an index of 1.

DAHasPageBox

Direct access functionality, Page properties

Version history

This function was introduced in Quick PDF Library version 7.23.

Description

Determines if a page has a particular page boundary rectangle.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAHasPageBox(FileHandle, PageRef,
  BoxIndex: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAHasPageBox(FileHandle As Long,
  PageRef As Long, BoxIndex As Long) As Long
```

DLL

```
int DPLDAHasPageBox(int InstanceID, int FileHandle, int PageRef,
  int BoxIndex);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
PageRef	A page reference returned by the **DAFindPage** or **DANewPage** functions
BoxIndex	1 = MediaBox 2 = CropBox 3 = BleedBox 4 = TrimBox 5 = ArtBox

Return values

0	The page does not have the specified page boundary rectangle
1	The page has the specified page boundary rectangle

DAHidePage

Description

Hides the specified page from a document originally opened with **DAOpenFile**. The content of the page is still in the document, but the page will not be visible.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAHidePage(FileHandle,
  PageRef: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAHidePage(FileHandle As Long,
  PageRef As Long) As Long
```

DLL

```
int DPLDAHidePage(int InstanceID, int FileHandle, int PageRef);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
PageRef	A page reference returned by the **DAFindPage** or **DANewPage** functions

Return values

0	The specified FileHandle or PageRef were not valid
1	The page was hidden successfully

DAMovePage

Direct access functionality, Page manipulation

Description

Moves a page to a new location in the document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAMovePage(FileHandle, PageRef,
  TargetPageRef, Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAMovePage(FileHandle As Long,
  PageRef As Long, TargetPageRef As Long,
  Options As Long) As Long
```

DLL

```
int DPLDAMovePage(int InstanceID, int FileHandle, int PageRef,
  int TargetPageRef, int Options);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
PageRef	A page reference returned by the **DAFindPage** or **DANewPage** functions. This is the page that will be moved.
TargetPageRef	A page reference returned by the **DAFindPage** or **DANewPage** functions. The page will be moved before or after this page.
Options	0 = Move before target page 1 = Move after target page

Return values

0	The page could not be moved. Check that the FileHandle, PageRef and TargetPageRef values are correct.
1	The page was moved successfully

DANewPage

Direct access functionality, Page manipulation

Description

Adds a new blank page to the end of the document. The page will have a standard size of 612x792 points.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DANewPage(FileHandle: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DANewPage(
  FileHandle As Long) As Long
```

DLL

```
int DPLDANewPage(int InstanceID, int FileHandle);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions

Return values

0	The specified FileHandle was not valid
Non-zero	An ID that can be used as the PageRef parameter for any of the direct access functions

DANewPages

Description

Adds a number of new pages to the end of the document. All new pages have a standard size of 612x792 points.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DANewPages(FileHandle,
  PageCount: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DANewPages(FileHandle As Long,
  PageCount As Long) As Long
```

DLL

```
int DPLDANewPages(int InstanceID, int FileHandle, int PageCount);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
PageCount	The number of pages to add to the document

Return values

0	The specified FileHandle was not valid
Non-zero	The total number of pages in the document after the new pages were added

DANormalizePage

Version history

This function was introduced in Quick PDF Library version 11.11.

Description

Moves and/or rotates the contents of the page so that subsequent drawing operations are at the expected position on the page. All the page boundary boxes are adjusted to the physical size of the page and the page's rotation attribute is reset to zero.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DANormalizePage(FileHandle, PageRef,
  NormalizeOptions: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DANormalizePage(
  FileHandle As Long, PageRef As Long,
  NormalizeOptions As Long) As Long
```

DLL

```
int DPLDANormalizePage(int InstanceID, int FileHandle, int PageRef,
  int NormalizeOptions);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
PageRef	A page reference returned by the **DAFindPage** or **DANewPage** functions
NormalizeOptions	0 = Standard normalization

DAOpenFile

Description

Opens a file in direct access mode. This allows large files to be processed. The file will not be accessible by other processes until the file is closed using the one of the following functions: **DACloseFile**, **DAAppendFile** or **DASaveAsFile**. Read only files can be opened and all other direct access functions will work but **DAAppendFile** will not work as the file cannot be written.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAOpenFile(InputFileName,
  Password: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAOpenFile(
  InputFileName As String, Password As String) As Long
```

DLL

```
int DPLDAOpenFile(int InstanceID, wchar_t * InputFileName,
  wchar_t * Password);
```

Parameters

InputFileName	The path and name of the document to open in direct access mode.
Password	The password to use when opening the document. This can be the owner or user password. If the user password is used certain functionality may be restricted depending on the permissions of the document.

Return values

0	The file could not be opened. Use the **LastErrorCode** function to determine the cause of the failure.
Non-zero	A FileHandle that can be used with the other Direct Access functions

DAOpenFileReadOnly

Document management, Direct access functionality

Description

Opens a file in direct access mode. This allows large files to be processed. The file is opened with read only access so other processes will also be able to open the file in read only mode. **DASaveAsFile** should be used to save any changes to a new file as **DAAppendFile** cannot update read only files.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAOpenFileReadOnly(InputFileName,
  Password: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAOpenFileReadOnly(
  InputFileName As String, Password As String) As Long
```

DLL

```
int DPLDAOpenFileReadOnly(int InstanceID, wchar_t * InputFileName,
  wchar_t * Password);
```

Parameters

InputFileName	The path and name of the document to open in direct access mode with read only access.
Password	The password to use when opening the document. This can be the owner or user password. If the user password is used certain functionality may be restricted depending on the permissions of the document.

Return values

0	The file could not be opened. Use the **LastErrorCode** function to determine the cause of the failure.
Non-zero	A FileHandle that can be used with the other Direct Access functions

DAOpenFromStream

Version history

This function was introduced in Quick PDF Library version 7.21.

Description

Opens a PDF stored inside a Delphi TStream object in direct access mode.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAOpenFromStream(InStream: TStream;
  Password: WideString): Integer;
```

Parameters

InStream	The TStream object containing the PDF document data
Password	The password to use when opening the document. This can be the owner or user password. If the user password is used certain functionality may be restricted depending on the permissions of the document.

Return values

0	The file could not be opened from the stream
Non-zero	A FileHandle that can be used with the other Direct Access functions

DAPageRotation

Description

Returns the rotation of the specified page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAPageRotation(FileHandle,
  PageRef: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAPageRotation(
  FileHandle As Long, PageRef As Long) As Long
```

DLL

```
int DPLDAPageRotation(int InstanceID, int FileHandle, int PageRef);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
PageRef	A page reference returned by the **DAFindPage** or **DANewPage** functions

DAReleaseImageList

Image handling, Direct access functionality, Page properties

Version history

This function was introduced in Quick PDF Library version 8.15.

Description

Releases the specified image list including all the image data extracted from the images in the list. Releasing the image list does not affect the original images.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAReleaseImageList(FileHandle,
  ImageListID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAReleaseImageList(
  FileHandle As Long, ImageListID As Long) As Long
```

DLL

```
int DPLDAReleaseImageList(int InstanceID, int FileHandle, int ImageListID);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
ImageListID	A value returned by the **DAGetPageImageList** function

Return values

0	The image list could not be released. The ImageListID parameter might be invalid or does not refer to an image list within the specified document.
1	The image list was released successfully.

DAReleaseTextBlocks

Direct access functionality

Version history

This function was introduced in Quick PDF Library version 9.13.

Description

Releases the memory used by a text block list.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAReleaseTextBlocks(
  TextBlockListID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAReleaseTextBlocks(
  TextBlockListID As Long) As Long
```

DLL

```
int DPLDAReleaseTextBlocks(int InstanceID, int TextBlockListID);
```

Parameters

TextBlockListID	A value returned by the **DAExtractPageTextBlocks** or **ExtractFilePageTextBlocks** functions

DARemoveUsageRights

Document manipulation, Direct access functionality

Version history

This function was introduced in Quick PDF Library version 7.25.

Description

Removes any usage rights from the document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DARemoveUsageRights(
  FileHandle: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DARemoveUsageRights(
  FileHandle As Long) As Long
```

DLL

```
int DPLDARemoveUsageRights(int InstanceID, int FileHandle);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions

Return values

0	The document did not have any usage rights
1	Success

DARenderPageToDC

Direct access functionality, Rendering and printing

Version history

This function was introduced in Quick PDF Library version 7.12.

Description

Renders the specified page from the specified document directly onto a graphics surface.

On Windows the target surface is a Device Context handle (DC).

By default rendering uses the GDI+ system which is available by default in Windows XP and later.

It is also possible to render using Cairo, use the **SetCairoFileName** and **SelectRenderer** functions.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DARenderPageToDC(FileHandle,
  PageRef: Integer; DPI: Double; DC: HDC): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DARenderPageToDC(
  FileHandle As Long, PageRef As Long, DPI As Double,
  DC As Long) As Long
```

DLL

```
int DPLDARenderPageToDC(int InstanceID, int FileHandle, int PageRef,
  double DPI, HDC DC);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
PageRef	A page reference returned by the **DAFindPage** or **DANewPage** functions
DPI	The DPI to use when rendering the page
DC	The device context handle

Return values

0	The page could not be rendered
1	The page was rendered successfully

DARenderPageToFile

Description

Renders the specified page from the specified document to an image and saves the image data as a file on disk.

By default rendering uses the GDI+ system which is available by default in Windows XP and later.

Option 10, TIFF (G4) output, is only available on Windows Vista and Windows Server 2008 and later.

It is also possible to render using Cairo, use the **SetCairoFileName** and **SelectRenderer** functions.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DARenderPageToFile(FileHandle, PageRef,
  Options: Integer; DPI: Double; FileName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DARenderPageToFile(
  FileHandle As Long, PageRef As Long, Options As Long,
  DPI As Double, FileName As String) As Long
```

DLL

```
int DPLDARenderPageToFile(int InstanceID, int FileHandle, int PageRef,
  int Options, double DPI, wchar_t * FileName);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
PageRef	A page reference returned by the **DAFindPage** or **DANewPage** functions
Options	0 = BMP output 1 = JPEG output 2 = WMF output 3 = EMF output 4 = EPS output 5 = PNG output 6 = GIF output 7 = TIFF output 8 = EMF+ output 9 = HTML5 output 10 = G4 TIFF output
DPI	The DPI to use when rendering the page. Values over 300 will cause excessive memory usage.
FileName	The path and file name of the file to create to store the rendered page image data in.

Return values

0	The page could not be rendered. Check the value of the FileHandle and PageRef parameters.
1	The page was rendered correctly and the image file was saved to disk
2	The file could not be written to disk

DARenderPageToStream

Description

This function is only available in the Delphi edition.

It renders the specified page from the specified document to an image and returns the image data in the supplied TStream.

By default rendering uses the GDI+ system which is available by default in Windows XP and later.

Option 10, TIFF (G4) output, is only available on Windows Vista and Windows Server 2008 and later.

It is also possible to render using Cairo, use the **SetCairoFileName** and **SelectRenderer** functions.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DARenderPageToStream(FileHandle, PageRef,
  Options: Integer; DPI: Double; Target: TStream): Integer;
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
PageRef	A page reference returned by the **DAFindPage** or **DANewPage** functions
Options	0 = BMP output 1 = JPEG output 2 = WMF output 3 = EMF output 4 = EPS output 5 = PNG output 6 = GIF output 7 = TIFF output 8 = EMF+ output 9 = HTML5 output 10 = G4 TIFF output
DPI	The DPI to use when rendering the page. Values over 300 will cause excessive memory usage.
Target	The stream to place the rendered page into

Return values

0	The page could not be rendered. Check that the FileHandle and PageRef parameters contain valid values.
1	The page was rendered and the image data was put into the stream

DARenderPageToString

Description

It renders the specified page from the specified document to an image and returns the image data as a string.

By default rendering uses the GDI+ system which is available by default in Windows XP and later.

Option 10, TIFF (G4) output, is only available on Windows Vista and Windows Server 2008 and later.

It is also possible to render using Cairo, use the **SetCairoFileName** and **SelectRenderer** functions.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DARenderPageToString(FileHandle, PageRef,
  Options: Integer; DPI: Double): AnsiString;
```

DLL

```
char * DPLDARenderPageToString(int InstanceID, int FileHandle,
  int PageRef, int Options, double DPI);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
PageRef	A page reference returned by the **DAFindPage** or **DANewPage** functions
Options	0 = BMP output 1 = JPEG output 2 = WMF output 3 = EMF output 4 = EPS output 5 = PNG output 6 = GIF output 7 = TIFF output 8 = EMF+ output 9 = HTML5 output 10 = G4 TIFF output
DPI	The DPI to use when rendering the page. Values over 300 will cause excessive memory usage.

DARenderPageToVariant

Version history

This function was introduced in Quick PDF Library version 7.11.

Description

Renders the specified page from the specified document to an image and returns the image data as a byte array variant.

By default rendering uses the GDI+ system which is available by default in Windows XP and later.

Option 10, TIFF (G4) output, is only available on Windows Vista and Windows Server 2008 and later.

It is also possible to render using Cairo, use the **SetCairoFileName** and **SelectRenderer** functions.

Syntax

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DARenderPageToVariant(
  FileHandle As Long, PageRef As Long, Options As Long,
  DPI As Double) As Variant
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
PageRef	A page reference returned by the **DAFindPage** or **DANewPage** functions
Options	0 = BMP output 1 = JPEG output 2 = WMF output 3 = EMF output 4 = EPS output 5 = PNG output 6 = GIF output 7 = TIFF output 8 = EMF+ output 9 = HTML5 output 10 = G4 TIFF output
DPI	The DPI to use when rendering the page. Values over 300 will cause excessive memory usage.

DARotatePage

Description

Sets the rotation of the selected page. The rotation is only applicable to the viewed page, the co-ordinate system rotates with the page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DARotatePage(FileHandle, PageRef, Angle,
  Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DARotatePage(FileHandle As Long,
  PageRef As Long, Angle As Long, Options As Long) As Long
```

DLL

```
int DPLDARotatePage(int InstanceID, int FileHandle, int PageRef,
  int Angle, int Options);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
PageRef	A page reference returned by the **DAFindPage** or **DANewPage** functions
Angle	The clockwise angle in degrees to rotate the page by, must be a multiple of 90
Options	Reserved for future use. Must be set to 0.

Return values

0	The page rotation could not be set. Check that the FileHandle and PageRef parameters are correct, and ensure that the Angle parameter is a multiple of 90.
1	The page rotation was set successfully

DASaveAsFile

Document management, Direct access functionality

Description

Rewrites the entire file, including all changes, to a new file. This operation may take some time with large files or files with many objects. The original file is closed after this operation and the file handle will no longer be valid. The original file cannot be overwritten. Use **DAAppendFile** if you want to append changes to original file.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DASaveAsFile(FileHandle: Integer;
  OutputFileName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DASaveAsFile(FileHandle As Long,
  OutputFileName As String) As Long
```

DLL

```
int DPLDASaveAsFile(int InstanceID, int FileHandle,
  wchar_t * OutputFileName);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
OutputFileName	The path and name of the new document to create.

Return values

0	The new file could not be created
1	The document was saved to the new file successfully

DASaveCopyToStream

Version history

This function was introduced in Quick PDF Library version 10.11.

Description

Similar to **DASaveToStream** but the input file is not closed.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DASaveCopyToStream(FileHandle: Integer;
  OutStream: TStream): Integer;
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
OutStream	The Delphi TStream object to save the file into

Return values

0	The file could not be saved to the stream
1	The file was successfully saved to the stream

DASaveImageDataToFile

Image handling, Direct access functionality

Description

Saves an image in an image list to a file on disk. The type of image file depends on the type of the image. The **DAGetImageIntProperty** function can be used to determine the image type.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DASaveImageDataToFile(FileHandle,
  ImageListID, ImageIndex: Integer; ImageFileName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DASaveImageDataToFile(
  FileHandle As Long, ImageListID As Long, ImageIndex As Long,
  ImageFileName As String) As Long
```

DLL

```
int DPLDASaveImageDataToFile(int InstanceID, int FileHandle,
  int ImageListID, int ImageIndex, wchar_t * ImageFileName);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
ImageListID	A value returned by the **DAGetPageImageList** function
ImageIndex	The index of the image. The first image in the list has an index of 1. Use the **DAGetImageListCount** function to determine the number of images in the list.
ImageFileName	The path and file name of the file to create to store the image data in.

DASaveToStream

Version history

This function was introduced in Quick PDF Library version 7.16.

Description

Saves the file to a TStream.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DASaveToStream(FileHandle: Integer;
  OutStream: TStream): Integer;
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
OutStream	The Delphi TStream object to save the file into

Return values

0	The file could not be saved to the stream
1	The file was successfully saved to the stream

DASetInformation

Description

Sets values in the document information section. This could be standard information such as Author and Subject, or custom information.

For CreationDate and ModDate (modification date), the format of the date should be: D:YYYYMMDDHHmmSSOHH'mm'

where
YYYY shall be the year
MM shall be the month (01-12)
DD shall be the day (01-31)
HH shall be the hour (00-23)
mm shall be the minute (00-59)
SS shall be the second (00-59)
O shall be the relationship of local time to Universal Time (UT) using a +, - or Z character
HH followed by APOSTROPHE (U+0027) (') shall be the absolute value of the offset from UT in hours (00-23)
mm followed by an optional APOSTROPHE (U+0027) (') shall be the absolute value of the offset from UT in minutes (00-59)

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DASetInformation(FileHandle: Integer; Key,
  NewValue: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DASetInformation(
  FileHandle As Long, Key As String, NewValue As String) As Long
```

DLL

```
int DPLDASetInformation(int InstanceID, int FileHandle, wchar_t * Key,
  wchar_t * NewValue);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
Key	For standard information use "Author", "Title", "Subject", "Keywords", "Creator", "Producer", "CreationDate" or "ModDate". For custom information any other string can be used.
NewValue	The new value for the specified key.

Return values

0	The specified FileHandle was not valid
1	The information key was set or updated successfully

DASetPageBox

Version history

This function was introduced in Quick PDF Library version 7.22.

Description

Sets the dimensions of the specified page's boundary rectangles.

The MediaBox represents the physical medium of the page.

The CropBox represents the visible region of the page, the contents will be clipped to this region.

The BleedBox is similar to the CropBox, but is the rectangle used in a production environment.

The TrimBox indicates the intended dimensions of the finished page after trimming, and the ArtBox defines the extent of the page's meaningful content as intended by the page's creator.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DASetPageBox(FileHandle, PageRef,
  BoxIndex: Integer; X1, Y1, X2, Y2: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DASetPageBox(FileHandle As Long,
  PageRef As Long, BoxIndex As Long, X1 As Double, Y1 As Double,
  X2 As Double, Y2 As Double) As Long
```

DLL

```
int DPLDASetPageBox(int InstanceID, int FileHandle, int PageRef,
  int BoxIndex, double X1, double Y1, double X2, double Y2);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
PageRef	A page reference returned by the **DAFindPage** or **DANewPage** functions
BoxIndex	1 = MediaBox 2 = CropBox 3 = BleedBox 4 = TrimBox 5 = ArtBox
X1	The horizontal coordinate of the bottom left corner of the box measured in points from the left edge of the page
Y1	The vertical coordinate of the bottom left corner of the box measured in points from the bottom of the page
X2	The horizontal coordinate of the top right corner of the box measured in points from the bottom of the page
Y2	The vertical coordinate of the top right corner of the box measured in points from the bottom of the page

Return values

0	The FileHandle or PageRef parameters were invalid
1	Success

DASetPageLayout

Document properties, Direct access functionality

Version history

This function was introduced in Quick PDF Library version 11.11.

Description

Sets the initial page layout of the document using the irect Access Functionality.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DASetPageLayout(FileHandle,
  NewPageLayout: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DASetPageLayout(
  FileHandle As Long, NewPageLayout As Long) As Long
```

DLL

```
int DPLDASetPageLayout(int InstanceID, int FileHandle, int NewPageLayout);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
NewPageLayout	0 = Single page 1 = One column 2 = Two columns, odd-numbered pages on left 3 = Two columns, odd-numbered pages on right 4 = Two pages, odd-numbered pages on left 5 = Two pages, odd-numbered pages on right 6 = No preference (setting removed from document)

Return values

0	The page layout could not be set
1	The page layout was set successfully

DASetPageMode

Version history

This function was introduced in Quick PDF Library version 11.11.

Description

Sets the initial page mode of the document using the Direct Access functionality.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DASetPageMode(FileHandle,
  NewPageMode: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DASetPageMode(
  FileHandle As Long, NewPageMode As Long) As Long
```

DLL

```
int DPLDASetPageMode(int InstanceID, int FileHandle, int NewPageMode);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
NewPageMode	0 = Normal view 1 = Show the outlines pane 2 = Show the thumbnails pane 3 = Show the document in full screen mode 4 = Optional content group panel visible 5 = Attachments panel visible

Return values

0	The page mode could not be set
1	The page mode was set successfully

DASetPageSize

Description

Sets the specified page to have a certain width and height.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DASetPageSize(FileHandle, PageRef: Integer;
  PntWidth, PntHeight: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DASetPageSize(
  FileHandle As Long, PageRef As Long, PntWidth As Double,
  PntHeight As Double) As Long
```

DLL

```
int DPLDASetPageSize(int InstanceID, int FileHandle, int PageRef,
  double PntWidth, double PntHeight);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions
PageRef	A page reference returned by the **DAFindPage** or **DANewPage** functions
PntWidth	The new width of the page, measured in points
PntHeight	The new height of the page, measured in points

Return values

0	The specified FileHandle or PageRef were not valid
1	The page size was set successfully

DASetTextExtractionArea

Text, Extraction, Direct access functionality

Version history

This function was introduced in Quick PDF Library version 8.13.

Description

Sets the area for certain modes of text extraction. Any text that appears outside this area will be excluded from the results. This function has no effect on text extraction using modes 0 to 2.

This function affects the results of the **ExtractFilePageText** and **DAExtractPageText** functions only.

The coordinate values passed into this function are specified in points with the bottom left corner of the page as the origin.

The area limitation can be removed by calling this function with a value of zero for both the Width and Height parameters.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DASetTextExtractionArea(Left, Top, Width,
  Height: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DASetTextExtractionArea(
  Left As Double, Top As Double, Width As Double,
  Height As Double) As Long
```

DLL

```
int DPLDASetTextExtractionArea(int InstanceID, double Left, double Top,
  double Width, double Height);
```

Parameters

Left	The horizontal coordinate of the left edge of the area
Top	The vertical coordinate of the top edge of the area
Width	The width of the area
Height	The height of the area

Return values

1	The text extraction area was set successfully
2	The text extraction area was cleared

DASetTextExtractionOptions

Text, Extraction, Direct access functionality

Version history

This function was introduced in Quick PDF Library version 8.13.

Description

Sets various options that affect the text extraction functionality.

This function affects the results of the **ExtractFilePageText** and **DAExtractPageText** functions only.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DASetTextExtractionOptions(OptionID,
  NewValue: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DASetTextExtractionOptions(
  OptionID As Long, NewValue As Long) As Long
```

DLL

```
int DPLDASetTextExtractionOptions(int InstanceID, int OptionID,
  int NewValue);
```

Parameters

OptionID	1 = Ignore Font changes to allow grouping different blocks together
	2 = Ignore Color changes to allow grouping different blocks together
	3 = Ignore Text Block changes to allow grouping different blocks together
	4 = Output CMYK color values
	5 = Sort text blocks based on top left position
	6 = Descenders from font metrics
	7 = Ignore overlaps
	8 = Ignore duplicates
	9 = Split on double space
	10 = Trim characters outside area
	11 = Alternative block matching
	12 = Ignore rotated text blocks
	13 = Trim leading and trailing whitespace from text blocks
	14 = Output non ASCII characters below Space character (0x32)
	15 = Remove certain character strings such as underscore lines (see below)
NewValue	For OptionID = 1, 2, 3 and 6:
	0 = Use, 1 = Ignore
	For OptionID = 4:
	0 = Show as RGB (default), 1 = Show as CMYK
	For OptionID = 5:
	0 = Do not sort blocks (default), 1 = Sort blocks
	For OptionID = 7, 8 and 12:
	0 = Do not ignore, 1 = Ignore
	OptionID = 9:
	0 = Do not split on double space (default)
	1 = Split on double space
	OptionID = 10:
	0 = Do not trim characters outside area (default)
	1 = Trim characters outside area
	OptionID = 11:
	0 = Regular block matching
	1 = Alternative block matching
	OptionID = 13:
	0 = Do not trim leading or trailing whitespace
	1 = Trim leading and trailing whitespace
	OptionID = 14
	0 = Remove non ASCII chracters below space character from output (default)
	1 = Output raw unfiltered ASCII characters
	OptionID = 15
	0 = Output text lines made with Underscore characters (default)
	1 = Remove text lines made with Underscore characters

Return values

0	The OptionID or NewValue parameter was not valid
1	The text extraction option was set successfully

DASetTextExtractionScaling

Text, Extraction, Direct access functionality

Version history

This function was introduced in Quick PDF Library version 8.16.

Description

Sets the scaling to use for text extraction in Mode 7. This controls the number of rows and columns in the monospaced text output.

This function affects the results of the **ExtractFilePageText** and **DAExtractPageText** functions only.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DASetTextExtractionScaling(
  Options: Integer; Horizontal, Vertical: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DASetTextExtractionScaling(
  Options As Long, Horizontal As Double,
  Vertical As Double) As Long
```

DLL

```
int DPLDASetTextExtractionScaling(int InstanceID, int Options,
  double Horizontal, double Vertical);
```

Parameters

Options	Should always be set to 0. This indicates a scaling factor will be set for the Horizontal and Vertical parameters, with a default value of 5 for horizontal and 8 for vertical. Smaller values stretch the text out into more rows/columns.
Horizontal	The scaling to use for the horizontal axis in units defined by the Options parameter.
Vertical	The scaling to use for the vertical axis in units defined by the Options parameter.

Return values

0	The Options parameter was not valid or a value less than 1 was used for the Horizontal or Vertical parameters.
1	Text extraction scaling was set successfully.

DASetTextExtractionWordGap

Text, Extraction, Direct access functionality

Version history

This function was introduced in Quick PDF Library version 8.13.

Description

Sets the word gap ratio for the text extraction functionality.

This function affects the results of the **ExtractFilePageText** and **DAExtractPageText** functions only.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DASetTextExtractionWordGap(
    NewWordGap: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DASetTextExtractionWordGap(
    NewWordGap As Double) As Long
```

DLL

```
int DPLDASetTextExtractionWordGap(int InstanceID, double NewWordGap);
```

Parameters

NewWordGap	The new WordGap ratio

Return values

1	The word gap ratio was set successfully.

DAShiftedHeader

Version history

> This function was introduced in Quick PDF Library version 9.15.

Description

> Returns a value to determine if the source PDF was malformed due to byte shifting. For example, leading whitespace added to the file.

> In such a case the file will be loaded taking this offset into account. This function will return a non-zero number indicating the number of bytes the file was shifted by.

> Note that if the file is loaded this way it will not be possible to use the **DAAppendFile** function to add an incremental update.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DAShiftedHeader(
  FileHandle: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DAShiftedHeader(
  FileHandle As Long) As Long
```

DLL

```
int DPLDAShiftedHeader(int InstanceID, int FileHandle);
```

Parameters

FileHandle	A handle returned by the **DAOpenFile**, **DAOpenFileReadOnly** or **DAOpenFromStream** functions

Return values

0	The file was loaded as usual
Non-zero	The number of bytes the file was shifted by

Decrypt

Version history

> This function was renamed in Quick PDF Library version 7.11.
> The function name in earlier versions was Unencrypt.

Description

> This function attempts to remove the encryption setting from the selected document using the password provided when originally opening the document.
>
> This function will succeed even if the user password was used (including an valid blank password) rather than the master password. Developers are advised that they should respect the security wishes of the document's author.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.Decrypt: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::Decrypt As Long
```

DLL

```
int DPLDecrypt(int InstanceID);
```

Return values

0	The document could not be decrypted
1	The document was decrypted successfully

DecryptFile

Description

This function attempts to remove the encryption from a file on disk, saving the decrypted document to a new file.

This function will succeed even if the user password is supplied (including an valid blank password) rather than the master password. Developers are advised that they should respect the security wishes of the document's author.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DecryptFile(InputFileName, OutputFileName,
  Password: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DecryptFile(
  InputFileName As String, OutputFileName As String,
  Password As String) As Long
```

DLL

```
int DPLDecryptFile(int InstanceID, wchar_t * InputFileName,
  wchar_t * OutputFileName, wchar_t * Password);
```

Parameters

InputFileName	The name of the file to decrypt.
OutputFileName	The name of the destination file to create. If this file already exists it will be overwritten.
Password	The password to use when decrypting the file.

Return values

0	The document could not be decrypted. Check the result of the **LastErrorCode** function to determine the cause of the failure.
1	The document was decrypted successfully

DeleteAnalysis

Description

Removes a set of analysis results from memory. Call this function after calling **AnalyseFile** and **GetAnalysisInfo** when you no longer need the information.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DeleteAnalysis(
  AnalysisID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DeleteAnalysis(
  AnalysisID As Long) As Long
```

DLL

```
int DPLDeleteAnalysis(int InstanceID, int AnalysisID);
```

Parameters

AnalysisID	The ID of the set of analysis results to delete, as returned by the **AnalyseFile** function

Return values

0	The specified analysis ID was not valid
1	The set of analysis results with the specified ID was deleted successfully

DeleteAnnotation

Version history

This function was introduced in Quick PDF Library version 7.16.

Description

Removes an annotation from the selected page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DeleteAnnotation(Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DeleteAnnotation(
  Index As Long) As Long
```

DLL

```
int DPLDeleteAnnotation(int InstanceID, int Index);
```

Parameters

Index	The index of the annotation to delete. The first annotation on the page has an index of 1. The **AnnotationCount** function returns the total number of annotations on the selected page.

Return values

0	The specified annotation could not be deleted. Check the value of the Index parameter is between 1 and the value returned by the **AnnotationCount** function.
1	The specified annotation was deleted from the page successfully.

DeleteContentStream

Version history

This function was renamed in Quick PDF Library version 8.11.
The function name in earlier versions was DeleteLayer.

Description

A page in a PDF document has one or more content stream parts that together contain all the PDF page description commands for the page.

This function removes the specified content stream part that was selected with the **SelectContentStream** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DeleteContentStream: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DeleteContentStream As Long
```

DLL

```
int DPLDeleteContentStream(int InstanceID);
```

Return values

0	The content stream part could not be deleted
1	The content stream part was deleted successfully

DeleteFormField

Form fields

Description

Deletes the specified form field. If the field is deleted successfully the field index of subsequent form fields will be decreased by 1.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DeleteFormField(Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DeleteFormField(
  Index As Long) As Long
```

DLL

```
int DPLDeleteFormField(int InstanceID, int Index);
```

Parameters

Index	The index of the form field to delete

Return values

0	The form field was not found
1	The form field was deleted successfully

DeleteOptionalContentGroup

Content Streams and Optional Content Groups

Description

Deletes an optional content group.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DeleteOptionalContentGroup(
  OptionalContentGroupID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DeleteOptionalContentGroup(
  OptionalContentGroupID As Long) As Long
```

DLL

```
int DPLDeleteOptionalContentGroup(int InstanceID,
  int OptionalContentGroupID);
```

Parameters

OptionalContentGroupID	An ID returned by the **NewOptionalContentGroup**, **GetOptionalContentGroupID** or **GetOptionalContentConfigOrderItemID** functions

DeletePageLGIDict

Version history

This function was introduced in Quick PDF Library version 7.15.

Description

Deletes the specified LGIDict dictionary from the selected page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DeletePageLGIDict(
  DictIndex: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DeletePageLGIDict(
  DictIndex As Long) As Long
```

DLL

```
int DPLDeletePageLGIDict(int InstanceID, int DictIndex);
```

Parameters

DictIndex	The index of the LGIDict dictionary to delete. The first dictionary has an index of 1. Use the **GetPageLGIDictCount** function to determine the number of LGIDict dictionaries attached to the selected page.

Return values

0	The dictionary could not be deleted. Check that the DictIndex parameter is in range.
1	The specified dictionary was deleted successfully.

DeletePages

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Removes one or more pages from the document. The document will always have at least one page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DeletePages(StartPage,
  PageCount: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DeletePages(StartPage As Long,
  PageCount As Long) As Long
```

DLL

```
int DPLDeletePages(int InstanceID, int StartPage, int PageCount);
```

Parameters

StartPage	The page number of the first page to delete
PageCount	The total number of pages to delete. The value will be automatically adjusted if necessary so that the document is left with at least one page.

Return values

0	The PageCount parameter was 0 or there was only a single page in the document.
Non-zero	The number of pages remaining in the original document.

DocJavaScriptAction

Document properties, JavaScript

Description

This function is used to add JavaScript to document events. This JavaScript will be executed when, for example, the document is closed or printed.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DocJavaScriptAction(ActionType,
  JavaScript: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DocJavaScriptAction(
  ActionType As String, JavaScript As String) As Long
```

DLL

```
int DPLDocJavaScriptAction(int InstanceID, wchar_t * ActionType,
  wchar_t * JavaScript);
```

Parameters

ActionType	The event to attach the JavaScript to: "WC" = Will close "WS" = Will save "DS" = Did save "WP" = Will print "DP" = Did print
JavaScript	The JavaScript to attach to the event.

Return values

0	The specified ActionType was not valid
1	The JavaScript was added successfully

DocumentCount

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Returns the total number of documents.

When an instance of Quick PDF Library is first created a blank one page document is automatically created so the document count will be 1. Each time a new document is created or loaded the document count will be increased. The **RemoveDocument** function will only succeed if there are at least two documents loaded, so the document count will always be at least 1.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DocumentCount: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DocumentCount As Long
```

DLL

```
int DPLDocumentCount(int InstanceID);
```

DrawArc

Description

Draw a circular arc on the selected page. The arc is drawn in a clockwise direction from StartAngle to EndAngle.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawArc(XPos, YPos, Radius, StartAngle,
  EndAngle: Double; Pie, DrawOptions: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawArc(XPos As Double,
  YPos As Double, Radius As Double, StartAngle As Double,
  EndAngle As Double, Pie As Long, DrawOptions As Long) As Long
```

DLL

```
int DPLDrawArc(int InstanceID, double XPos, double YPos, double Radius,
  double StartAngle, double EndAngle, int Pie, int DrawOptions);
```

Parameters

XPos	Horizontal co-ordinate of the center of the arc
YPos	Vertical co-ordinate of center of the arc
Radius	Radius of the arc
StartAngle	Angle to start drawing from
EndAngle	Angle to end drawing at
Pie	Draw the arms of the arc: 0 = No 1 = Yes
DrawOptions	0 = Outline 1 = Fill 2 = Fill and Outline 3 = Close, Fill and Outline (if Pie = 1)

DrawBarcode

Description

Draws a barcode on the selected page.

For Code128, the barcode is a combination of the "B" and "C" character sets resulting in the most compact representation.

GS1-128 barcodes (also known as EAN-128) can be drawn by setting the Barcode parameter to 3 (Code128) and using the string "[FNC1]" in the appropriate place. For example:
"[FNC1]21ABC123[FNC1]2013"

The previous example indicates a serial number (AI 21) of "ABC123" and a product variant (AI 20) of "13".

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawBarcode(Left, Top, Width,
  Height: Double; Text: WideString; Barcode, Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawBarcode(Left As Double,
  Top As Double, Width As Double, Height As Double,
  Text As String, Barcode As Long, Options As Long) As Long
```

DLL

```
int DPLDrawBarcode(int InstanceID, double Left, double Top, double Width,
  double Height, wchar_t * Text, int Barcode, int Options);
```

Parameters

Left	Horizontal co-ordinate of left edge of the barcode
Top	Vertical co-ordinate of top edge of the barcode
Width	Width of the barcode
Height	Height of the barcode
Text	The barcode data. The barcode can be rotated by appending the following to the barcode data string: /RC = Rotate clockwise /RA = Rotate anti-clockwise /RU = Rotate 180 degrees
Barcode	1 = Code39 (or Code 3 of 9) 2 = EAN-13 3 = Code128 4 = PostNet 5 = Interleaved 2 of 5
Options	Code39: 0 = Default drawing EAN-13: 0 = Only draw the barcode 1 = Extend the guard bars 2 = Draw the human-readable numbers 3 = Draw the human-readable numbers, with right spacer Code128: 0 = Default drawing PostNet: 0 = Default drawing Interleaved 2 of 5: 0 = Do not add a checksum, no bearer bars 1 = Add a checksum character, no bearer bars 2 = Do not add a checksum, draw bearer bars 3 = Add a checksum character, draw bearer bars To apply 10% bar width reduction to the barcode, increase the value of the Options parameter by 10

Return values

0	The barcode could not be drawn. Invalid Barcode or Options parameters.
1	The barcode was drawn successfully

DrawBox

Description

Draw a rectangle on the selected page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawBox(Left, Top, Width, Height: Double;
  DrawOptions: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawBox(Left As Double,
  Top As Double, Width As Double, Height As Double,
  DrawOptions As Long) As Long
```

DLL

```
int DPLDrawBox(int InstanceID, double Left, double Top, double Width,
  double Height, int DrawOptions);
```

Parameters

Left	Horizontal co-ordinate of left edge of rectangle
Top	Vertical co-ordinate of top edge of rectangle
Width	Rectangle width
Height	Rectangle height
DrawOptions	0 = Outline 1 = Fill 2 = Fill and Outline

DrawCapturedPage

Description

This function draws a page previously captured with the **CapturePage** function onto the current page. It can be drawn at any size and position, allowing for imposition of pages.

You cannot use CapturePage to move pages from one document to another so all the required pages must be merged into a single document before calling CapturePage. The CaptureID is just a pointer to a hidden page therefore does not need to be released.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawCapturedPage(CaptureID: Integer; Left,
  Top, Width, Height: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawCapturedPage(
  CaptureID As Long, Left As Double, Top As Double,
  Width As Double, Height As Double) As Long
```

DLL

```
int DPLDrawCapturedPage(int InstanceID, int CaptureID, double Left,
  double Top, double Width, double Height);
```

Parameters

CaptureID	The ID returned by the **CapturePage** function when a page was previously captured
Left	The co-ordinate of the left edge of the destination area
Top	The co-ordinate of the top edge of the destination area
Width	The width of the destination area
Height	The height of the destination area

Return values

0	An invalid CaptureID was specified
1	The captured page was drawn successfully

DrawCapturedPageMatrix

Page layout

Version history

This function was introduced in Quick PDF Library version 9.15.

Description

This function draws a page previously captured with the **CapturePage** function onto the current page. The size/position/rotation is specified using a transformation matrix, allowing for advanced imposition of pages.

You cannot use CapturePage to move pages from one document to another so all the required pages must be merged into a single document before calling CapturePage. The CaptureID is just a pointer to a hidden page therefore does not need to be released.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawCapturedPageMatrix(CaptureID: Integer;
  M11, M12, M21, M22, MDX, MDY: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawCapturedPageMatrix(
  CaptureID As Long, M11 As Double, M12 As Double,
  M21 As Double, M22 As Double, MDX As Double,
  MDY As Double) As Long
```

DLL

```
int DPLDrawCapturedPageMatrix(int InstanceID, int CaptureID, double M11,
  double M12, double M21, double M22, double MDX, double MDY);
```

Parameters

CaptureID	The ID returned by the **CapturePage** function when a page was previously captured
M11	Matrix component
M12	Matrix component
M21	Matrix component
M22	Matrix component
MDX	Matrix component
MDY	Matrix component

Return values

0	An invalid CaptureID was specified
1	The captured page was drawn successfully

DrawCircle

Description

Draw a circle on the selected page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawCircle(XPos, YPos, Radius: Double;
  DrawOptions: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawCircle(XPos As Double,
  YPos As Double, Radius As Double, DrawOptions As Long) As Long
```

DLL

```
int DPLDrawCircle(int InstanceID, double XPos, double YPos, double Radius,
  int DrawOptions);
```

Parameters

XPos	Horizontal co-ordinate of the center of the circle
YPos	Vertical co-ordinate of center of the circle
Radius	Size of the circle
DrawOptions	0 = Outline 1 = Fill 2 = Fill and Outline

DrawDataMatrixSymbol

Vector graphics, Barcodes

Description

This function draws a Data Matrix symbol onto the page. Data Matrix is a 2D barcode symbology allowing large amounts of data to be stored.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawDataMatrixSymbol(Left, Top,
  ModuleSize: Double; Text: WideString; Encoding, SymbolSize,
  Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawDataMatrixSymbol(
  Left As Double, Top As Double, ModuleSize As Double,
  Text As String, Encoding As Long, SymbolSize As Long,
  Options As Long) As Long
```

DLL

```
int DPLDrawDataMatrixSymbol(int InstanceID, double Left, double Top,
  double ModuleSize, wchar_t * Text, int Encoding,
  int SymbolSize, int Options);
```

Parameters

Left	The horizontal co-ordinate of the left edge of the symbol
Top	The vertical co-ordinate of the top edge of the symbol
ModuleSize	This value is used for the width and height of the dots which make up the symbol
Text	The text/data to store in the symbol
Encoding	1 = ASCII encoding. See the Data Matrix specification for details.
SymbolSize	0 = Auto size 1 = 10x10 2 = 12x12 3 = 8x18 4 = 14x14 5 = 8x32 6 = 16x16 7 = 12x26 8 = 18x18 9 = 20x20 10 = 12x36 11 = 22x22 12 = 16x36 13 = 24x24 14 = 26x26 15 = 16x48 16 = 32x32 17 = 36x36 18 = 40x40 19 = 44x44 20 = 48x48 21 = 52x52 22 = 64x64 23 = 72x72 24 = 80x80 25 = 88x88 26 = 96x96 27 = 104x104 28 = 120x120 29 = 132x132
Options	0 = Normal 1 = Rotate 90 degrees counter clockwise 2 = Rotate 180 degrees 3 = Rotate 90 degrees clockwise Add 100 to for 1 unit quiet zone (white border) - (default) Add 200 to for 2 units quiet zone Add 300 to for 3 units quiet zone Add 400 to for 4 units quiet zone

Return values

0	The Encoding, SymbolSize or Options parameter was invalid
1	The Data Matrix symbol was drawn successfully

DrawEllipse

Description

Draws an ellipse centered at a certain point which fits into the specified size box.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawEllipse(XPos, YPos, Width,
  Height: Double; DrawOptions: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawEllipse(XPos As Double,
  YPos As Double, Width As Double, Height As Double,
  DrawOptions As Long) As Long
```

DLL

```
int DPLDrawEllipse(int InstanceID, double XPos, double YPos, double Width,
  double Height, int DrawOptions);
```

Parameters

XPos	The horizontal co-ordinate of the center of the ellipse
YPos	The vertical co-ordinate of the center of the ellipse
Width	The width of the ellipse
Height	The height of the ellipse
DrawOptions	0 = Outline 1 = Fill 2 = Fill and Outline

DrawEllipticArc
Vector graphics

Description

Draws an arc which is the result of cutting an ellipse between the start angle and the end angle. The angles are measured anti-clockwise with 0 being at the top of the ellipse. ie. 12 O'Clock = 0 degrees and 9 O'Clock is 90 degrees.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawEllipticArc(XPos, YPos, Width, Height,
  StartAngle, EndAngle: Double; Pie, DrawOptions: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawEllipticArc(XPos As Double,
  YPos As Double, Width As Double, Height As Double,
  StartAngle As Double, EndAngle As Double, Pie As Long,
  DrawOptions As Long) As Long
```

DLL

```
int DPLDrawEllipticArc(int InstanceID, double XPos, double YPos,
  double Width, double Height, double StartAngle,
  double EndAngle, int Pie, int DrawOptions);
```

Parameters

XPos	The horizontal co-ordinate of the center of the ellipse
YPos	The vertical co-ordinate of the center of the ellipse
Width	The width of the ellipse
Height	The height of the ellipse
StartAngle	The angle to start the curve at
EndAngle	The angle to end the curve at
Pie	Draw the arms of the arc: 0 = No 1 = Yes
DrawOptions	0 = Outline 1 = Fill 2 = Fill and Outline 3 = Close, Fill and Outline (if Pie = 1)

DrawHTMLText

Text, HTML text, Page layout

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Draws HTML text onto the page. See **Appendix A** for details of the supported HTML tags.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawHTMLText(Left, Top, Width: Double;
  HTMLText: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawHTMLText(Left As Double,
  Top As Double, Width As Double, HTMLText As String) As Long
```

DLL

```
int DPLDrawHTMLText(int InstanceID, double Left, double Top, double Width,
  wchar_t * HTMLText);
```

Parameters

Left	The left edge of the area to draw the text into
Top	The top edge of the area to draw the text into
Width	The width of the area to draw the text into
HTMLText	The HTML text to draw

DrawHTMLTextBox

Text, HTML text, Page layout

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Similar to the **DrawHTMLText** function, but the text drawn is limited to a specific area. The remaining HTML text is returned, which can be passed to this function again (perhaps on a different page or location) until the function returns an empty string. See **Appendix A** for details of the supported HTML tags.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawHTMLTextBox(Left, Top, Width,
  Height: Double; HTMLText: WideString): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawHTMLTextBox(Left As Double,
  Top As Double, Width As Double, Height As Double,
  HTMLText As String) As String
```

DLL

```
wchar_t * DPLDrawHTMLTextBox(int InstanceID, double Left, double Top,
  double Width, double Height, wchar_t * HTMLText);
```

Parameters

Left	Horizontal co-ordinate of the left edge of the drawing area
Top	Vertical co-ordinate of the top edge of the drawing area
Width	The width of the drawing area
Height	The height of the drawing area
HTMLText	The HTML text to draw

Return values

LeftOverText	A "string" containing the text that did not fit into the TextBox. This value can be resused to draw the undrawn text into a new text box often on the next page.

DrawHTMLTextBoxMatrix

Version history

This function was introduced in Quick PDF Library version 9.15.

Description

Similar to the **DrawHTMLTextBox** function but the position/scaling/rotation is specified using a transformation matrix.

The remaining HTML text is returned, which can be passed to this function again (perhaps on a different page or location) until the function returns an empty string. See **Appendix A** for details of the supported HTML tags.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawHTMLTextBoxMatrix(Width,
  Height: Double; HTMLText: WideString; M11, M12, M21, M22, MDX,
  MDY: Double): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawHTMLTextBoxMatrix(
  Width As Double, Height As Double, HTMLText As String,
  M11 As Double, M12 As Double, M21 As Double, M22 As Double,
  MDX As Double, MDY As Double) As String
```

DLL

```
wchar_t * DPLDrawHTMLTextBoxMatrix(int InstanceID, double Width,
  double Height, wchar_t * HTMLText, double M11, double M12,
  double M21, double M22, double MDX, double MDY);
```

Parameters

Width	The width of the drawing area
Height	The height of the drawing area
HTMLText	The HTML text to draw
M11	Matrix component
M12	Matrix component
M21	Matrix component
M22	Matrix component
MDX	Matrix component
MDY	Matrix component

DrawHTMLTextMatrix

HTML text, Page layout

Version history

This function was introduced in Quick PDF Library version 10.11.

Description

Similar to the **DrawHTMLText** function but the position/scaling/rotation is specified using a transformation matrix.

See **Appendix A** for details of the supported HTML tags.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawHTMLTextMatrix(Width: Double;
  HTMLText: WideString; M11, M12, M21, M22, MDX, MDY: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawHTMLTextMatrix(
  Width As Double, HTMLText As String, M11 As Double,
  M12 As Double, M21 As Double, M22 As Double, MDX As Double,
  MDY As Double) As Long
```

DLL

```
int DPLDrawHTMLTextMatrix(int InstanceID, double Width,
  wchar_t * HTMLText, double M11, double M12, double M21,
  double M22, double MDX, double MDY);
```

Parameters

Width	The width of the area to draw the text into
HTMLText	The HTML text to draw
M11	Matrix component
M12	Matrix component
M21	Matrix component
M22	Matrix component
MDX	Matrix component
MDY	Matrix component

DrawImage

Image handling, Page layout

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Draw the selected image on the page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawImage(Left, Top, Width,
  Height: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawImage(Left As Double,
  Top As Double, Width As Double, Height As Double) As Long
```

DLL

```
int DPLDrawImage(int InstanceID, double Left, double Top, double Width,
  double Height);
```

Parameters

Left	Horizontal co-ordinate of the left edge of the image
Top	Vertical co-ordinate of the top edge of the image
Width	Width of the image
Height	Height of the image

Return values

0	An image has not been selected
1	The image was drawn successfully

DrawImageMatrix

Image handling, Page layout

Version history

This function was introduced in Quick PDF Library version 7.25.

Description

Draws the selected image on the page using a transformation matrix.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawImageMatrix(M11, M12, M21, M22, MDX,
  MDY: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawImageMatrix(M11 As Double,
  M12 As Double, M21 As Double, M22 As Double, MDX As Double,
  MDY As Double) As Long
```

DLL

```
int DPLDrawImageMatrix(int InstanceID, double M11, double M12, double M21,
  double M22, double MDX, double MDY);
```

Parameters

M11	Matrix component
M12	Matrix component
M21	Matrix component
M22	Matrix component
MDX	Matrix component
MDY	Matrix component

Return values

0	An image has not been selected
1	The image was drawn successfully

DrawIntelligentMailBarcode

Vector graphics, Barcodes

Version history

This function was introduced in Quick PDF Library version 8.15.

Description

This function draws a USPS Intelligent Mail (also known as OneCode) barcode onto the page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawIntelligentMailBarcode(Left, Top,
  BarWidth, FullBarHeight, TrackerHeight, SpaceWidth: Double;
  BarcodeData: WideString; Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawIntelligentMailBarcode(
  Left As Double, Top As Double, BarWidth As Double,
  FullBarHeight As Double, TrackerHeight As Double,
  SpaceWidth As Double, BarcodeData As String,
  Options As Long) As Long
```

DLL

```
int DPLDrawIntelligentMailBarcode(int InstanceID, double Left, double Top,
  double BarWidth, double FullBarHeight, double TrackerHeight,
  double SpaceWidth, wchar_t * BarcodeData, int Options);
```

Parameters

Left	Horizontal co-ordinate of the left edge of the barcode
Top	Vertical co-ordinate of the top edge of the barcode
BarWidth	The width of the bars
FullBarHeight	The height of a full bar
TrackerHeight	The height of a tracker bar
SpaceWidth	The width of the spaces between the bars
BarcodeData	The barcode data to encode. This should be a 20, 25, 29 or 31 character string containing only the digits 0 to 9. No spaces or any other non-numeric characters will be accepted. The second digit has a maximum value of 4.
Options	0 = Normal 10 = Bar width reduction

Return values

0	The barcode could not be drawn
1	The barcode was drawn successfully

DrawLine

Description

Draws a line between two points.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawLine(StartX, StartY, EndX,
  EndY: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawLine(StartX As Double,
  StartY As Double, EndX As Double, EndY As Double) As Long
```

DLL

```
int DPLDrawLine(int InstanceID, double StartX, double StartY, double EndX,
  double EndY);
```

Parameters

StartX	Horizontal co-ordinate of start point
StartY	Vertical co-ordinate of start point
EndX	Horizontal co-ordinate of end point
EndY	Vertical co-ordinate of end point

DrawMultiLineText

Description

Draw text which is wrapped at a specific delimiter. The **SetTextAlign** function can be used to change the alignment of the text.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawMultiLineText(XPos, YPos: Double;
  Delimiter, Text: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawMultiLineText(
  XPos As Double, YPos As Double, Delimiter As String,
  Text As String) As Long
```

DLL

```
int DPLDrawMultiLineText(int InstanceID, double XPos, double YPos,
  wchar_t * Delimiter, wchar_t * Text);
```

Parameters

XPos	The horizontal reference point of the text block
YPos	The baseline of the first line of text
Delimiter	The delimiter to use when splitting the text into lines. The only valid characters to use as the delimiter are characters which have a "width", as well as the CR and LF characters (ASCII values 13 and 10).
Text	The text to draw

DrawPDF417Symbol

Vector graphics, Barcodes

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Draws a PDF417 symbol onto the selected page.

From version 9.15 the **DrawPDF417SymbolEx** function can be used for extra functionality.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawPDF417Symbol(Left, Top: Double;
  Text: WideString; Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawPDF417Symbol(Left As Double,
  Top As Double, Text As String, Options As Long) As Long
```

DLL

```
int DPLDrawPDF417Symbol(int InstanceID, double Left, double Top,
  wchar_t * Text, int Options);
```

Parameters

Left	The horizontal coordinate of the left edge of the PDF417 symbol
Top	The vertical coordinate of the top edge of the PDF417 symbol
Text	The text to store in the symbol
Options	0 = Normal 1 = Rotate 90 degrees counter clockwise 2 = Rotate 180 degrees 3 = Rotate 90 degrees clockwise

Return values

0	The Options parameter was invalid
1	The PDF417 symbol was drawn successfully

DrawPDF417SymbolEx

Vector graphics, Barcodes

Version history

This function was introduced in Quick PDF Library version 9.15.

Description

Draws a PDF417 symbol onto the selected page. Similar to **DrawPDF417Symbol** but providing extra functionality.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawPDF417SymbolEx(Left, Top: Double;
  Text: WideString; Options, FixedColumns, FixedRows, ErrorLevel: Integer;
  ModuleSize, HeightWidthRatio: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawPDF417SymbolEx(
  Left As Double, Top As Double, Text As String,
  Options As Long, FixedColumns As Long, FixedRows As Long,
  ErrorLevel As Long, ModuleSize As Double,
  HeightWidthRatio As Double) As Long
```

DLL

```
int DPLDrawPDF417SymbolEx(int InstanceID, double Left, double Top,
  wchar_t * Text, int Options, int FixedColumns, int FixedRows,
  int ErrorLevel, double ModuleSize, double HeightWidthRatio);
```

Parameters

Left	The horizontal coordinate of the left edge of the PDF417 symbol
Top	The vertical coordinate of the top edge of the PDF417 symbol
Text	The text to store in the symbol
Options	0 = Normal 1 = Rotate 90 degrees counter clockwise 2 = Rotate 180 degrees 3 = Rotate 90 degrees clockwise
FixedColumns	0 = Auto Non-zero = fixed number of columns
FixedRows	0 = Auto Non-zero = fixed number of rows
ErrorLevel	-1 = Auto 0 to 8 = User error level
ModuleSize	The width of the smallest element in units defined by a call to **SetMeasurementUnits**
HeightWidthRatio	The ratio of the needed module height to the module width

Return values

0	One of the parameters was invalid or the text was too big for the symbol site.
1	The PDF417 symbol was drawn successfully

DrawPath

Vector graphics, Path definition and drawing

Description

Draws the path defined by calls to **StartPath**, **AddLineToPath**, **AddCurveToPath** and/or **ClosePath**.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawPath(PathOptions: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawPath(
  PathOptions As Long) As Long
```

DLL

```
int DPLDrawPath(int InstanceID, int PathOptions);
```

Parameters

PathOptions	0 = Outline
	1 = Fill
	2 = Fill and Outline

DrawPathEvenOdd

Description

Similar to the **DrawPath** function, but draws the path using the "even odd" method. This is important when different parts of the path overlap.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawPathEvenOdd(
  PathOptions: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawPathEvenOdd(
  PathOptions As Long) As Long
```

DLL

```
int DPLDrawPathEvenOdd(int InstanceID, int PathOptions);
```

Parameters

PathOptions	0 = Outline 1 = Fill 2 = Fill and outline

DrawPostScriptXObject

Description

Adds a reference to a PostScript XObject at the current position in the page contents.

This function is for specific advanced use and will not be useful to the majority of users.

For historical reasons, the PDF specification allows raw PostScript language commands to be embedded inside a document.

When the document is printed (using certain PDF software tools) on a PostScript printer, these raw PostScript commands will be sent directly to the printer.

Most PDF viewers are not able to display this embedded PostScript because this would require a full PostScript language interpreter.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawPostScriptXObject(
  PSRef: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawPostScriptXObject(
  PSRef As Long) As Long
```

DLL

```
int DPLDrawPostScriptXObject(int InstanceID, int PSRef);
```

Parameters

PSRef	A value that was returned by the **NewPostScriptXObject** function

Return values

0	The PostScript XObject could not be drawn
1	The PostScript XObject was drawn successfully

DrawQRCode

Vector graphics, Barcodes

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Version history

This function was introduced in Quick PDF Library version 10.11.

Description

Draws a QR Code onto the selected page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawQRCode(Left, Top, SymbolSize: Double;
  Text: WideString; EncodeOptions, DrawOptions: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawQRCode(Left As Double,
  Top As Double, SymbolSize As Double, Text As String,
  EncodeOptions As Long, DrawOptions As Long) As Long
```

DLL

```
int DPLDrawQRCode(int InstanceID, double Left, double Top,
  double SymbolSize, wchar_t * Text, int EncodeOptions,
  int DrawOptions);
```

Parameters

Left	The horizontal coordinate of the left edge of the QR Code
Top	The vertical coordinate of the top edge of the QR Code
SymbolSize	The width and height of the QR Code
Text	The text to encode in the QR Code
EncodeOptions	0=Auto 1=Numeric 2=Alphanumeric 3=ISO-8859-1 4=UTF-8 with BOM 5=UTF-8 without BOM
DrawOptions	0 = Normal 1 = Rotate 90 degrees counter clockwise 2 = Rotate 180 degrees 3 = Rotate 90 degrees clockwise

Return values

0	The QR Code could not be drawn, check for an out of range value for the EncodeOptions or DrawOptions parameter.
1	The QR Code was drawn successfully.

DrawRotatedBox

Vector graphics, Page manipulation

Version history

This function was introduced in Quick PDF Library version 8.14.

Description

Draws a rotated rectangle on the selected page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawRotatedBox(Left, Bottom, Width, Height,
  Angle: Double; DrawOptions: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawRotatedBox(Left As Double,
  Bottom As Double, Width As Double, Height As Double,
  Angle As Double, DrawOptions As Long) As Long
```

DLL

```
int DPLDrawRotatedBox(int InstanceID, double Left, double Bottom,
  double Width, double Height, double Angle, int DrawOptions);
```

Parameters

Left	The horizontal co-ordinate of the anchor point
Bottom	The vertical co-ordinate of the anchor point
Width	The width of the rectangle
Height	The height of the rectangle
Angle	The angle to rotate the rectangle, measured anti-clockwise in degrees from the baseline, around the anchor point (bottom-left of the rectangle)
DrawOptions	0 = Outline 1 = Fill 2 = Fill and Outline

DrawRotatedCapturedPage

Page layout, Page manipulation

Description

Similar to the **DrawCapturedPage** function, but allows the captured page to be drawn at any angle. Note that the anchor point is the bottom-left corner, not the top-left corner as with the **DrawCapturedPage** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawRotatedCapturedPage(CaptureID: Integer;
  Left, Bottom, Width, Height, Angle: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawRotatedCapturedPage(
  CaptureID As Long, Left As Double, Bottom As Double,
  Width As Double, Height As Double, Angle As Double) As Long
```

DLL

```
int DPLDrawRotatedCapturedPage(int InstanceID, int CaptureID, double Left,
  double Bottom, double Width, double Height, double Angle);
```

Parameters

CaptureID	The ID returned by the **CapturePage** function
Left	The horizontal co-ordinate of the anchor point
Bottom	The vertical co-ordinate of the anchor point
Width	The width of the rectangle to place the captured page in
Height	The height of the rectangle to place the captured page in
Angle	The angle to rotate the captured page by, measured anti-clockwise in degrees from the baseline

Return values

0	The CaptureID was not valid
1	The captured page was drawn successfully

DrawRotatedImage

Image handling, Page layout

Description

Similar to the **DrawImage** function but the image can be rotated at any angle. Note that the anchor point is the bottom left corner of the image, not the top-left as in the **DrawImage** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawRotatedImage(Left, Bottom, Width,
  Height, Angle: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawRotatedImage(Left As Double,
  Bottom As Double, Width As Double, Height As Double,
  Angle As Double) As Long
```

DLL

```
int DPLDrawRotatedImage(int InstanceID, double Left, double Bottom,
  double Width, double Height, double Angle);
```

Parameters

Left	The horizontal co-ordinate of the anchor point
Bottom	The vertical co-ordinate of the anchor point
Width	The width of the image
Height	The height of the image
Angle	The angle to rotate the image, measured anti-clockwise in degrees from the baseline, around the anchor point (bottom-left of the image)

Return values

0	No image has been selected
1	The image was drawn successfully

DrawRotatedMultiLineText

Version history

This function was introduced in Quick PDF Library version 8.14.

Description

Draws rotated text which is wrapped at a specific delimiter.

The **SetTextAlign** function can be used to change the alignment of the text.

The first line of text will start with the baseline at the anchor point used for rotation.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawRotatedMultiLineText(XPos, YPos,
  Angle: Double; Delimiter, Text: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawRotatedMultiLineText(
  XPos As Double, YPos As Double, Angle As Double,
  Delimiter As String, Text As String) As Long
```

DLL

```
int DPLDrawRotatedMultiLineText(int InstanceID, double XPos, double YPos,
  double Angle, wchar_t * Delimiter, wchar_t * Text);
```

Parameters

XPos	The horizontal coordinate of the anchor point
YPos	The vertical coordinate of the anchor point
Angle	The angle to rotate the text, measured anti-clockwise in degrees from the baseline, around the anchor point
Delimiter	The delimiter to use when splitting the text into lines. The only valid characters to use as the delimiter are characters which have a "width", as well as the CR and LF characters (ASCII values 13 and 10).
Text	The text to draw

DrawRotatedText

Description

Draws text on the selected page, using the selected font at the predetermined font size. If no fonts have been added, then the standard font Helvetica will automatically be added, selected and set to 12pt. The alignment of the text is determined by the previous call to the **SetTextAlign** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawRotatedText(XPos, YPos, Angle: Double;
  Text: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawRotatedText(XPos As Double,
  YPos As Double, Angle As Double, Text As String) As Long
```

DLL

```
int DPLDrawRotatedText(int InstanceID, double XPos, double YPos,
  double Angle, wchar_t * Text);
```

Parameters

XPos	The horizontal position of where to draw the text
YPos	The vertical position of where to draw the text. The reference point is the text baseline.
Angle	The angle to draw the text, measured anti-clockwise from the horizontal. Must be between 0 and 360, inclusive.
Text	The text to draw on the page

Return values

0	The Angle parameter was less than 0 or greater than 360, or the Text parameter was blank
1	The text was drawn successfully

DrawRotatedTextBox

Description

Similar to the **DrawTextBox** function, but allows the text box to be rotated at any angle.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawRotatedTextBox(Left, Top, Width,
  Height, Angle: Double; Text: WideString; Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawRotatedTextBox(
  Left As Double, Top As Double, Width As Double,
  Height As Double, Angle As Double, Text As String,
  Options As Long) As Long
```

DLL

```
int DPLDrawRotatedTextBox(int InstanceID, double Left, double Top,
  double Width, double Height, double Angle, wchar_t * Text,
  int Options);
```

Parameters

Left	The horizontal co-ordinate of the top-left corner of the text box
Top	The vertical co-ordinate of the top-left corner of the text box
Width	The width of the box
Height	The height of the box
Angle	The angle the box should be rotated around the top-left corner, measured anti-clockwise in degrees
Text	The text to place in the box
Options	0 = Center vertical alignment 1 = Top vertical alignment 2 = Bottom vertical alignment 3 = Center vertical alignment, no wrapping 4 = Top vertical alignment, no wrapping 5 = Bottom vertical alignment, no wrapping

Return values

0	The Options parameter was out of range, or the Width parameter was too small to contain any text
Non-zero	The number of lines of text actually drawn

DrawRotatedTextBoxEx

Description

Similar to the **DrawRotatedTextBoxEx** function, but allows the text box to show borders.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawRotatedTextBoxEx(Left, Top, Width,
  Height, Angle: Double; Text: WideString; Options, Border, Radius,
  DrawOptions: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawRotatedTextBoxEx(
  Left As Double, Top As Double, Width As Double,
  Height As Double, Angle As Double, Text As String,
  Options As Long, Border As Long, Radius As Long,
  DrawOptions As Long) As Long
```

DLL

```
int DPLDrawRotatedTextBoxEx(int InstanceID, double Left, double Top,
  double Width, double Height, double Angle, wchar_t * Text,
  int Options, int Border, int Radius, int DrawOptions);
```

Parameters

Left	The horizontal co-ordinate of the top-left corner of the text box
Top	The vertical co-ordinate of the top-left corner of the text box
Width	The width of the box
Height	The height of the box
Angle	The angle the box should be rotated around the top-left corner, measured anti-clockwise in degrees
Text	The text to draw on the page
Options	0 = Center vertical alignment 1 = Top vertical alignment 2 = Bottom vertical alignment 3 = Center vertical alignment, no wrapping 4 = Top vertical alignment, no wrapping 5 = Bottom vertical alignment, no wrapping
Border	0 = No Border 1 = Border 2 = Border with rounded corners
Radius	Radius of the corner arcs
DrawOptions	0 = Outline 1 = Fill 2 = Fill and outline

Return values

0	The Options parameter was out of range, or the Width parameter was too small to contain any text
Non-zero	The number of lines of text actually drawn

DrawRoundedBox

Vector graphics, Page layout

Description

Draw a rectangle with rounded corners on the selected page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawRoundedBox(Left, Top, Width, Height,
  Radius: Double; DrawOptions: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawRoundedBox(Left As Double,
  Top As Double, Width As Double, Height As Double,
  Radius As Double, DrawOptions As Long) As Long
```

DLL

```
int DPLDrawRoundedBox(int InstanceID, double Left, double Top,
  double Width, double Height, double Radius, int DrawOptions);
```

Parameters

Left	Horizontal co-ordinate of left edge of rectangle
Top	Vertical co-ordinate of top edge of rectangle
Width	Rectangle width
Height	Rectangle height
Radius	Radius of the corner arcs
DrawOptions	0 = Outline 1 = Fill 2 = Fill and outline

DrawRoundedRotatedBox

Vector graphics, Page layout

Description

Draw a rotated rectangle with rounded corners on the selected page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawRoundedRotatedBox(Left, Bottom, Width,
  Height, Radius, Angle: Double; DrawOptions: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawRoundedRotatedBox(
  Left As Double, Bottom As Double, Width As Double,
  Height As Double, Radius As Double, Angle As Double,
  DrawOptions As Long) As Long
```

DLL

```
int DPLDrawRoundedRotatedBox(int InstanceID, double Left, double Bottom,
  double Width, double Height, double Radius, double Angle,
  int DrawOptions);
```

Parameters

Left	Horizontal co-ordinate of left edge of rectangle
Bottom	Vertical co-ordinate of bottom edge of rectangle
Width	Rectangle width
Height	Rectangle height
Radius	Radius of the corner arcs
Angle	The angle the box should be rotated around the bottom-left corner, measured anti-clockwise in degrees
DrawOptions	0 = Outline 1 = Fill 2 = Fill and outline

DrawScaledImage

Image handling, Page layout

Description

Draw the selected image on the page. The image is drawn at the scale specified, assuming 72 DPI for both the horizontal and vertical resolution.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawScaledImage(Left, Top,
  Scale: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawScaledImage(Left As Double,
  Top As Double, Scale As Double) As Long
```

DLL

```
int DPLDrawScaledImage(int InstanceID, double Left, double Top,
  double Scale);
```

Parameters

Left	Horizontal co-ordinate of the left edge of the image
Top	Vertical co-ordinate of the top edge of the image
Scale	The scale to use, for example: 0.5 = 50% 1 = 100%

Return values

0	An image was not selected
1	The image was drawn successfully

DrawSpacedText

Description

Draws text on the selected page, using the selected font at the predetermined font size. If no fonts have been added, then the 12 pt Helvetica will automatically be added and selected. Each character will be spaced at regular intervals. The individual characters will be aligned relative to the XPos variable depending on how the **SetTextAlign** function has been used.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawSpacedText(XPos, YPos, Spacing: Double;
  Text: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawSpacedText(XPos As Double,
  YPos As Double, Spacing As Double, Text As String) As Long
```

DLL

```
int DPLDrawSpacedText(int InstanceID, double XPos, double YPos,
  double Spacing, wchar_t * Text);
```

Parameters

XPos	The horizontal position of where to draw the text
YPos	The vertical position of where to draw the text. The reference point is the text baseline.
Spacing	The spacing between the same point on each character
Text	The text to draw on the page

DrawTableRows

Version history

This function was introduced in Quick PDF Library version 7.14.

Description

Draws multiple rows from the specified table onto the selected page and returns the total height of the drawn rows. Only the number of rows that fit into the specified height will be drawn. Use the **GetTableLastDrawnRow** function to determine the row number of the last row.
In Quick PDF Library version 7.18 and earlier the result of this function was always reported in points. From version 7.19 and later the value returned by this function is correctly scaled according to the current co-ordinate system settings as set by the **SetMeasurementUnits** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawTableRows(TableID: Integer; Left, Top,
  Height: Double; FirstRow, LastRow: Integer): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawTableRows(TableID As Long,
  Left As Double, Top As Double, Height As Double,
  FirstRow As Long, LastRow As Long) As Double
```

DLL

```
double DPLDrawTableRows(int InstanceID, int TableID, double Left,
  double Top, double Height, int FirstRow, int LastRow);
```

Parameters

TableID	A TableID returned by the **CreateTable** function
Left	The horizontal distance from the origin to the left edge of the table
Top	The vertical distance from the origin to the top of the table
Height	The maximum height available to draw the table in
FirstRow	The the number of the first row to draw. Top row is row number 1.
LastRow	0 = All remaining rows Non-zero = The number of the final row to set

Return values

0	No rows were drawn
Non-zero	The total height of all the rows that were drawn onto the page.

DrawText

Text, Page layout

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Draws text on the selected page, using the selected font at the predetermined font size. If no fonts have been added, then 12 pt Helvetica will automatically be added and selected. The alignment of the text can be changed with the **SetTextAlign** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawText(XPos, YPos: Double;
  Text: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawText(XPos As Double,
  YPos As Double, Text As String) As Long
```

DLL

```
int DPLDrawText(int InstanceID, double XPos, double YPos, wchar_t * Text);
```

Parameters

XPos	The horizontal position of where to draw the text. The reference point is usually to the left of the first character, unless the **SetTextAlign** function has been used to change the alignment.
YPos	The vertical position of where to draw the text. The reference point is the text baseline.
Text	The text to draw on the page

DrawTextArc

Description

Draws text fitted to an imaginary arc with the specified center point and radius. The text will be drawn with it's left edge at the requested angle, where 0 degrees is the "12 o'clock" position, and positive angles are clockwise. The **SetTextAlign** function can be used to change the alignment of the text relative to the specified angle.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawTextArc(XPos, YPos, Radius,
  Angle: Double; Text: WideString; DrawOptions: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawTextArc(XPos As Double,
  YPos As Double, Radius As Double, Angle As Double,
  Text As String, DrawOptions As Long) As Long
```

DLL

```
int DPLDrawTextArc(int InstanceID, double XPos, double YPos,
  double Radius, double Angle, wchar_t * Text, int DrawOptions);
```

Parameters

XPos	The horizontal co-ordinate of the center of the arc
YPos	The vertical co-ordinate of the center of the arc
Radius	The radius of the arc
Angle	The angle at which the text should be placed
Text	The actual text to draw
DrawOptions	0 = Draw the text outside the arc in a clockwise direction 1 = Draw the text inside the arc in an anti-clockwise direction

Return values

0	The text was blank or the DrawOptions parameter was out of range
1	The text was drawn successfully

DrawTextBox

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

This function is similar to the **DrawText** function, but the text is placed within the bounding box specified. The vertical alignment can be set using the Options parameter, and the horizontal alignment can be set with the **SetTextAlign** function. The text will be word-wrapped to fit inside the bounding box.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawTextBox(Left, Top, Width,
  Height: Double; Text: WideString; Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawTextBox(Left As Double,
  Top As Double, Width As Double, Height As Double,
  Text As String, Options As Long) As Long
```

DLL

```
int DPLDrawTextBox(int InstanceID, double Left, double Top, double Width,
  double Height, wchar_t * Text, int Options);
```

Parameters

Left	The horizontal co-ordinate of the left edge of the bounding box
Top	The vertical co-ordinate of the top edge of the bounding box
Width	The width of the bounding box
Height	The height of the bounding box
Text	The text to draw on the page
Options	0 = Center vertical alignment 1 = Top vertical alignment 2 = Bottom vertical alignment 3 = Center vertical alignment, no wrapping 4 = Top vertical alignment, no wrapping 5 = Bottom vertical alignment, no wrapping

Return values

0	The Options parameter was out of range, or the Width parameter was too small to contain any text
Non-zero	The number of lines of text actually drawn

DrawTextBoxMatrix

Version history

This function was introduced in Quick PDF Library version 9.15.

Description

This function is similar to the **DrawTextBox** function but the position/scaling/rotation is specified using a transformation matrix.

The vertical alignment can be set using the Options parameter, and the horizontal alignment can be set with the **SetTextAlign** function. The text will be word-wrapped to fit inside the bounding box.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawTextBoxMatrix(Width, Height: Double;
  Text: WideString; Options: Integer; M11, M12, M21, M22, MDX,
  MDY: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawTextBoxMatrix(
  Width As Double, Height As Double, Text As String,
  Options As Long, M11 As Double, M12 As Double, M21 As Double,
  M22 As Double, MDX As Double, MDY As Double) As Long
```

DLL

```
int DPLDrawTextBoxMatrix(int InstanceID, double Width, double Height,
  wchar_t * Text, int Options, double M11, double M12,
  double M21, double M22, double MDX, double MDY);
```

Parameters

Width	The width of the bounding box
Height	The height of the bounding box
Text	The text to draw on the page
Options	0 = Center vertical alignment 1 = Top vertical alignment 2 = Bottom vertical alignment 3 = Center vertical alignment, no wrapping 4 = Top vertical alignment, no wrapping 5 = Bottom vertical alignment, no wrapping
M11	Matrix component
M12	Matrix component
M21	Matrix component
M22	Matrix component
MDX	Matrix component
MDY	Matrix component

Return values

0	The Options parameter was out of range, or the Width parameter was too small to contain any text
Non-zero	The number of lines of text actually drawn

DrawWrappedText

Text, Page layout

Description

Draw text which is wrapped to a certain width. The **SetTextAlign** function can be used to change the alignment of the text. The **SetBreakString** function can be used to set the delimiter for the linebreak. The default is CR / LF pair. On some systems a LF may be default.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.DrawWrappedText(XPos, YPos, Width: Double;
  Text: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::DrawWrappedText(XPos As Double,
  YPos As Double, Width As Double, Text As String) As Long
```

DLL

```
int DPLDrawWrappedText(int InstanceID, double XPos, double YPos,
  double Width, wchar_t * Text);
```

Parameters

XPos	The left edge of the text block
YPos	The baseline of the first line of text
Width	The width of the text block
Text	The text to draw

EditableContentStream

Content Streams and Optional Content Groups

Version history

This function was renamed in Quick PDF Library version 8.11.
The function name in earlier versions was EditableLayer.

Description

Use this function to determine if the content stream part that was selected with the
SelectContentStream function can be drawn on.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.EditableContentStream: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::EditableContentStream As Long
```

DLL

```
int DPLEditableContentStream(int InstanceID);
```

Return values

0	The selected content stream part cannot be drawn on
1	The selected content stream part is editable

EmbedFile

Description

Embeds a file into the PDF document and creates a file attachment link to the embedded file. The file can then be accessed in Acrobat under the File Attachments function.

This is equivalent to calling **AddEmbeddedFile** followed by **AddFileAttachment**.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.EmbedFile(Title, FileName,
  MIMEType: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::EmbedFile(Title As String,
  FileName As String, MIMEType As String) As Long
```

DLL

```
int DPLEmbedFile(int InstanceID, wchar_t * Title, wchar_t * FileName,
  wchar_t * MIMEType);
```

Parameters

Title	A unique title for this file. No two files can have the same title. If a file with this title already exists in the document the new file will not be embedded.
FileName	The full path and name of the file to embed.
MIMEType	The optional MIME type of the file, for example "image/jpg" for a JPEG image. See http://www.iana.org/assignments/media-types/ for a full list of MIME types. If the MIME type is not known it can be set to an empty string.

Return values

0	The file could not be embedded
1	The file was embedded successfully

EmbeddedFileCount

Version history

This function was introduced in Quick PDF Library version 7.13.

Description

Returns the number of embedded files in the document.

This total only includes embedded files that are listed as file attachments and does not include embedded files that are only referenced by a link annotation.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.EmbeddedFileCount: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::EmbeddedFileCount As Long
```

DLL

```
int DPLEmbeddedFileCount(int InstanceID);
```

EncapsulateContentStream

Version history

This function was renamed in Quick PDF Library version 8.11.
The function name in earlier versions was EncapsulateLayer.

Description

A page in a PDF document has one or more content stream parts that together contain all the PDF page description commands for the page.

This function combines the content stream parts and surrounds the content stream with "save graphics state" and "restore graphics state" operators. This has the effect of clearing the current clipping path.

Some pages may contain unbalanced "save graphics state" and "restore graphics state" operators. The **BalanceContentStream** function can be used to repair such pages.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.EncapsulateContentStream: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::EncapsulateContentStream As Long
```

DLL

```
int DPLEncapsulateContentStream(int InstanceID);
```

EncodePermissions

Version history

This function was renamed in Quick PDF Library version 7.11.
The function name in earlier versions was Permissions.

Description

Create a value for the Permissions parameter of the **Encrypt** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.EncodePermissions(CanPrint, CanCopy,
  CanChange, CanAddNotes, CanFillFields, CanCopyAccess, CanAssemble,
  CanPrintFull: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::EncodePermissions(
  CanPrint As Long, CanCopy As Long, CanChange As Long,
  CanAddNotes As Long, CanFillFields As Long,
  CanCopyAccess As Long, CanAssemble As Long,
  CanPrintFull As Long) As Long
```

DLL

```
int DPLEncodePermissions(int InstanceID, int CanPrint, int CanCopy,
  int CanChange, int CanAddNotes, int CanFillFields,
  int CanCopyAccess, int CanAssemble, int CanPrintFull);
```

Parameters

CanPrint	Set this to 1 to allow the user to print the document
CanCopy	Set this to 1 to allow the user to copy text and graphics from the document
CanChange	Set this to 1 to allow the user to edit the document
CanAddNotes	Set this to 1 to allow the user to add annotations
CanFillFields	Set this to 1 to allow the user to fill in form fields. Only works with 128-bit encryption.
CanCopyAccess	Set this to 1 to enable copying for use with accessibility features. Only works with 128-bit encryption.
CanAssemble	Set this to 1 to allow the user to assemble the document. Only works with 128-bit encryption.
CanPrintFull	Set this to 0 to force low-resolution printing of the document only. This prevents the document from being distilled into a new PDF document. Only works with 128-bit encryption or higher.

Return values

Result is a 32-bit encoded number which should be passed to the **Encrypt** function

EncodeStringFromVariant

Version history

This function was introduced in Quick PDF Library version 7.18.

Description

This function is used to encode a string in UTF-16LE format from an array of numbers stored as a Variant type.

Syntax

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::EncodeStringFromVariant(
    NumberList As Variant, Encoding As String,
    UnmatchedAction As Long) As String
```

Parameters

NumberList	A variant array of numbers. The numbers in the array can be stored in any ordinal variant type (signed or unsigned integers from 8 to 32 bits).
Encoding	A string that defines how numbers in the array should be mapped to character codes: "Unicode" = The numbers represent Unicode code points with values ranging from 0x0000 to 0x10FFFD. "UTF-8" = The numbers represent the bytes of Unicode code points encoded using the variable-length UTF-8 encoding scheme with values ranging from 0 to 244. "UTF-16" = The numbers represent the 16-bit values of Unicode code points encoded using the variable-length UTF-16 encoding scheme with values ranging from 0 to 65533. Unicode values from U+010000 to U+10FFFD are represented by a surrogate pair consisting of a sequence of two numbers. "UTF-16LE" = The numbers represent the bytes of the UTF-16 encoding scheme stored in little-endian format with values ranging from 0 to 255. "UTF-16BE" = The numbers represent the bytes of the UTF-16 encoding scheme stored in big-endian format with values ranging from 0 to 255. "CP932" = The numbers represent either individual bytes or a combination of 8-bit and 16-bit values from Microsoft code page 932 (an extension of Shift JIS encoding). Double-byte values can be presented as a 16-bit number or as two 8-bit numbers. For encodings where numbers represent bytes this function will cast signed 8-bit values to unsigned 8-bit values.
UnmatchedAction	Specifies how to handle numbers that are out of range or that map to invalid character codes: 0 = Unmatched characters are ignored 1 = Unmatched characters are replaced with the Unicode U+FFFD replacement character

Encrypt

Description

This function adds the specified security settings to the selected document.

From Quick PDF Library 8.11, the actual encryption of the document is delayed until the document is saved so this function can be called at any time, even before further content is added to the document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.Encrypt(Owner, User: WideString; Strength,
  Permissions: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::Encrypt(Owner As String,
  User As String, Strength As Long, Permissions As Long) As Long
```

DLL

```
int DPLEncrypt(int InstanceID, wchar_t * Owner, wchar_t * User,
  int Strength, int Permissions);
```

Parameters

Owner	The owner or master password for the document
User	The user password for the document
Strength	The strength of encryption to use: 0 = 40-bit encryption 1 = 128-bit RC4 encryption 2 = 128-bit AES encryption (requires Acrobat 7 or later) 3 = 256-bit AES encryption (requires Acrobat 9 or later) 4 = 256-bit AES encryption (requires Acrobat X or later)
Permissions	A value created with the **EncodePermissions** function

Return values

0	The document could not be encrypted. Use the **LastErrorCode** function to determine the reason for failure.
1	The document was encrypted successfully

EncryptFile

Description

Encrypts a file on disk and saves the results to a new file. The entire document does not have to be loaded into memory so this function can be used to encrypt huge documents.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.EncryptFile(InputFileName, OutputFileName,
   Owner, User: WideString; Strength, Permissions: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::EncryptFile(
   InputFileName As String, OutputFileName As String,
   Owner As String, User As String, Strength As Long,
   Permissions As Long) As Long
```

DLL

```
int DPLEncryptFile(int InstanceID, wchar_t * InputFileName,
   wchar_t * OutputFileName, wchar_t * Owner, wchar_t * User,
   int Strength, int Permissions);
```

Parameters

InputFileName	The name of the file to encrypt.
OutputFileName	The name of the destination file to create.
Owner	The owner password to use for the encrypted file. This is sometimes called the "master" password or the "permissions" password. This password will be needed to change the document.
User	The user password to use for the encrypted file. This is sometimes called the "open" password, it will allow the user to open the document but not to use the document in ways not permitted.
Strength	The strength of encryption to use: 0 = 40-bit RC4 encryption 1 = 128-bit RC4 encryption 2 = 128-bit AES encryption (requires Acrobat 7 or later) 3 = 256-bit AES encryption (requires Acrobat 9 or later) 4 = 256-bit AES encryption (requires Acorbat X or later)
Permissions	A value created with the **EncodePermissions** function

Return values

0	The file could not be encrypted. Check the result of the **LastErrorCode** function to determine the cause of the failure.
1	The document was encrypted successfully

EncryptWithFingerprint

Description

Encrypts the selected document using the encryption "fingerprint" obtained from another document using the **GetEncryptionFingerprint** function. The selected document will be encrypted with the same owner and user passwords as the document the fingerprint was taken from.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.EncryptWithFingerprint(
    Fingerprint: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::EncryptWithFingerprint(
    Fingerprint As String) As Long
```

DLL

```
int DPLEncryptWithFingerprint(int InstanceID, wchar_t * Fingerprint);
```

Parameters

Fingerprint	A fingerprint returned by the **GetEncryptionFingerprint** function

Return values

0	The fingerprint was invalid or the document was already encrypted
1	The document was successfully encrypted using the supplied fingerprint

EncryptionAlgorithm

Document properties, Security and Signatures

Version history

This function was introduced in Quick PDF Library version 7.11.

Description

Returns the encryption algorithm used to encrypt the selected document.
The **EncryptionStrength** function can be used to determine the encryption key length.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.EncryptionAlgorithm: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::EncryptionAlgorithm As Long
```

DLL

```
int DPLEncryptionAlgorithm(int InstanceID);
```

Return values

0	The document is not encrypted
1	The document is encrypted using RC4 encryption
2	The document is encrypted using AES encryption

EncryptionStatus

Document properties, Security and Signatures

Version history

This function was renamed in Quick PDF Library version 7.11.
The function name in earlier versions was Encrypted.

Description

Determines the encryption status of the selected document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.EncryptionStatus: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::EncryptionStatus As Long
```

DLL

```
int DPLEncryptionStatus(int InstanceID);
```

Return values

0	The selected document is not encrypted
1	The document is encrypted with Adobe "Standard" encryption
2	The document is encrypted with an unknown encryption

EncryptionStrength

Document properties, Security and Signatures

Description

If the selected document has been encrypted this function returns the encryption strength. This is the length of the key used to encrypt the contents of the document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.EncryptionStrength: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::EncryptionStrength As Long
```

DLL

```
int DPLEncryptionStrength(int InstanceID);
```

Return values

0	The selected document is not encrypted
40	The document has been encrypted with 40-bit encryption (Adobe Acrobat 3.x and 4.x)
128	The document has been encrypted with 128-bit encryption (Adobe Acrobat 5.x)
256	The document has been encrypted with 256-bit encryption (Acrobat 9 or Acrobat 10). Use the **SecurityInfo** function to determine which version of encryption was used.

EndPageUpdate

Version history

This function was introduced in Quick PDF Library version 7.12.

Description

For detailed page layouts the **BeginPageUpdate** function can be called before a group of drawing commands. The page layout commands will then be buffered until a matching call to this function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.EndPageUpdate: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::EndPageUpdate As Long
```

DLL

```
int DPLEndPageUpdate(int InstanceID);
```

EndSignProcessToFile

Version history

This function was introduced in Quick PDF Library version 9.13.

Description

Completes a digital signature process and writes the signed document to a file.

The result returned by EndSignProcessToFile will always be zero. To check the result of the digital signature signing process call the **GetSignProcessResult** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.EndSignProcessToFile(
  SignProcessID: Integer; OutputFile: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::EndSignProcessToFile(
  SignProcessID As Long, OutputFile As String) As Long
```

DLL

```
int DPLEndSignProcessToFile(int InstanceID, int SignProcessID,
  wchar_t * OutputFile);
```

Parameters

SignProcessID	A value returned by the **NewSignProcessFromFile**, **NewSignProcessFromStream** or **NewSignProcessFromString** functions.
OutputFile	The path and name of the file to save the signed PDF to.

EndSignProcessToStream

Version history

This function was introduced in Quick PDF Library version 9.13.

Description

Completes a digital signature process and writes the signed document to a TStream.

The result returned by EndSignProcessToStream will always be zero. To check the result of the digital signature signing process call the **GetSignProcessResult** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.EndSignProcessToStream(
  SignProcessID: Integer; OutputStream: TStream): Integer;
```

Parameters

SignProcessID	A value returned by the **NewSignProcessFromFile**, **NewSignProcessFromStream** or **NewSignProcessFromString** functions.
OutputStream	The TStream object to write the signed PDF to.

EndSignProcessToString

Version history

This function was introduced in Quick PDF Library version 9.13.

Description

Completes a digital signature process and returns the signed document as a string of 8-bit bytes.

The result returned by EndSignProcessToString will always be zero. To check the result of the digital signature signing process call the **GetSignProcessResult** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.EndSignProcessToString(
  SignProcessID: Integer): AnsiString;
```

DLL

```
char * DPLEndSignProcessToString(int InstanceID, int SignProcessID);
```

Parameters

SignProcessID	A value returned by the **NewSignProcessFromFile**, **NewSignProcessFromStream** or **NewSignProcessFromString** functions.

ExtractFilePageContentToString

Version history

This function was renamed in Quick PDF Library version 8.11.
The function name in earlier versions was ExtractFilePageContent.

Description

Retrieves the page description operators that define the layout of any page in a PDF document. This function does not load the entire file into memory so it can be used with arbitrarily large documents.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ExtractFilePageContentToString(
  InputFileName, Password: WideString; Page: Integer): AnsiString;
```

DLL

```
char * DPLExtractFilePageContentToString(int InstanceID,
  wchar_t * InputFileName, wchar_t * Password, int Page);
```

Parameters

InputFileName	The path and file name of the file to extract page content from.
Password	The password to use when opening the file
Page	The number of the page to extract. The first page in the document is page 1.

ExtractFilePageContentToVariant

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Retrieves the page description operators that define the layout of any page in a PDF document as a variant byte array. This function does not load the entire file into memory so it can be used with arbitrarily large documents.

Syntax

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ExtractFilePageContentToVariant(
  InputFileName As String, Password As String,
  Page As Long) As Variant
```

Parameters

InputFileName	The path and file name of the file to extract page content from.
Password	The password to use when opening the file
Page	The number of the page to extract. The first page in the document is page 1.

ExtractFilePageText

Description

Extracts the text of any page in a PDF file.

This function internally uses the direct access functionality. The entire file is not loaded into memory, so this function can be used on arbitrarily large documents.

Two different methods are provided for extracting text from the selected page in a variety of output formats.

The **DASetTextExtractionWordGap**, **DASetTextExtractionOptions** and **DASetTextExtractionArea** functions can be used to adjust the text extraction process.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ExtractFilePageText(InputFileName,
  Password: WideString; Page, Options: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ExtractFilePageText(
  InputFileName As String, Password As String, Page As Long,
  Options As Long) As String
```

DLL

```
wchar_t * DPLExtractFilePageText(int InstanceID, wchar_t * InputFileName,
  wchar_t * Password, int Page, int Options);
```

Parameters

InputFileName	The path and file name of the file to extract text from.
Password	The password to use, if any, when opening the file
Page	The number of the page that must be extracts. The first page in the document is page 1.
Options	Using the standard text extraction algorithm: 0 = Extract text in human readable format 1 = Deprecated 2 = Return a CSV string including font, color, size and position of each piece of text on the page Using the more accurate but slower text extraction algorithm: 3 = Return a CSV string for each piece of text on the page with the following format: Font Name, Text Color, Text Size, X1, Y1, X2, Y2, X3, Y3, X4, Y4, Text The co-ordinates are the four points bounding the text, measured using the units set with the **SetMeasurementUnits** function and the origin set with the **SetOrigin** function. Co-ordinate order is anti-clockwise with the bottom left corner first. 4 = Similar to option 3, but individual words are returned, making searching for words easier 5 = Similar to option 3 but character widths are output after each block of text 6 = Similar to option 4 but character widths are output after each line of text 7 = Extract text in human readable format with improved accuracy compared to option 0 8 = Similar output format as option 0 but using the more accurate algorithm. Returns unformatted lines.

ExtractFilePageTextBlocks

Version history

This function was introduced in Quick PDF Library version 9.13.

Description

Similar to the **ExtractFilePageText** function but the results are stored in a text block list rather than returned as a CSV string.

This function internally uses the direct access functionality.

Once the results are in the text block list, functions such as **DAGetTextBlockCount**, **DAGetTextBlockText** and **DAGetTextBlockColor** can be used to retrieve the properties of each block of text.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ExtractFilePageTextBlocks(InputFileName,
  Password: WideString; Page, Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ExtractFilePageTextBlocks(
  InputFileName As String, Password As String, Page As Long,
  Options As Long) As Long
```

DLL

```
int DPLExtractFilePageTextBlocks(int InstanceID, wchar_t * InputFileName,
  wchar_t * Password, int Page, int Options);
```

Parameters

InputFileName	The path and file name of the file to extract text from.
Password	The password to use, if any, when opening the file
Page	The number of the page that must be extracts. The first page in the document is page 1.
Options	3 = Normal extraction 4 = Split words

Return values

0	The text could not be extracted
1	A TextBlockListID value

ExtractFilePages

Description

Extracts ranges of pages from a PDF document on disk and places the extracted pages into a new PDF document.

The **ExtractFilePagesEx** function (introduced in version 9.14) is able to produce smaller output files using a cross reference stream instead of a cross reference table.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ExtractFilePages(InputFileName, Password,
  OutputFileName, RangeList: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ExtractFilePages(
  InputFileName As String, Password As String,
  OutputFileName As String, RangeList As String) As Long
```

DLL

```
int DPLExtractFilePages(int InstanceID, wchar_t * InputFileName,
  wchar_t * Password, wchar_t * OutputFileName,
  wchar_t * RangeList);
```

Parameters

InputFileName	The path and name of the document that contains the pages to extract.
Password	The password to use when opening the document
OutputFileName	The path and name of the document to create containing the extracted pages.
RangeList	The pages to extract, for example "10,15,18-20,25-35". Invalid characters will be ignored. Reversed page ranges such as "5-1" will be accepted. Duplicate page numbers will be accepted but if a change is made to such a page the same changes will appear on the duplicate pages. The list of pages will not be sorted so the resulting document will have the pages in the specified order.

Return values

0	The pages could not be extracted. Use the **LastErrorCode** function to determine the cause of the failure.
1	The pages were extracted successfully

ExtractFilePagesEx

Document manipulation, Extraction, Page manipulation

Version history

This function was introduced in Quick PDF Library version 9.14.

Description

Similar to the **ExtractFilePages** function but is able to generate smaller output files using cross reference streams rather than a cross reference table.

Options can be logically OR'd to together to invoke multiple options.

For example Options = 3 (1 + 2) will Use a cross reference stream (smaller output file size) and also Remove all AcroForm and XFA based FormFields as well as Usage Rights

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ExtractFilePagesEx(InputFileName, Password,
  OutputFileName, RangeList: WideString; Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ExtractFilePagesEx(
  InputFileName As String, Password As String,
  OutputFileName As String, RangeList As String,
  Options As Long) As Long
```

DLL

```
int DPLExtractFilePagesEx(int InstanceID, wchar_t * InputFileName,
  wchar_t * Password, wchar_t * OutputFileName,
  wchar_t * RangeList, int Options);
```

Parameters

InputFileName	The path and name of the document that contains the pages to extract.
Password	The password to use when opening the document.
OutputFileName	The path and name of the document to create containing the extracted pages.
RangeList	The pages to extract, for example "10,15,18-20,25-35". Invalid characters will be ignored. Reversed page ranges such as "5-1" will be accepted. Duplicate page numbers will be accepted but if a change is made to such a page the same changes will appear on the duplicate pages. The list of pages will not be sorted so the resulting document will have the pages in the specified order.
Options	0 = Use a cross reference table 1 = Use a cross reference stream (smaller output file size) 2 = Remove all AcroForm and XFA based FormFields as well as Usage Rights

Return values

0	The pages could not be extracted. Use the **LastErrorCode** function to determine the cause of the failure.
1	The pages were extracted successfully

ExtractPageRanges

Document manipulation, Extraction, Page manipulation

Description

Use this function to extract one or more non-consecutive pages from a document to a new document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ExtractPageRanges(
  RangeList: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ExtractPageRanges(
  RangeList As String) As Long
```

DLL

```
int DPLExtractPageRanges(int InstanceID, wchar_t * RangeList);
```

Parameters

RangeList	The pages to extract, for example "10,15,18-20,25-35". Invalid characters and duplicate page numbers in the string will be ignored. Reversed page ranges such as "5-1" will be accepted. The list of pages will be sorted resulting in the pages being extracted in numerical order.

Return values

0	The page extraction did not succeed. The original document remains as the selected document.
1	The page extraction was successful. The new document containing the selected pages is now the selected document.

ExtractPageTextBlocks

Text, Extraction

Version history

This function was introduced in Quick PDF Library version 9.13.

Description

Similar to the **GetPageText** function but the results are stored in a text block list rather than returned as a CSV string.

Once the results are in the text block list, functions such as **GetTextBlockCount**, **GetTextBlockText** and **GetTextBlockColor** can be used to retrieve the properties of each block of text.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ExtractPageTextBlocks(
  ExtractOptions: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ExtractPageTextBlocks(
  ExtractOptions As Long) As Long
```

DLL

```
int DPLExtractPageTextBlocks(int InstanceID, int ExtractOptions);
```

Parameters

ExtractOptions	3 = Normal extraction 4 = Split words

Return values

0	The text could not be extracted
Non-zero	A TextBlockListID value

ExtractPages

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Copies the selected document to a new document, but retains only the specified pages.

If successful, the new document will be selected and the original document will be removed from memory.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ExtractPages(StartPage,
  PageCount: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ExtractPages(StartPage As Long,
  PageCount As Long) As Long
```

DLL

```
int DPLExtractPages(int InstanceID, int StartPage, int PageCount);
```

Parameters

StartPage	The page number of the first page to extract
PageCount	The total number of pages to extract

Return values

0	Failed, use **LastErrorCode** for further details
1	Success

FileListCount

Description

Returns the number of items in the specified file list.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.FileListCount(
  ListName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::FileListCount(
  ListName As String) As Long
```

DLL

```
int DPLFileListCount(int InstanceID, wchar_t * ListName);
```

Parameters

ListName	The name of the file list

FileListItem

Description

Returns the file name stored at the specified index in the named list.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.FileListItem(ListName: WideString;
  Index: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::FileListItem(ListName As String,
  Index As Long) As String
```

DLL

```
wchar_t * DPLFileListItem(int InstanceID, wchar_t * ListName, int Index);
```

Parameters

ListName	The name of the list to work with
Index	The index of the file name to retrieve. The first item has an index of 1.

FindFonts

Description

Analyses the selected document and finds all available fonts. The number of found fonts is returned. Calling this function a second time will return zero as all relevant fonts were found the first time the function was called. These fonts are then available in conjunction to the fonts added with the Add*Font functions and will also be counted in subsequent calls to the **FontCount** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.FindFonts: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::FindFonts As Long
```

DLL

```
int DPLFindFonts(int InstanceID);
```

Return values

0	No fonts were found in the document
Non-zero	The number of fonts that were found

FindFormFieldByTitle

Description

Finds the index of the form field with the specified title.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.FindFormFieldByTitle(
  Title: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::FindFormFieldByTitle(
  Title As String) As Long
```

DLL

```
int DPLFindFormFieldByTitle(int InstanceID, wchar_t * Title);
```

Parameters

Title	The title of the form field to find.

Return values

0	The form field could not be found
Non-zero	The Index of the form field with the specified title

FindImages

Image handling, Document properties

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Searches the selected document for embedded images. This functions searches for image in the Resources dictionary for the entire document. It cannot report where the image was drawn or even if it was drawn at all.

To get the location and number of images draw for each page you will need to use the **GetPageImageList** and related functions.

This function returns the number of images found.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.FindImages: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::FindImages As Long
```

DLL

```
int DPLFindImages(int InstanceID);
```

Return values

0	No images were found
1-n	Number of images found

FitImage

Description

This function allows an image to be placed into an area on the page. The aspect ratio of the image is preserved, and the alignment and rotation of the image can be specified.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.FitImage(Left, Top, Width, Height: Double;
  HAlign, VAlign, Rotate: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::FitImage(Left As Double,
  Top As Double, Width As Double, Height As Double,
  HAlign As Long, VAlign As Long, Rotate As Long) As Long
```

DLL

```
int DPLFitImage(int InstanceID, double Left, double Top, double Width,
  double Height, int HAlign, int VAlign, int Rotate);
```

Parameters

Left	The horizontal co-ordinate of the left-edge of the bounding box
Top	The vertical co-ordinate of the top-edge of the bounding box
Width	The width of the bounding box
Height	The height of the bounding box
HAlign	Horizontal alignment of the image within the bounding box: 0 = Left 1 = Center 2 = Right
VAlign	Vertical alignment of the image within the bounding box: 0 = Top 1 = Center 2 = Bottom
Rotate	The rotation of the image: 0 = Normal 1 = 90 degrees anti-clockwise 2 = 90 degrees clockwise 3 = 180 degrees

Return values

0	The image could not be drawn. Either a valid image has not been selected or the HAlign, VAlign or Rotate parameters are out of range.
1	The image was drawn successfully

FitRotatedTextBox

Description

Similar to the **FitTextBox** function, but the angle of the box can be rotated by any angle. The text size is adjusted to ensure that all the text fits into the available space. The top-left corner of the box before it is rotated is used as the rotation point.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.FitRotatedTextBox(Left, Top, Width, Height,
  Angle: Double; Text: WideString; Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::FitRotatedTextBox(
  Left As Double, Top As Double, Width As Double,
  Height As Double, Angle As Double, Text As String,
  Options As Long) As Long
```

DLL

```
int DPLFitRotatedTextBox(int InstanceID, double Left, double Top,
  double Width, double Height, double Angle, wchar_t * Text,
  int Options);
```

Parameters

Left	The horizontal co-ordinate of the top-left corner of the box before it is rotated
Top	The vertical co-ordinate of the top-left corner of the box before it is rotated
Width	The width of the box before it is rotated
Height	The height of the box before it is rotated
Angle	The angle in degrees that the box should be rotated by. A positive angle rotates the box in an anti-clockwise direction, a negative angle rotated the box in a clockwise direction.
Text	The text that will be fitted into the box
Options	Vertical alignment: 0 = Centered 1 = Top 2 = Bottom If 100 is added to these values long words will not be split up, the font size will be reduced until the longest word fits into the available width. If 1000 is addd to these values the font size will be allowed to increase until the text fills the available area.

Return values

0	The Options parameter was out of range
1	The rotated text box was drawn successfully

FitTextBox

Description

Similar to the **DrawText** function, but the text size is adjusted to ensure that all the text fits into the available space.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.FitTextBox(Left, Top, Width,
  Height: Double; Text: WideString; Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::FitTextBox(Left As Double,
  Top As Double, Width As Double, Height As Double,
  Text As String, Options As Long) As Long
```

DLL

```
int DPLFitTextBox(int InstanceID, double Left, double Top, double Width,
  double Height, wchar_t * Text, int Options);
```

Parameters

Left	The horizontal co-ordinate of the left edge of the bounding box
Top	The vertical co-ordinate of the top edge of the bounding box
Width	The width of the bounding box
Height	The height of the bounding box
Text	The text to display in the box
Options	Vertical alignment: 0 = Centered 1 = Top 2 = Bottom If 100 is added to these values long words will not be split up, the font size will be reduced until the longest word fits into the available width. If 1000 is addd to these values the font size will be allowed to increase until the text fills the available area.

Return values

0	The Options specified were out of range
1	The text was drawn successfully

FlattenAnnot

Version history

This function was introduced in Quick PDF Library version 9.14.

Description

Flattens the specified annotation by merging the appearance stream with the selected page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.FlattenAnnot(Index,
  Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::FlattenAnnot(Index As Long,
  Options As Long) As Long
```

DLL

```
int DPLFlattenAnnot(int InstanceID, int Index, int Options);
```

Parameters

Index	The index of the annotation. The first annotation on the page has an index of 1.
Options	This parameter is reserved for future use and should always be set to zero.

Return values

0	The specified annotation could not be flattened
1	Success

FlattenFormField

Description

Use this function to draw the visual appearance onto the page it is associated with. The form field will then be removed from the document and only it's appearance will remain - it will no longer be an interactive field.

If the field is flattened successfully the field index of subsequent form fields will be decreased by 1.

From version 9.11 this function no longer updates the form field's appearance stream before flattening. To update the appearance stream before flattening, use the **UpdateAndFlattenFormField** function or call **UpdateAppearanceStream** followed by a call to this function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.FlattenFormField(Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::FlattenFormField(
  Index As Long) As Long
```

DLL

```
int DPLFlattenFormField(int InstanceID, int Index);
```

Parameters

Index	The index of the form field to work with. The first form field has an index of 1.

Return values

0	The form field could not be found or it was not possible to flatten the form field
1	The form field was flattened successfully

FontCount

Description

Returns the total number of fonts added to the PDF file. This function does not take into account the fonts that may have already been in an existing PDF document which was loaded with the **LoadFromFile** function unless the **FindFonts** function has been called.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.FontCount: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::FontCount As Long
```

DLL

```
int DPLFontCount(int InstanceID);
```

Return values

0	No fonts have been added to the document or **FindFonts** has not found any fonts in an existing document.
Non-zero	The number of fonts added to the PDF plus the number of fonts found with **FindFonts**.

FontFamily

Version history

This function was introduced in Quick PDF Library version 7.16.

Description

Returns the font family of the selected font, if available.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.FontFamily: WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::FontFamily As String
```

DLL

```
wchar_t * DPLFontFamily(int InstanceID);
```

FontHasKerning

Description

Indicated whether the selected font has kerning information.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.FontHasKerning: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::FontHasKerning As Long
```

DLL

```
int DPLFontHasKerning(int InstanceID);
```

Return values

0	The selected font does not have any kerning information
1	The selected font has at least one kerning pair

FontName

Description

Returns the name of the selected font. A font is automatically selected when it is added to the document. The **GetFontID** and **SelectFont** functions can be used to select a different font.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.FontName: WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::FontName As String
```

DLL

```
wchar_t * DPLFontName(int InstanceID);
```

FontReference

Description

Returns the internal reference of the selected font.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.FontReference: WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::FontReference As String
```

DLL

```
wchar_t * DPLFontReference(int InstanceID);
```

FontSize

Description

Returns the size in bytes of the selected font. A value will only be returned for embedded TrueType or Type1 fonts. A value will not be returned for subsetted fonts or standard fonts.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.FontSize: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::FontSize As Long
```

DLL

```
int DPLFontSize(int InstanceID);
```

FontType

Description

Used to determine the type of the selected font.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.FontType: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::FontType As Long
```

DLL

```
int DPLFontType(int InstanceID);
```

Return values

0	No font has been selected
1	Unknown
2	Standard
3	TrueType
4	Embedded TrueType
5	Packaged
6	Type1
7	Subsetted
8	Type3
9	Type1 CID
10	TrueType CID
11	CJK

FormFieldCount

Description

Returns the total number of form fields in the selected document. The Index parameter of the various form field functions must be a number from 1 to the value returned by this function.

If a form field is deleted or flattened successfully it will be removed from the document, the total field count will be reduced and the field Index of the subsequent fields will be reduced by 1.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.FormFieldCount: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::FormFieldCount As Long
```

DLL

```
int DPLFormFieldCount(int InstanceID);
```

Return values

0	There are no form fields
Non-zero	The number of form fields in the document

FormFieldHasParent

Description

This function returns 1 if the specified form field is the child of another field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.FormFieldHasParent(Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::FormFieldHasParent(
  Index As Long) As Long
```

DLL

```
int DPLFormFieldHasParent(int InstanceID, int Index);
```

Parameters

Index	The index of the form field. The first field has an index of 1.

FormFieldJavaScriptAction

Form fields, JavaScript

Description

Adds JavaScript to a form field for any of the possible action types.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.FormFieldJavaScriptAction(Index: Integer;
  ActionType, JavaScript: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::FormFieldJavaScriptAction(
  Index As Long, ActionType As String,
  JavaScript As String) As Long
```

DLL

```
int DPLFormFieldJavaScriptAction(int InstanceID, int Index,
  wchar_t * ActionType, wchar_t * JavaScript);
```

Parameters

Index	Index of the form field
ActionType	The action type: E = An action to be performed when the cursor enters the annotation's active area X = An action to be performed when the cursor exits the annotation's active area D = An action to be performed when the mouse button is pressed inside the annotation's active area U = An action to be performed when the mouse button is released inside the annotation's active area Fo = An action to be performed when the annotation receives the input focus Bl = An action to be performed when the annotation loses the input focus (blurred) K = An action to be performed when the user types a keystroke into a text field or combo box or modifies the selection in a scrollable list box. This allows the keystroke to be checked for validity and rejected or modified. F = An action to be performed before the field is formatted to display its current value. This allows the field's value to be modified before formatting. V = An action to be performed when the field's value is changed. This allows the new value to be checked for validity. C = An action to be performed in order to recalculate the value of this field when that of another field changes
JavaScript	The JavaScript to execute.

Return values

0	Cannot find the form field
1	The JavaScript action was added to the form field successfully

FormFieldWebLinkAction

Form fields

Description

Adds an action to the specified form field that links to an internet address.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.FormFieldWebLinkAction(Index: Integer;
  ActionType, Link: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::FormFieldWebLinkAction(
  Index As Long, ActionType As String, Link As String) As Long
```

DLL

```
int DPLFormFieldWebLinkAction(int InstanceID, int Index,
  wchar_t * ActionType, wchar_t * Link);
```

Parameters

Index	The index of the form field to set the action of
ActionType	The action type: E = An action to be performed when the cursor enters the annotation's active area X = An action to be performed when the cursor exits the annotation's active area D = An action to be performed when the mouse button is pressed inside the annotation's active area U = An action to be performed when the mouse button is released inside the annotation's active area Fo = An action to be performed when the annotation receives the input focus Bl = An action to be performed when the annotation loses the input focus (blurred) K = An action to be performed when the user types a keystroke into a text field or combo box or modifies the selection in a scrollable list box. This allows the keystroke to be checked for validity and rejected or modified. F = An action to be performed before the field is formatted to display its current value. This allows the field's value to be modified before formatting. V = An action to be performed when the field's value is changed. This allows the new value to be checked for validity. C = An action to be performed in order to recalculate the value of this field when that of another field changes
Link	The URL to link to. Some examples: "http://www.example.com" "mailto:info@example.com"

Return values

0	The form field could not be found, or the ActionType was invalid
1	The web link action was added to the form field successfully

GetActionDest

Version history

This function was introduced in Quick PDF Library version 7.21.

Description

This function will return a DestID if the specified action has a destination entry. The DestID can be used with the **GetDestPage**, **GetDestType** and **GetDestValue** functions.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetActionDest(ActionID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetActionDest(
  ActionID As Long) As Long
```

DLL

```
int DPLGetActionDest(int InstanceID, int ActionID);
```

Parameters

ActionID	An ActionID as returned by the **GetAnnotActionID**, **GetOutlineActionID** or **GetFormFieldActionID** functions

Return values

0	The specified action does not have a destination entry
Non-zero	A DestID that can be used with the destination functions.

GetActionType

Version history

This function was introduced in Quick PDF Library version 7.16.

Description

Returns the action type of the specified action, for example "GoTo" or "GoToR".

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetActionType(
  ActionID: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetActionType(
  ActionID As Long) As String
```

DLL

```
wchar_t * DPLGetActionType(int InstanceID, int ActionID);
```

Parameters

ActionID	An ActionID as returned by the **GetAnnotActionID**, **GetOutlineActionID** or **GetFormFieldActionID** functions

GetActionURL

Version history

This function was introduced in Quick PDF Library version 7.21.

Description

Returns the target URL of the specified action.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetActionURL(ActionID: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetActionURL(
  ActionID As Long) As String
```

DLL

```
wchar_t * DPLGetActionURL(int InstanceID, int ActionID);
```

Parameters

ActionID	An ActionID as returned by the **GetAnnotActionID**, **GetOutlineActionID** or **GetFormFieldActionID** functions

GetAnalysisInfo

Document properties

Description

Returns individual items from the results of the analysis done by the **AnalyseFile** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetAnalysisInfo(AnalysisID,
  AnalysisItem: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetAnalysisInfo(
  AnalysisID As Long, AnalysisItem As Long) As String
```

DLL

```
wchar_t * DPLGetAnalysisInfo(int InstanceID, int AnalysisID,
  int AnalysisItem);
```

Parameters

AnalysisID	The ID of the set of analysis results to query, as returned by the **AnalyseFile** function
AnalysisItem	The specific analysis result to retrieve: 0 = File name (eg. "c:\hello.pdf") 1 = File size (eg. "2048" for a file exactly 2K in size) 2 = Author 3 = Title 4 = Subject 5 = Keywords 6 = Creator 7 = Producer 8 = PDF version (eg. "1.4") 9 = Page count (eg. "120") 10 = Creation date 11 = Modification date 12 = Document ID 13 = The supplied password: "None" for no security "User" for the user password "Owner" for the owner password 14 = Document contains usage rights (eg. Reader Extensions) "No" if there is no usage rights dictionary "Yes" if there is a usage rights dictionary 15 = Name of signature in the usage rights dictionary 20..30 = Equivalent to **SecurityInfo**(0)..**SecurityInfo**(10) 31 = Number of form fields in the document 41..43 = Equivalent to **SecurityInfo**(11)..**SecurityInfo**(13)

GetAnnotActionID

Version history

This function was introduced in Quick PDF Library version 7.16.

Description

This function will return an ActionID if the specified annotation has an action dictionary.
The ActionID can be used with the **GetActionType** and **GetActionDest** functions and can also be compared to the values returned by **GetOutlineActionID** to determine if an annotation action is shared with an outline action.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetAnnotActionID(Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetAnnotActionID(
  Index As Long) As Long
```

DLL

```
int DPLGetAnnotActionID(int InstanceID, int Index);
```

Parameters

Index	The index of the annotation. The first annotation on the page has an index of 1.

GetAnnotDblProperty

Description

Returns a property of the specified annotation.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetAnnotDblProperty(Index,
  Tag: Integer): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetAnnotDblProperty(
  Index As Long, Tag As Long) As Double
```

DLL

```
double DPLGetAnnotDblProperty(int InstanceID, int Index, int Tag);
```

Parameters

Index	The index of the annotation. The first annotation on the page has an index of 1.
Tag	105 = Left 106 = Top 107 = Width 108 = Height 119 = Gray color component 120 = Red color component 121 = Green color component 122 = Blue color component 123 = Cyan color component 124 = Magenta color component 125 = Yellow color component 126 = Black color component 132 = Border width

GetAnnotDest

Version history

This function was introduced in Quick PDF Library version 7.21.

Description

This function will return a DestID if the specified annotation has a destination entry. The DestID can be used with the **GetDestPage**, **GetDestType** and **GetDestValue** functions.

If the annotation does not have a destination entry, this function will return zero.

The **GetAnnotActionID** function might return a value that can be used with the **GetActionDest** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetAnnotDest(Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetAnnotDest(
  Index As Long) As Long
```

DLL

```
int DPLGetAnnotDest(int InstanceID, int Index);
```

Parameters

Index	The index of the annotation. The first annotation on the page has an index of 1.

Return values

0	The specified annotation does not have a destination entry.
Non-zero	A DestID that can be used with the destination functions.

GetAnnotEmbeddedFileName

Version history

This function was introduced in Quick PDF Library version 10.13.

Description

Returns the filename of the embedded attachment that is stored in this annotation object

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetAnnotEmbeddedFileName(Index,
  Options: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetAnnotEmbeddedFileName(
  Index As Long, Options As Long) As String
```

DLL

```
wchar_t * DPLGetAnnotEmbeddedFileName(int InstanceID, int Index,
  int Options);
```

Parameters

Index	The index of the annotation. The first annotation on the page has an index of 1.
Options	Currently not used. Default = 0

GetAnnotEmbeddedFileToFile

Version history

This function was introduced in Quick PDF Library version 10.13.

Description

Saves the embedded file inside the annotation object to the specified file on disk.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetAnnotEmbeddedFileToFile(Index,
  Options: Integer; FileName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetAnnotEmbeddedFileToFile(
  Index As Long, Options As Long, FileName As String) As Long
```

DLL

```
int DPLGetAnnotEmbeddedFileToFile(int InstanceID, int Index, int Options,
  wchar_t * FileName);
```

Parameters

Index	The index of the annotation. The first annotation on the page has an index of 1.
Options	Currently not used. Default = 0
FileName	The filename of where to save the file

GetAnnotEmbeddedFileToString

Version history

This function was introduced in Quick PDF Library version 10.13.

Description

Returns the embedded file inside the annotation object as a string.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetAnnotEmbeddedFileToString(Index,
  Options: Integer): AnsiString;
```

DLL

```
char * DPLGetAnnotEmbeddedFileToString(int InstanceID, int Index,
  int Options);
```

Parameters

Index	The index of the annotation. The first annotation on the page has an index of 1.
Options	Currently not used. Default = 0

GetAnnotIntProperty

Description

Returns a property of the specified annotation.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetAnnotIntProperty(Index,
  Tag: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetAnnotIntProperty(
  Index As Long, Tag As Long) As Long
```

DLL

```
int DPLGetAnnotIntProperty(int InstanceID, int Index, int Tag);
```

Parameters

Index	The index of the annotation. The first annotation on the page has an index of 1.
Tag	109 = Flags 116 = Page number of "GoToR" action (1 is first page) 128 = Index of the annotation that this annotation is in reply to 131 = Page number of "GoTo" action 133 = Returns 1 if a "Launch" or "GoToR" action's NewWindow property is set

GetAnnotQuadCount

Version history

This function was introduced in Quick PDF Library version 7.21.

Description

Returns the number of quads (rectangular areas) within the specified annotation.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetAnnotQuadCount(Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetAnnotQuadCount(
  Index As Long) As Long
```

DLL

```
int DPLGetAnnotQuadCount(int InstanceID, int Index);
```

Parameters

Index	The index of the annotation. The first annotation on the page has an index of 1.

GetAnnotQuadPoints

Annotations and hotspot links

Version history

> This function was introduced in Quick PDF Library version 7.21.

Description

> Returns a component of the specified quad (rectangular area) contained within the specified annotation.

> From version 7.25 the order of the co-ordinates has changed for consistency between **GetPageText** and **SetAnnotQuadPoints**.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetAnnotQuadPoints(Index, QuadNumber,
  PointNumber: Integer): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetAnnotQuadPoints(
  Index As Long, QuadNumber As Long,
  PointNumber As Long) As Double
```

DLL

```
double DPLGetAnnotQuadPoints(int InstanceID, int Index, int QuadNumber,
  int PointNumber);
```

Parameters

Index	The index of the annotation. The first annotation on the page has an index of 1.
QuadNumber	The number of the quad to access. The first quad has a QuadNumber of 1.
PointNumber	1 = The horizontal co-ordinate of the bottom-left corner 2 = The vertical co-ordinate of the bottom-left corner 3 = The horizontal co-ordinate of the bottom-right corner 4 = The vertical co-ordinate of the bottom-right corner 5 = The horizontal co-ordinate of the top-right corner 6 = The vertical co-ordinate of the top-right corner 7 = The horizontal co-ordinate of the top-left corner 8 = The vertical co-ordinate of the top-left corner

GetAnnotSoundToFile

Version history

This function was introduced in Quick PDF Library version 9.14.

Description

Copies the sound data stored in the specified annotation into a file.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetAnnotSoundToFile(Index,
  Options: Integer; SoundFileName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetAnnotSoundToFile(
  Index As Long, Options As Long,
  SoundFileName As String) As Long
```

DLL

```
int DPLGetAnnotSoundToFile(int InstanceID, int Index, int Options,
  wchar_t * SoundFileName);
```

Parameters

Index	The index of the annotation. The first annotation on the page has an index of 1.
Options	0 = Sound data as stored in the PDF 1 = Encode data as a WAV file
SoundFileName	The path and name of the file to create containing the sound data.

Return values

0	The sound could not be written
1	The sound was written successfully

GetAnnotSoundToString

Version history

This function was introduced in Quick PDF Library version 9.14.

Description

Returns the sound data stored in the specified annotation.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetAnnotSoundToString(Index,
  Options: Integer): AnsiString;
```

DLL

```
char * DPLGetAnnotSoundToString(int InstanceID, int Index, int Options);
```

Parameters

Index	The index of the annotation. The first annotation on the page has an index of 1.
Options	0 = Sound data as stored in the PDF 1 = Encode data as a WAV file

GetAnnotStrProperty

Annotations and hotspot links

Version history

This function was introduced in Quick PDF Library version 7.15.

Description

Returns a property of the specified annotation.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetAnnotStrProperty(Index,
   Tag: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetAnnotStrProperty(
   Index As Long, Tag As Long) As String
```

DLL

```
wchar_t * DPLGetAnnotStrProperty(int InstanceID, int Index, int Tag);
```

Parameters

Index	The index of the annotation. The first annotation on the page has an index of 1.
Tag	101 = Annotation type 102 = Contents 103 = Name 104 = Modified date 110 = Author 111 = URL of a link annotation 112 = Action type of link annotation, eg. "URI", "Launch", "GoToR" 113 = The "Win" file name of a "Launch" action 114 = The "F" file name of a "Launch" action 115 = The "F" file name of a "GoToR" action 117 = The name of the annotation icon 118 = Color space, eg. "Gray", "RGB", "CMYK" 127 = Subject of the annotation 129 = The "UF" file name of a "Launch" action 130 = The "UF" file name of a "GoToR" action 140 = The "OverlayText" of the Radact annotation object

GetBarcodeWidth

Vector graphics, Page layout

Description

Returns the total width of a barcode based on the width of the smallest bars in the barcode.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetBarcodeWidth(NominalWidth: Double;
  Text: WideString; Barcode: Integer): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetBarcodeWidth(
  NominalWidth As Double, Text As String,
  Barcode As Long) As Double
```

DLL

```
double DPLGetBarcodeWidth(int InstanceID, double NominalWidth,
  wchar_t * Text, int Barcode);
```

Parameters

NominalWidth	The desired width of the narrowest bars in the barcode
Text	The barcode data
Barcode	1 = Code39 (or Code 3 of 9) 2 = EAN-13 3 = Code128 4 = PostNet 5 = Interleaved 2 of 5

GetBaseURL

Version history

This function was introduced in Quick PDF Library version 7.26.

Description

Returns the Base URL for all URL links in the document.

For example, if the Base URL was set to "http://www.example.com/" and a URL link destination was set to "index.html" then the link will point to "http://www.example.com/index.html".

Use the **AddLinkToWeb** function to add a URL link to the current page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetBaseURL: WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetBaseURL As String
```

DLL

```
wchar_t * DPLGetBaseURL(int InstanceID);
```

GetCSDictEPSG

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Returns the EPSG reference code for a coordinate system dictionary (see www.epsg.org).

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetCSDictEPSG(CSDictID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetCSDictEPSG(
  CSDictID As Long) As Long
```

DLL

```
int DPLGetCSDictEPSG(int InstanceID, int CSDictID);
```

Parameters

CSDictID	A value returned from the **GetMeasureDictGCSDict** or **GetMeasureDictDCSDict** functions

GetCSDictType

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Returns the coordinate system type for a coordinate system dictionary.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetCSDictType(CSDictID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetCSDictType(
  CSDictID As Long) As Long
```

DLL

```
int DPLGetCSDictType(int InstanceID, int CSDictID);
```

Parameters

CSDictID	A value returned from the **GetMeasureDictGCSDict** or **GetMeasureDictDCSDict** functions

Return values

0	The CSDictID parameter was incorrect
1	A geographic coordinate system (GEOGCS)
2	A projected coordinate system (PROJCS)

GetCSDictWKT

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Returns the Well Known Text (WKT) description of a coordinate system dictionary.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetCSDictWKT(CSDictID: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetCSDictWKT(
  CSDictID As Long) As String
```

DLL

```
wchar_t * DPLGetCSDictWKT(int InstanceID, int CSDictID);
```

Parameters

CSDictID	A value returned from the **GetMeasureDictGCSDict** or **GetMeasureDictDCSDict** functions

GetCanvasDC

Version history

This function was introduced in Quick PDF Library version 7.14.

Description

Creates a canvas of the specified size and returns a Windows device context DC that can be drawn on using Win32 drawing commands. When drawing operations are complete, call the **LoadFromCanvasDC** function to create a new document from the supplied drawing commands.

The return value is defined as either an unsigned integer or a signed integer on different platforms and editions of the library.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetCanvasDC(CanvasWidth,
  CanvasHeight: Integer): HDC;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetCanvasDC(CanvasWidth As Long,
  CanvasHeight As Long) As Long
```

DLL

```
HDC DPLGetCanvasDC(int InstanceID, int CanvasWidth, int CanvasHeight);
```

Parameters

CanvasWidth	The width of the canvas
CanvasHeight	The height of the canvas

GetCanvasDCEx

Vector graphics, Document management

Version history

This function was introduced in Quick PDF Library version 10.15.

Description

Creates a canvas of the specified size and returns a Windows device context DC that can be drawn on using Win32 drawing commands. When drawing operations are complete, call the **LoadFromCanvasDC** function to create a new document from the supplied drawing commands.

The Ex version of the function allows you to pass an existing Device Context handle as a reference when creating the DC.

The return value is defined as either an unsigned integer or a signed integer on different platforms and editions of the library.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetCanvasDCEx(CanvasWidth, CanvasHeight,
  ReferenceDC: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetCanvasDCEx(
  CanvasWidth As Long, CanvasHeight As Long,
  ReferenceDC As Long) As Long
```

DLL

```
int DPLGetCanvasDCEx(int InstanceID, int CanvasWidth, int CanvasHeight,
  int ReferenceDC);
```

Parameters

CanvasWidth	The width of the canvas
CanvasHeight	The height of the canvas
ReferenceDC	The reference device context handle

GetCatalogInformation

Description

This function allows you to retrieve custom information from the "Catalog" section of the document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetCatalogInformation(
  Key: WideString): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetCatalogInformation(
  Key As String) As String
```

DLL

```
wchar_t * DPLGetCatalogInformation(int InstanceID, wchar_t * Key);
```

Parameters

Key	The name of the key to retrieve. This key must have a special prefix assigned to you by Adobe to avoid conflicts with other software.

GetContentStreamToString

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Returns the PDF page description commands in the content stream part that was selected with the **SelectContentStream** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetContentStreamToString: AnsiString;
```

DLL

```
char * DPLGetContentStreamToString(int InstanceID);
```

GetContentStreamToVariant

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Returns the PDF page description commands in the content stream part that was selected with the **SelectContentStream** function. The data is returned as a variant byte array.

Syntax

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetContentStreamToVariant As Variant
```

GetCustomInformation

Description

Returns a custom value from the document. This function and the **SetCustomInformation** function can be used to store and retrieve custom document metadata.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetCustomInformation(
   Key: WideString): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetCustomInformation(
   Key As String) As String
```

DLL

```
wchar_t * DPLGetCustomInformation(int InstanceID, wchar_t * Key);
```

Parameters

Key	Specifies which key to retrieve the value of

Return values

The value of the specified key, or an empty string if the key could not be found. An empty string will also be returned if the key is "Author", "Keywords", "Subject", "Title", "Creator" or "Producer". For these keys, use the **GetInformation function**.

GetCustomKeys

Description

Returns all the custom keys in either the Document Information Dictionary or Document Catalog as a CSV string. See the SetCustomInformation and GetCustomInformation functions for details of how to manipulate the data stored under these keys.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetCustomKeys(
  Location: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetCustomKeys(
  Location As Long) As String
```

DLL

```
wchar_t * DPLGetCustomKeys(int InstanceID, int Location);
```

Parameters

Location	The location to extract custom key names from: 1 = Document Information Dictionary 2 = Document Catalog

GetDefaultPrinterName

Description

Returns the name of the default printer. This name can be used with the **PrintDocument** or **NewCustomPrinter** functions.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetDefaultPrinterName: WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetDefaultPrinterName As String
```

DLL

```
wchar_t * DPLGetDefaultPrinterName(int InstanceID);
```

GetDestName

Version history

This function was introduced in Quick PDF Library version 7.22.

Description

Returns the name of the specified destination.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetDestName(DestID: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetDestName(
  DestID As Long) As String
```

DLL

```
wchar_t * DPLGetDestName(int InstanceID, int DestID);
```

Parameters

DestID	The ID of the destination to analyse. A valid destination ID is returned by the **GetOutlineDest** function.

GetDestPage

Description

Returns the page number of the specified destination, or zero if the destination is invalid or does not contain a page number.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetDestPage(DestID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetDestPage(
  DestID As Long) As Long
```

DLL

```
int DPLGetDestPage(int InstanceID, int DestID);
```

Parameters

DestID	The ID of the destination to analyse. A valid destination ID is returned by the **GetOutlineDest** function.

GetDestType

Description

Returns the type of the specified destination.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetDestType(DestID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetDestType(
  DestID As Long) As Long
```

DLL

```
int DPLGetDestType(int InstanceID, int DestID);
```

Parameters

DestID	The ID of the destination to analyse. A valid destination ID is returned by the **GetOutlineDest** function.

Return values

1	"XYZ" - the target page is positioned at the Left and Top properties of the destination, and the Zoom property specifies the zoom percentage
2	"Fit" - the entire page is zoomed to fit the window
3	"FitH" - the page is zoomed so that the entire width of the page is visible. The height of the page may be greater or less than the height of the window. The page is positioned vertically at the Top property of the destination.
4	"FitV" - the page is zoomed so that the entire height of the page can be seen. The width of the page may be greater or less than the width of the window. The page is positioned horizontally at the Left property of the destination.
5	"FitR" - the page is zoomed so that a certain rectangle on the page is visible. The Left, Top, Right and Bottom properties of the destination define the rectangle on the page.
6	"FitB" - the page is zoomed so that it's bounding box is visible
7	"FitBH" - the page is positioned vertically at the value of the Top property of the destination, and the page is zoomed so that the entire width of the page's bounding box is visible
8	"FitBV" - the page is positioned at the value of the Left property of the destination is visible, and the page is zoomed just enough to fit the entire height of the bounding box into the window

GetDestValue

Annotations and hotspot links

Description

Returns the value of a property of the specified destination.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetDestValue(DestID,
  ValueKey: Integer): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetDestValue(DestID As Long,
  ValueKey As Long) As Double
```

DLL

```
double DPLGetDestValue(int InstanceID, int DestID, int ValueKey);
```

Parameters

DestID	The ID of the destination to analyse. A valid destination ID is returned by the **GetOutlineDest** function.
ValueKey	1 = Left 2 = Top 3 = Right 4 = Bottom 5 = Zoom

GetDocJavaScript

Description

Retrieves the JavaScript linked to a specified document action.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetDocJavaScript(
  ActionType: WideString): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetDocJavaScript(
  ActionType As String) As String
```

DLL

```
wchar_t * DPLGetDocJavaScript(int InstanceID, wchar_t * ActionType);
```

Parameters

ActionType	Retrieve the JavaScript linked to this action: "DC" = Document close "WS" = Will save "DS" = Did save "WP" = Will print "DP" = Did print

GetDocumentFileName

Version history

This function was introduced in Quick PDF Library version 7.17.

Description

Returns the file name of the selected document if it was opened using **LoadFromFile**.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetDocumentFileName: WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetDocumentFileName As String
```

DLL

```
wchar_t * DPLGetDocumentFileName(int InstanceID);
```

GetDocumentFileSize

Version history

This function was introduced in Quick PDF Library version 7.21.

Description

Returns the file size of the selected document.

The size cannot be determined dynamically - it will only be set directly after a call to **LoadFromFile**, **LoadFromStream**, **LoadFromString**, **LoadFromVariant**, **SaveToFile**, **SaveToStream**, **SaveToString** or **SaveToVariant**.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetDocumentFileSize: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetDocumentFileSize As Long
```

DLL

```
int DPLGetDocumentFileSize(int InstanceID);
```

GetDocumentID

Version history

This function was renamed in Quick PDF Library version 7.11.
The function name in earlier versions was DocumentID.

Description

Returns the ID of the document with the specified index.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetDocumentID(Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetDocumentID(
  Index As Long) As Long
```

DLL

```
int DPLGetDocumentID(int InstanceID, int Index);
```

Parameters

Index	The index of the document to query. Must be 1 or greater.

Return values

0	The specified index was out of range
Non-zero	The ID of the specified document

GetDocumentIdentifier

Description

Returns the document identifier. This identifier consists of two parts, each strings. The first string does not change when the document is resaved with an "incremental update" in Acrobat. This can be seen as the permanent identifier for the document. The second part will change each time the document is resaved, even if the resave is an incremental update.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetDocumentIdentifier(Part,
  Options: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetDocumentIdentifier(
  Part As Long, Options As Long) As String
```

DLL

```
wchar_t * DPLGetDocumentIdentifier(int InstanceID, int Part, int Options);
```

Parameters

Part	0 = Permanent identifier 1 = Changeable identifier
Options	0 = Return the identifier as a string of characters 1 = Return the identifier as a hexadecimal string

GetDocumentMetadata

Description

Returns the document's metadata, if any.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetDocumentMetadata: WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetDocumentMetadata As String
```

DLL

```
wchar_t * DPLGetDocumentMetadata(int InstanceID);
```

GetDocumentRepaired

Version history

This function was introduced in Quick PDF Library version 9.11.

Description

Indicates whether the document was repaired when it was loaded.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetDocumentRepaired: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetDocumentRepaired As Long
```

DLL

```
int DPLGetDocumentRepaired(int InstanceID);
```

Return values

0	The document was not repaired
1	The document was repaired

GetDocumentResourceList

Version history

This function was introduced in Quick PDF Library version 7.11.

Description

Returns a list of the PDF resource names used in the document. For advanced use only.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetDocumentResourceList: WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetDocumentResourceList As String
```

DLL

```
wchar_t * DPLGetDocumentResourceList(int InstanceID);
```

GetEmbeddedFileContentToFile

Version history

This function was introduced in Quick PDF Library version 7.13.

Description

Extracts the specified embedded file and writes the content to the specified file.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetEmbeddedFileContentToFile(
  Index: Integer; FileName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetEmbeddedFileContentToFile(
  Index As Long, FileName As String) As Long
```

DLL

```
int DPLGetEmbeddedFileContentToFile(int InstanceID, int Index,
  wchar_t * FileName);
```

Parameters

Index	The index of the embedded file. Must be a value between 1 and the value returned by **EmbeddedFileCount**.
FileName	The path and file name of the file to write the contents to.

Return values

0	Could not write to the specified file or Index parameter was invalid.
1	Embedded file contents written to the specified file successfully.

GetEmbeddedFileContentToStream

Version history

This function was introduced in Quick PDF Library version 7.13.

Description

Extracts the specified embedded file and writes the content to the specified stream.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetEmbeddedFileContentToStream(
  Index: Integer; OutStream: TStream): Integer;
```

Parameters

Index	The index of the embedded file. Must be a value between 1 and the value returned by **EmbeddedFileCount**.
OutStream	The TStream object to write the contents to

Return values

0	Could not write to the specified stream or Index parameter was invalid.
1	Success

GetEmbeddedFileContentToString

Version history

This function was introduced in Quick PDF Library version 7.13.

Description

Extracts the specified embedded file and returns the content as a string.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetEmbeddedFileContentToString(
  Index: Integer): AnsiString;
```

DLL

```
char * DPLGetEmbeddedFileContentToString(int InstanceID, int Index);
```

Parameters

Index	The index of the embedded file. Must be a value between 1 and the value returned by **EmbeddedFileCount**.

GetEmbeddedFileContentToVariant

Version history

This function was introduced in Quick PDF Library version 7.13.

Description

Extracts the specified embedded file and returns the content as a byte array variant.

Syntax

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetEmbeddedFileContentToVariant(
  Index As Long) As Variant
```

Parameters

Index	The index of the embedded file. Must be a value between 1 and the value returned by **EmbeddedFileCount**.

GetEmbeddedFileID

Version history

This function was introduced in Quick PDF Library version 7.25.

Description

Returns the ID of the specified embedded file. This ID can be used with the **AddLinkToEmbeddedFile** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetEmbeddedFileID(Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetEmbeddedFileID(
  Index As Long) As Long
```

DLL

```
int DPLGetEmbeddedFileID(int InstanceID, int Index);
```

Parameters

Index	The index of the embedded file. Must be a value between 1 and the value returned by **EmbeddedFileCount**.

Return values

0	The specified index was invalid
Non-zero	An EmbeddedFileID value

GetEmbeddedFileIntProperty

Version history

This function was introduced in Quick PDF Library version 7.13.

Description

Retrieves an integer property of the specified embedded file.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetEmbeddedFileIntProperty(Index,
  Tag: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetEmbeddedFileIntProperty(
  Index As Long, Tag As Long) As Long
```

DLL

```
int DPLGetEmbeddedFileIntProperty(int InstanceID, int Index, int Tag);
```

Parameters

Index	The index of the embedded file. Must be a value between 1 and the value returned by **EmbeddedFileCount**.
Tag	5 = Deprecated (previously same as 6) 6 = File size in bytes

GetEmbeddedFileStrProperty

Version history

This function was introduced in Quick PDF Library version 7.13.

Description

Retrieves a string property of the specified embedded file.

Use the **SetEmbeddedFileStrProperty** function to change the values.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetEmbeddedFileStrProperty(Index,
  Tag: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetEmbeddedFileStrProperty(
  Index As Long, Tag As Long) As String
```

DLL

```
wchar_t * DPLGetEmbeddedFileStrProperty(int InstanceID, int Index,
  int Tag);
```

Parameters

Index	The index of the embedded file. Must be a value between 1 and the value returned by **EmbeddedFileCount**.
Tag	1 = File name 2 = MIME type 3 = Creation date 4 = Modification date 5 = Title 7 = Description

GetEncryptionFingerprint

Description

Returns all the encryption information for the selected document. This encryption "fingerprint" can be used to encrypt a different document using the **EncryptWithFingerprint** function. This allows a new document to be encrypted with the same passwords as an existing document without actually knowing these passwords.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetEncryptionFingerprint: WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetEncryptionFingerprint As String
```

DLL

```
wchar_t * DPLGetEncryptionFingerprint(int InstanceID);
```

GetFileMetadata

Description

Returns the metadata in a file, if any.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFileMetadata(InputFileName,
  Password: WideString): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFileMetadata(
  InputFileName As String, Password As String) As String
```

DLL

```
wchar_t * DPLGetFileMetadata(int InstanceID, wchar_t * InputFileName,
  wchar_t * Password);
```

Parameters

InputFileName	The path and name of the document to extract metadata from.
Password	The password to use when opening the document

GetFirstChildOutline

Description

Returns the ID of the outline that is the first child of the specified outline.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFirstChildOutline(
  OutlineID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFirstChildOutline(
  OutlineID As Long) As Long
```

DLL

```
int DPLGetFirstChildOutline(int InstanceID, int OutlineID);
```

Parameters

OutlineID	The ID of the outline item to work with. This ID is returned by the **NewOutline** or **NewStaticOutline** functions, or retrieved with the **GetOutlineID** function or Get*Outline functions.

GetFirstOutline

Description

Returns the ID of the first outline in the hierarchy.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFirstOutline: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFirstOutline As Long
```

DLL

```
int DPLGetFirstOutline(int InstanceID);
```

GetFontEncoding

Version history

This function was introduced in Quick PDF Library version 7.22.

Description

Returns the font encoding of the selected font.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFontEncoding: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFontEncoding As Long
```

DLL

```
int DPLGetFontEncoding(int InstanceID);
```

Return values

0	Unknown
1	MacRomanEncoding
2	WinAnsiEncoding
3	MacExpertEncoding
5	No encoding

GetFontFlags

Version history

This function was introduced in Quick PDF Library version 9.14.

Description

Returns the value of the specified bit in the flags property of the selected font.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFontFlags(
  FontFlagItemID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFontFlags(
  FontFlagItemID As Long) As Long
```

DLL

```
int DPLGetFontFlags(int InstanceID, int FontFlagItemID);
```

Parameters

FontFlagItemID	1 = Fixed
	2 = Serif
	3 = Symbolic
	4 = Script
	5 = Italic
	6 = AllCap
	7 = SmallCap
	8 = ForceBold

Return values

0	Flag is not set
1	Flag is set

GetFontID

Version history

This function was renamed in Quick PDF Library version 7.11.
The function name in earlier versions was FontID.

Description

Returns the ID of the specified font. Before this function is used a call to **FindFonts** or **FontCount** must be made in order to generate a list of fonts available for use in the selected document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFontID(Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFontID(Index As Long) As Long
```

DLL

```
int DPLGetFontID(int InstanceID, int Index);
```

Parameters

Index	The index of the font. The first font has an index of 1.

Return values

0	The index is out of bounds
Non-zero	The ID of the specified font

GetFontIsEmbedded

Description

This function will return 1 if the font is an embedded font

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFontIsEmbedded: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFontIsEmbedded As Long
```

DLL

```
int DPLGetFontIsEmbedded(int InstanceID);
```

Return values

1	Font is embedded
0	Font is not embedded

GetFontIsSubsetted

Description

This function will return 1 if the font is a subsetted font.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFontIsSubsetted: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFontIsSubsetted As Long
```

DLL

```
int DPLGetFontIsSubsetted(int InstanceID);
```

Return values

1	Font is subsetted
0	Font is not subsetted

GetFontMetrics

Description

Gets selected font parameters

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFontMetrics(
  MetricType: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFontMetrics(
  MetricType As Long) As Long
```

DLL

```
int DPLGetFontMetrics(int InstanceID, int MetricType);
```

Parameters

MetricType	1: FontAscent,
	2: FontDescent,
	3: FontInternalLeading,
	4: FontExternalLeading,
	5: EM Square,
	6: Average char width

GetFontObjectNumber

Version history

This function was introduced in Quick PDF Library version 7.14.

Description

This specialized function returns the internal object number of the selected font. This object number can sometimes be used by other systems.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFontObjectNumber: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFontObjectNumber As Long
```

DLL

```
int DPLGetFontObjectNumber(int InstanceID);
```

GetFormFieldActionID

Version history

This function was introduced in Quick PDF Library version 7.21.

Description

Returns an ActionID for the specified form field which can be used with the various action manipulation functions such as **GetActionType**, **GetActionDest**, **GetActionURL** and **SetActionURL**.

There are different trigger events and each one has it's own action.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldActionID(Index: Integer;
  TriggerEvent: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldActionID(
  Index As Long, TriggerEvent As String) As Long
```

DLL

```
int DPLGetFormFieldActionID(int InstanceID, int Index,
  wchar_t * TriggerEvent);
```

Parameters

Index	The index of the form field to work with. The first form field has an index of 1.
TriggerEvent	Retrieve the action for the specified trigger event: Empty string = simple activation action E = annotation enter action X = annotation exit action D = annotation button down action U = annotation button up action Fo = annotation input focus action Bl = annotation input blur (opposite of focus) action PO = annotation page open action PC = annotation page close action PV = annotation page visible action PI = annotation page invisible action K = form field change action F = form field format action V = form field validate action C = form field calculate action

Return values

0	The field index was invalid or the field does not have an action associated with the specified trigger event
Non-zero	The form field's ActionID for the specified trigger event

GetFormFieldAlignment

Description

Retrieves the text alignment of the specified form field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldAlignment(
  Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldAlignment(
  Index As Long) As Long
```

DLL

```
int DPLGetFormFieldAlignment(int InstanceID, int Index);
```

Parameters

Index	The index of the form field to work with. The first form field has an index of 1.

Return values

0	Left alignment (this value is also returned if the form field could not be found)
1	Centered
2	Right aligned

GetFormFieldAnnotFlags

Description

Get the "annotation" flags for the specified form field. This is for advanced use. See the PDF specification for full details.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldAnnotFlags(
  Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldAnnotFlags(
  Index As Long) As Long
```

DLL

```
int DPLGetFormFieldAnnotFlags(int InstanceID, int Index);
```

Parameters

Index	The index of the form field to check

GetFormFieldBackgroundColor

Description

Returns the background color of the specified field. The number of available values will depend on the color type specified in the form field. The number of components available can be retrieved using the**GetFormFieldBackgroundColorType** function.

0 = No color specified
1 = DeviceGray (1 component)
3 = DeviceRGB (3 components)
4 = CMYK (4 components

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldBackgroundColor(Index,
  ColorComponent: Integer): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldBackgroundColor(
  Index As Long, ColorComponent As Long) As Double
```

DLL

```
double DPLGetFormFieldBackgroundColor(int InstanceID, int Index,
  int ColorComponent);
```

Parameters

Index	The index of the form field to examine
ColorComponent	For DeviceGray (color type = 1) 1 = Gray level
	For DeviceRGB (color type = 3) 1 = Red 2 = Green 3 = Blue
	For DeviceCMYK (color type = 4) 1 = Cyan 2 = Magenta 3 = Yellow 4 = Black

GetFormFieldBackgroundColorType

Form fields, Color

Version history

This function was introduced in Quick PDF Library version 9.12.

Description

Returns the number of color components of the specified field's background.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldBackgroundColorType(
  Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldBackgroundColorType(
  Index As Long) As Long
```

DLL

```
int DPLGetFormFieldBackgroundColorType(int InstanceID, int Index);
```

Parameters

Index	The index of the field to examine

Return values

0	No color defined
1	Gray
3	RGB
4	CMYK

GetFormFieldBorderColor

Description

Returns the color of the specified field's border. The number of available values will depend on the color type specified in the form field. The number of components available can be retrieved using the**GetFormFieldBorderColorType** function.

0 = No color specified
1 = DeviceGray (1 component)
3 = DeviceRGB (3 components)
4 = CMYK (4 components

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldBorderColor(Index,
  ColorComponent: Integer): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldBorderColor(
  Index As Long, ColorComponent As Long) As Double
```

DLL

```
double DPLGetFormFieldBorderColor(int InstanceID, int Index,
  int ColorComponent);
```

Parameters

Index	The index of the form field to examine
ColorComponent	For DeviceGray (color type = 1) 1 = Gray level
	For DeviceRGB (color type = 3) 1 = Red 2 = Green 3 = Blue
	For DeviceCMYK (color type = 4) 1 = Cyan 2 = Magenta 3 = Yellow 4 = Black

GetFormFieldBorderColorType

Form fields, Color

Version history

This function was introduced in Quick PDF Library version 9.12.

Description

Returns the number of color components of the specified field's border.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldBorderColorType(
  Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldBorderColorType(
  Index As Long) As Long
```

DLL

```
int DPLGetFormFieldBorderColorType(int InstanceID, int Index);
```

Parameters

Index	The index of the form field to examine

Return values

0	No color defined
1	Gray
3	RGB
4	CMYK

GetFormFieldBorderProperty

Description

Returns various properties of the specified field's border.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldBorderProperty(Index,
  PropKey: Integer): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldBorderProperty(
  Index As Long, PropKey As Long) As Double
```

DLL

```
double DPLGetFormFieldBorderProperty(int InstanceID, int Index,
  int PropKey);
```

Parameters

Index	The index of the form field to examine
PropKey	1 = Border width 2 = Dash on 3 = Dash off

GetFormFieldBorderStyle

Form fields

Description

Returns the border style of the specified field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldBorderStyle(
  Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldBorderStyle(
  Index As Long) As Long
```

DLL

```
int DPLGetFormFieldBorderStyle(int InstanceID, int Index);
```

Parameters

Index	The index of the form field to examine

Return values

0	Solid
1	Dashed
2	Beveled
3	Inset
4	Underline

GetFormFieldBound

Description

Returns the bounding box of the specified form field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldBound(Index,
  Edge: Integer): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldBound(Index As Long,
  Edge As Long) As Double
```

DLL

```
double DPLGetFormFieldBound(int InstanceID, int Index, int Edge);
```

Parameters

Index	The index of the form field to measure. The first form field has an index of 1.
Edge	The required edge: 0 = Left 1 = Top 2 = Width 3 = Height

Return values

0	Could not find the specified form field
Non-zero	The requested measurement

GetFormFieldCaption

Description

Returns the caption of a form field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldCaption(
  Index: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldCaption(
  Index As Long) As String
```

DLL

```
wchar_t * DPLGetFormFieldCaption(int InstanceID, int Index);
```

Parameters

Index	The index of the form field

GetFormFieldCaptionEx

Form fields

Version history

This function was introduced in Quick PDF Library version 11.11.

Description

Returns the caption of a form field, based on the specified parameter.
The parameter specifies which string to extract. Otions are for CA, RC and AC strings, but the RC and AC strings are reserved only for pushbuttons. Trying to extract RC or AC string from radiobutton or checkbox will result in null string because these are not used in this types of buttons. More info can be found in PDF format reference manual.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldCaptionEx(Index,
  StringType: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldCaptionEx(
  Index As Long, StringType As Long) As String
```

DLL

```
wchar_t * DPLGetFormFieldCaptionEx(int InstanceID, int Index,
  int StringType);
```

Parameters

Index	The index of the form field
StringType	1 = CA String 2 = RC String 3 = AC String

GetFormFieldCheckStyle

Version history

This function was introduced in Quick PDF Library version 9.13.

Description

Returns the checkbox style of the specified form field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldCheckStyle(
  Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldCheckStyle(
  Index As Long) As Long
```

DLL

```
int DPLGetFormFieldCheckStyle(int InstanceID, int Index);
```

Parameters

Index	The index of the form field

Return values

0	Cross
1	Check (Tick)
2	Dot (Radio)
3	XP Check
4	XP Radio
5	Diamond
6	Square
7	Start

GetFormFieldChildTitle

Form fields

Description

Form fields can be arranged in a hierarchical structure, and the title of the form field will be the full path to the field, for example "names.first" or "address.zipcode". This function will return only the last part of the title, "first" or "zipcode" in this example.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldChildTitle(
  Index: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldChildTitle(
  Index As Long) As String
```

DLL

```
wchar_t * DPLGetFormFieldChildTitle(int InstanceID, int Index);
```

Parameters

Index	The index of the form field to retrieve the title of

GetFormFieldChoiceType

Description

Determines whether a choice form field is a combo box or list box field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldChoiceType(
  Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldChoiceType(
  Index As Long) As Long
```

DLL

```
int DPLGetFormFieldChoiceType(int InstanceID, int Index);
```

Parameters

Index	The index of the form field

Return values

0	The form field is not a choice form field
1	The form field is a scrollable list box
2	The form field is a drop-down combo box
3	The form field is a multiselect scrollable list box
4	The form field is a drop-down combo box with an edit box

GetFormFieldColor

Description

Retrieves the color of the text in the form field. This function must be called three times to retrieve all components of the color (red, green and blue).

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldColor(Index,
  ColorComponent: Integer): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldColor(Index As Long,
  ColorComponent As Long) As Double
```

DLL

```
double DPLGetFormFieldColor(int InstanceID, int Index, int ColorComponent);
```

Parameters

Index	The index of the form field to work with. The first form field has an index of 1.
ColorComponent	1 = Red 2 = Green 3 = Blue

GetFormFieldComb

Description

Returns 1 if the specified form field is marked as a comb field, where each character in the value occupies the same space in the field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldComb(Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldComb(
  Index As Long) As Long
```

DLL

```
int DPLGetFormFieldComb(int InstanceID, int Index);
```

Parameters

Index	The index of the form field

GetFormFieldDefaultValue

Description

Returns the default value of a form field. This is the value that the field will have when the form is reset.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldDefaultValue(
   Index: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldDefaultValue(
   Index As Long) As String
```

DLL

```
wchar_t * DPLGetFormFieldDefaultValue(int InstanceID, int Index);
```

Parameters

Index	The index of the form field

GetFormFieldDescription

Description

Retrieves the description of the specified form field if it has one.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldDescription(
  Index: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldDescription(
  Index As Long) As String
```

DLL

```
wchar_t * DPLGetFormFieldDescription(int InstanceID, int Index);
```

Parameters

Index	The index of the form field to work with

GetFormFieldFlags

Description

Retrieves a form field's flags. This setting is for advanced purposes and most users will not need to use it.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldFlags(Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldFlags(
  Index As Long) As Long
```

DLL

```
int DPLGetFormFieldFlags(int InstanceID, int Index);
```

Parameters

Index	The index of the form field

Return values

0	Cannot find the form field
Non-zero	The flags for the specified form field

GetFormFieldFontName

Description

Retrieves the name of the font that the specified form field is using.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldFontName(
  Index: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldFontName(
  Index As Long) As String
```

DLL

```
wchar_t * DPLGetFormFieldFontName(int InstanceID, int Index);
```

Parameters

Index   The index of the form field to work with. The first form field has an index of 1.

GetFormFieldJavaScript

Description

Retrieves the JavaScript associated with the specified action for the specified form field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldJavaScript(Index: Integer;
  ActionType: WideString): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldJavaScript(
  Index As Long, ActionType As String) As String
```

DLL

```
wchar_t * DPLGetFormFieldJavaScript(int InstanceID, int Index,
  wchar_t * ActionType);
```

Parameters

Index	The index of the form field
ActionType	The action type: E = An action to be performed when the cursor enters the annotation's active area X = An action to be performed when the cursor exits the annotation's active area D = An action to be performed when the mouse button is pressed inside the annotation's active area U = An action to be performed when the mouse button is released inside the annotation's active area Fo = An action to be performed when the annotation receives the input focus Bl = An action to be performed when the annotation loses the input focus (blurred) K = An action to be performed when the user types a keystroke into a text field or combo box or modifies the selection in a scrollable list box. This allows the keystroke to be checked for validity and rejected or modified. F = An action to be performed before the field is formatted to display its current value. This allows the field's value to be modified before formatting. V = An action to be performed when the field's value is changed. This allows the new value to be checked for validity. C = An action to be performed in order to recalculate the value of this field when that of another field changes

GetFormFieldKidCount

Form fields

Description

Returns the number of children fields that the specified field has.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldKidCount(
  Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldKidCount(
  Index As Long) As Long
```

DLL

```
int DPLGetFormFieldKidCount(int InstanceID, int Index);
```

Parameters

Index	The index of the form field. The first field has an index of 1.

GetFormFieldKidTempIndex

Version history

This function was renamed in Quick PDF Library version 10.11.
The function name in earlier versions was GetFormFieldSubTempIndex.

Description

Returns a temporary index for the item fields (kids) of a radio button or checkbox form field group. An index of 1 will select the first radio or checkbox in the group, 2 the second and so on. The number of kids can be determined by calling **GetFormFieldKidCount**. This temporary index can be used with the regular form field functions such as **GetFormFieldTabOrder** and **GetFormFieldValue**.
If you need to update the subname for a choice field then you should use **SetFormFieldSubChoice** instead.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldKidTempIndex(Index,
  SubIndex: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldKidTempIndex(
  Index As Long, SubIndex As Long) As Long
```

DLL

```
int DPLGetFormFieldKidTempIndex(int InstanceID, int Index, int SubIndex);
```

Parameters

Index	The index of the radio-button form field
SubIndex	The index of the sub-field. The first sub-field has an index of 1.

GetFormFieldMaxLen

Description

Retrieves the maximum allowed length for a text form field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldMaxLen(Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldMaxLen(
  Index As Long) As Long
```

DLL

```
int DPLGetFormFieldMaxLen(int InstanceID, int Index);
```

Parameters

Index	The index of the form field to work with. The first form field has an index of 1.

Return values

0	The form field does not have a maximum length specified
Non-zero	The maximum length of the form field

GetFormFieldNoExport

Version history

This function was introduced in Quick PDF Library version 7.24.

Description

Returns the state of a field's NoExport flag.

The field will not be exported by a submit-form action if the NoExport flag is set.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldNoExport(
  Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldNoExport(
  Index As Long) As Long
```

DLL

```
int DPLGetFormFieldNoExport(int InstanceID, int Index);
```

Parameters

Index	The index of the form field

Return values

0	The field's NoExport flag is not set
1	The field's NoExport flag is set

GetFormFieldPage

Description

Returns the page number that the specified form field is on.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldPage(Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldPage(
   Index As Long) As Long
```

DLL

```
int DPLGetFormFieldPage(int InstanceID, int Index);
```

Parameters

Index	The index of the form field to locate

Return values

0	The form field could not be found, or the form field does not have valid page information
Non-zero	The page number of the page that the form field is displayed on

GetFormFieldPrintable

Description

Returns 1 if the specified field will be printed.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldPrintable(
  Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldPrintable(
  Index As Long) As Long
```

DLL

```
int DPLGetFormFieldPrintable(int InstanceID, int Index);
```

Parameters

Index	The index of the form field to check

GetFormFieldReadOnly

Form fields

Description

Returns the state of a field's ReadOnly flag.

The user cannot change the value of a form field if the ReadOnly flag is set.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldReadOnly(
  Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldReadOnly(
  Index As Long) As Long
```

DLL

```
int DPLGetFormFieldReadOnly(int InstanceID, int Index);
```

Parameters

Index	The index of the form field

Return values

0	The field's ReadOnly flag is not set
1	The field's ReadOnly flag is set

GetFormFieldRequired

Version history

This function was introduced in Quick PDF Library version 7.24.

Description

Returns the state of a field's is Required flag.

If this flag is set the field must have a value when the form is exported by a submit-form action.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldRequired(
  Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldRequired(
  Index As Long) As Long
```

DLL

```
int DPLGetFormFieldRequired(int InstanceID, int Index);
```

Parameters

Index	The index of the form field

Return values

0	The field's Required flag is not set
1	The field's Required flag is set

GetFormFieldRichTextString

Version history

This function was introduced in Quick PDF Library version 9.15.

Description

Retrieves the rich text (RV) or default style (DS) string of the specified form field using the given key. The format of the return value is defined in the PDF Specification under the section titled "Field Dictionaries".

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldRichTextString(Index: Integer;
  Key: WideString): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldRichTextString(
  Index As Long, Key As String) As String
```

DLL

```
wchar_t * DPLGetFormFieldRichTextString(int InstanceID, int Index,
  wchar_t * Key);
```

Parameters

Index	The index of the required form field. The first form field has an index of 1.
Key	The Key value to return.
	"RV" = returns the rich text string
	"DS" = returns the default style string

GetFormFieldRotation

Description

Returns the angle in degrees that the form field is rotated by. This is always a multiple of 90 degrees.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldRotation(
  Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldRotation(
  Index As Long) As Long
```

DLL

```
int DPLGetFormFieldRotation(int InstanceID, int Index);
```

Parameters

Index	The index of the form field to query

GetFormFieldSubCount

Form fields

Description

For radio button, checkbox items and choice fields (scrollable list box or combo box drop-down list), this function returns the number of possible values the form field can be set to.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldSubCount(
  Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldSubCount(
  Index As Long) As Long
```

DLL

```
int DPLGetFormFieldSubCount(int InstanceID, int Index);
```

Parameters

Index	The index of the form field to examine

Return values

0	The form field could not be found or it does not have sub-values
Non-zero	The number of possible values the form field can be set to

GetFormFieldSubDisplayName

Version history

This function was introduced in Quick PDF Library version 9.11.

Description

Similar to **GetformFieldSubName** but returns the display name of the specified choice field item.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldSubDisplayName(Index,
  SubIndex: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldSubDisplayName(
  Index As Long, SubIndex As Long) As String
```

DLL

```
wchar_t * DPLGetFormFieldSubDisplayName(int InstanceID, int Index,
  int SubIndex);
```

Parameters

Index	The index of the form field to examine
SubIndex	The index of the sub-value to retrieve

GetFormFieldSubName

Description

For radio button, checkbox and choice (scrollable list box or combo box drop-down list) form fields, this function returns the specified possible value.

For choice fields the **GetformFieldSubDisplayName** function can be used to retrieve the display name of the choice item.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldSubName(Index,
   SubIndex: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldSubName(
   Index As Long, SubIndex As Long) As String
```

DLL

```
wchar_t * DPLGetFormFieldSubName(int InstanceID, int Index, int SubIndex);
```

Parameters

Index	The index of the form field to examine
SubIndex	The index of the sub-value to retrieve

GetFormFieldSubmitActionString

Form fields

Version history

This function was introduced in Quick PDF Library version 10.14.

Description

Returns the string assocaiated with a FormField submit action action and its specified ActionType
Support ActionTypes
'U' : Returns the URL link string

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldSubmitActionString(
  Index: Integer; ActionType: WideString): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldSubmitActionString(
  Index As Long, ActionType As String) As String
```

DLL

```
wchar_t * DPLGetFormFieldSubmitActionString(int InstanceID, int Index,
  wchar_t * ActionType);
```

Parameters

Index	The index of the form field to examine
ActionType	The action type: U = An action to be performed when the mouse button is released inside the annotation's active area

GetFormFieldTabOrder

Description

Returns the tab order of the specified form field. The first form field on the page has a tab order of 1.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldTabOrder(
  Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldTabOrder(
  Index As Long) As Long
```

DLL

```
int DPLGetFormFieldTabOrder(int InstanceID, int Index);
```

Parameters

Index	The index of the form field

GetFormFieldTabOrderEx

Version history

> This function was introduced in Quick PDF Library version 9.13.

Description

> Returns the tab order of the specified form field. Similar to the **GetFormFieldTabOrder** function but the order is adjusted to match certain popular PDF viewers.
>
> The first form field on the page has a tab order of 1.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldTabOrderEx(Index,
  Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldTabOrderEx(
  Index As Long, Options As Long) As Long
```

DLL

```
int DPLGetFormFieldTabOrderEx(int InstanceID, int Index, int Options);
```

Parameters

Index	The index of the form field
Options	0 = Acrobat style 1 = Nuance style

Return values

0	The Index parameters was invalid or the Options parameter was out of range
1	Success

GetFormFieldTextFlags

Description

Returns certain properties of a text field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldTextFlags(Index,
  ValueKey: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldTextFlags(
  Index As Long, ValueKey As Long) As Long
```

DLL

```
int DPLGetFormFieldTextFlags(int InstanceID, int Index, int ValueKey);
```

Parameters

Index	The index of the form field
ValueKey	Indicates which property to analyse: 1 = Multiline 2 = Password 3 = FileSelect 4 = DoNotSpellCheck 5 = DoNotScroll

Return values

0	The flag for the specific property is not turned on. For example, if ValueKey is 5 and the function returns 0 this indicates that the form field is allowed to scroll.
1	The flag is turned on. For example, if ValueKey is 2 and the function returns 1 this indicates that the form field is a password field.

GetFormFieldTextSize

Description

Retrieves the size of the text in the specified form field. A value of 0 indicates that the form field autosizes the text to fit into the available space.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldTextSize(
  Index: Integer): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldTextSize(
  Index As Long) As Double
```

DLL

```
double DPLGetFormFieldTextSize(int InstanceID, int Index);
```

Parameters

Index	The index of the form field to work with. The first form field has an index of 1.

GetFormFieldTitle

Description

Returns the title of the specified form field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldTitle(
  Index: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldTitle(
  Index As Long) As String
```

DLL

```
wchar_t * DPLGetFormFieldTitle(int InstanceID, int Index);
```

Parameters

| Index | The index of the required form field. The first form field has an index of 1. |

GetFormFieldType

Description

Returns the type of the specified form field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldType(Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldType(
  Index As Long) As Long
```

DLL

```
int DPLGetFormFieldType(int InstanceID, int Index);
```

Parameters

Index	The index of the form field

Return values

0	Unknown
1	Text
2	Pushbutton
3	Checkbox
4	Radiobutton
5	Choice
6	Signature
7	Parent

GetFormFieldValue

Description

Retrieves the value of the specified form field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldValue(
   Index: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldValue(
   Index As Long) As String
```

DLL

```
wchar_t * DPLGetFormFieldValue(int InstanceID, int Index);
```

Parameters

Index	The index of the form field to retrieve the value of

GetFormFieldValueByTitle

Description

Returns the value of the form field with the specified title.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldValueByTitle(
  Title: WideString): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldValueByTitle(
  Title As String) As String
```

DLL

```
wchar_t * DPLGetFormFieldValueByTitle(int InstanceID, wchar_t * Title);
```

Parameters

Title	The title of the field.

GetFormFieldVisible

Description

Returns 1 if the specified field will be visible when the document is viewed.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldVisible(
  Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldVisible(
  Index As Long) As Long
```

DLL

```
int DPLGetFormFieldVisible(int InstanceID, int Index);
```

Parameters

Index	The index of the form field to check

GetFormFieldWebLink

Description

Returns the internet address that the specified form field's action points to, if any.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFieldWebLink(Index: Integer;
  ActionType: WideString): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFieldWebLink(
  Index As Long, ActionType As String) As String
```

DLL

```
wchar_t * DPLGetFormFieldWebLink(int InstanceID, int Index,
  wchar_t * ActionType);
```

Parameters

Index	The index of the form field to change
ActionType	The action type: E = An action to be performed when the cursor enters the annotation's active area X = An action to be performed when the cursor exits the annotation's active area D = An action to be performed when the mouse button is pressed inside the annotation's active area U = An action to be performed when the mouse button is released inside the annotation's active area Fo = An action to be performed when the annotation receives the input focus Bl = An action to be performed when the annotation loses the input focus (blurred) K = An action to be performed when the user types a keystroke into a text field or combo box or modifies the selection in a scrollable list box. This allows the keystroke to be checked for validity and rejected or modified. F = An action to be performed before the field is formatted to display its current value. This allows the field's value to be modified before formatting. V = An action to be performed when the field's value is changed. This allows the new value to be checked for validity. C = An action to be performed in order to recalculate the value of this field when that of another field changes

GetFormFontCount

Description

Returns the number of fonts available to fields in the form.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFontCount: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFontCount As Long
```

DLL

```
int DPLGetFormFontCount(int InstanceID);
```

GetFormFontName

Description

Returns the name of the font with the specified index.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetFormFontName(
  FontIndex: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetFormFontName(
  FontIndex As Long) As String
```

DLL

```
wchar_t * DPLGetFormFontName(int InstanceID, int FontIndex);
```

Parameters

FontIndex   The index of the font to work with. The first font in the form has an index of 1.
Use **GetFormFontCount** to determine the number of fonts available in the form.

GetGlobalJavaScript

Description

Retrieves the global JavaScript for the specified package.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetGlobalJavaScript(
  PackageName: WideString): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetGlobalJavaScript(
  PackageName As String) As String
```

DLL

```
wchar_t * DPLGetGlobalJavaScript(int InstanceID, wchar_t * PackageName);
```

Parameters

PackageName	The JavaScript stored under this package name will be retrieved.

GetHTMLTextHeight

Description

Returns the height that a certain block of HTML text will occupy if drawn onto the page. See **Appendix A** for details of the supported HTML tags.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetHTMLTextHeight(Width: Double;
  HTMLText: WideString): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetHTMLTextHeight(
  Width As Double, HTMLText As String) As Double
```

DLL

```
double DPLGetHTMLTextHeight(int InstanceID, double Width,
  wchar_t * HTMLText);
```

Parameters

Width	The width of the area the text would be drawn into
HTMLText	The HTML to determine the height of. See **Appendix A** for details of the supported HTML tags.

GetHTMLTextLineCount

Version history

This function was introduced in Quick PDF Library version 7.11.

Description

Returns the number of lines a block of HTML text will take up if it is drawn using the **DrawHTMLText** function. See **Appendix A** for details of the supported HTML tags.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetHTMLTextLineCount(Width: Double;
  HTMLText: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetHTMLTextLineCount(
  Width As Double, HTMLText As String) As Long
```

DLL

```
int DPLGetHTMLTextLineCount(int InstanceID, double Width,
  wchar_t * HTMLText);
```

Parameters

Width	The width of the area the text would be drawn into
HTMLText	The HTML text to determine the number of lines of

GetHTMLTextWidth

Version history

This function was introduced in Quick PDF Library version 7.11.

Description

Returns the actual horizontal size of a block of HTML text when wrapped to a maximum width by the **DrawHTMLText** function. See **Appendix A** for details of the supported HTML tags.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetHTMLTextWidth(MaxWidth: Double;
  HTMLText: WideString): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetHTMLTextWidth(
  MaxWidth As Double, HTMLText As String) As Double
```

DLL

```
double DPLGetHTMLTextWidth(int InstanceID, double MaxWidth,
  wchar_t * HTMLText);
```

Parameters

MaxWidth	The width of the area the text would be drawn into
HTMLText	The HTML text to determine the width of

GetImageID

Version history

This function was renamed in Quick PDF Library version 7.11.
The function name in earlier versions was ImageID.

Description

Returns the ID of the specified image.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetImageID(Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetImageID(Index As Long) As Long
```

DLL

```
int DPLGetImageID(int InstanceID, int Index);
```

Parameters

Index	The index of the image. The first image has an index of 1.

Return values

0	The index is out of bounds
Non-zero	The ID of the specified image

GetImageListCount

Image handling

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Version history

This function was introduced in Quick PDF Library version 8.13.

Description

Returns the number of images in an image list.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetImageListCount(
  ImageListID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetImageListCount(
  ImageListID As Long) As Long
```

DLL

```
int DPLGetImageListCount(int InstanceID, int ImageListID);
```

Parameters

ImageListID   A value returned by the **GetPageImageList** function

GetImageListItemDataToString

Version history

This function was introduced in Quick PDF Library version 8.13.

Description

Returns the image data of the specified image list item as a string of 8-bit bytes.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetImageListItemDataToString(ImageListID,
  ImageIndex, Options: Integer): AnsiString;
```

DLL

```
char * DPLGetImageListItemDataToString(int InstanceID, int ImageListID,
  int ImageIndex, int Options);
```

Parameters

ImageListID	A value returned by the **GetPageImageList** function
ImageIndex	The index of the image in the list. The first image has an index of 1.
Options	Reserved for future use. Should be set to 0.

GetImageListItemDataToVariant

Image handling

Version history

This function was introduced in Quick PDF Library version 8.13.

Description

Returns the image data of the specified image list item as a variant byte array.

Syntax

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetImageListItemDataToVariant(
  ImageListID As Long, ImageIndex As Long,
  Options As Long) As Variant
```

Parameters

ImageListID	A value returned by the **GetPageImageList** function
ImageIndex	The index of the image in the list. The first image has an index of 1.
Options	Reserved for future use. Should be set to 0.

GetImageListItemDblProperty

Version history

> This function was introduced in Quick PDF Library version 8.13.

Description

> Returns a Double type property of the specified image list item.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetImageListItemDblProperty(ImageListID,
  ImageIndex, PropertyID: Integer): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetImageListItemDblProperty(
  ImageListID As Long, ImageIndex As Long,
  PropertyID As Long) As Double
```

DLL

```
double DPLGetImageListItemDblProperty(int InstanceID, int ImageListID,
  int ImageIndex, int PropertyID);
```

Parameters

ImageListID	A value returned by the **GetPageImageList** function
ImageIndex	The index of the image in the list. The first image has an index of 1.
PropertyID	501 = Horizontal co-ordinate of top-left corner 502 = Vertical co-ordinate of top-left corner 503 = Horizontal co-ordinate of top-right corner 504 = Vertical co-ordinate of top-right corner 505 = Horizontal co-ordinate of bottom-right corner 506 = Vertical co-ordinate of bottom-right corner 507 = Horizontal co-ordinate of bottom-left corner 508 = Vertical co-ordinate of bottom-left corner

GetImageListItemFormatDesc

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Version history

This function was introduced in Quick PDF Library version 11.14.

Description

Returns a string containing the format details of the specified image in the image list.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetImageListItemFormatDesc(ImageListID,
  ImageIndex, Options: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetImageListItemFormatDesc(
  ImageListID As Long, ImageIndex As Long,
  Options As Long) As String
```

DLL

```
wchar_t * DPLGetImageListItemFormatDesc(int InstanceID, int ImageListID,
  int ImageIndex, int Options);
```

Parameters

ImageListID	A value returned by the **GetPageImageList** function
ImageIndex	The index of the image in the list. The first image has an index of 1.
Options	0 = Key/value pair 1 = Textual description

GetImageListItemIntProperty

Image handling

Version history

This function was introduced in Quick PDF Library version 8.13.

Description

Returns an Integer type property of the specified image list item.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetImageListItemIntProperty(ImageListID,
  ImageIndex, PropertyID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetImageListItemIntProperty(
  ImageListID As Long, ImageIndex As Long,
  PropertyID As Long) As Long
```

DLL

```
int DPLGetImageListItemIntProperty(int InstanceID, int ImageListID,
  int ImageIndex, int PropertyID);
```

Parameters

ImageListID	A value returned by the **GetPageImageList** function
ImageIndex	The index of the image in the list. The first image has an index of 1.
PropertyID	400 = Image type (see **ImageType**) for values 401 = Width in pixels 402 = Height in pixels 403 = Bits per pixel 404 = Color space type 405 = Image ID (will be 0 if it is an Inline image) 406 = Constant Image ID 407 = Image object number

Return values

1	JPEG (for image type) DeviceGray (for color space type)
2	BMP (for image type) DeviceRGB (for color space type)
3	TIFF (for image type) DeviceCMYK (for color space type)
4	PNG (for image type) The selected image is a PNG image. This is only possible when using the **GetPageImageList** function where an image has a mask. The library will create an Transparent PNG file if a mask is found.
-1	Unknown (for color space type)

GetImageMeasureDict

Measurement and coordinate units

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Returns the measurement dictionary for the selected image as a MeasureDictID value.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetImageMeasureDict: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetImageMeasureDict As Long
```

DLL

```
int DPLGetImageMeasureDict(int InstanceID);
```

Return values

0	The measure dictionary of the selected image could not be found
Non-zero	A MeasureDictID value

GetImagePageCount

Description

Returns the number of pages in the specified image file. Most images consist of 1 page, but TIFF images may contain multiple pages.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetImagePageCount(
  FileName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetImagePageCount(
  FileName As String) As Long
```

DLL

```
int DPLGetImagePageCount(int InstanceID, wchar_t * FileName);
```

Parameters

FileName	The name of the image file to analyse.

Return values

0	The image file is invalid or does not exist
Non-zero	The number of pages in the specified image

GetImagePageCountFromString

Version history

This function was introduced in Quick PDF Library version 7.19.

Description

Returns the number of pages in the provided image data. Most images consist of 1 page, but TIFF images may contain multiple pages.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetImagePageCountFromString(
  const Source: AnsiString): Integer;
```

DLL

```
int DPLGetImagePageCountFromString(int InstanceID, char * Source);
```

Parameters

Source	A string containing the image data. In the ActiveX version of the library this string must contain 16-bit characters, only the lower 8-bits of each character will be used.

Return values

0	The image data is invalid
Non-zero	The number of pages in the image

GetImagePtDataDict

Measurement and coordinate units

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Returns the PtData dictionary for the selected image as a PtDataDictID value.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetImagePtDataDict: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetImagePtDataDict As Long
```

DLL

```
int DPLGetImagePtDataDict(int InstanceID);
```

Return values

0	The PtData dictionary for the selected image could not be found
Non-zero	A PtDataDictID value

GetInformation

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Get the properties of the selected document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetInformation(Key: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetInformation(
  Key As Long) As String
```

DLL

```
wchar_t * DPLGetInformation(int InstanceID, int Key);
```

Parameters

Key   The property to get:
0 = PDF Version
1 = Author
2 = Title
3 = Subject
4 = Keywords
5 = Creator
6 = Producer
7 = Creation date
8 = Modification date

GetInstalledFontsByCharset

Version history

This function was introduced in Quick PDF Library version 7.21.

Description

Returns a list of the names of fonts that are installed. These font names can be used with the **AddTrueTypeFont** and **AddSubsettedFont** functions.
The list is filtered by the specified character set. To show all fonts, set CharsetIndex to 2 (corresponding to DEFAULT_CHARSET).

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetInstalledFontsByCharset(CharsetIndex,
  Options: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetInstalledFontsByCharset(
  CharsetIndex As Long, Options As Long) As String
```

DLL

```
wchar_t * DPLGetInstalledFontsByCharset(int InstanceID, int CharsetIndex,
  int Options);
```

Parameters

CharsetIndex	1 = ANSI 2 = Default 3 = Symbol 4 = Shift JIS 5 = Hangeul 6 = GB2312 7 = Chinese Big 5 8 = OEM 9 = Johab 10 = Hebrew 11 = Arabic 12 = Greek 13 = Turkish 14 = Vietnamese 15 = Thai 16 = East Europe 17 = Russian 18 = Mac 19 = Baltic
Options	0 = Font names enclosed in double quotes with comma delimiters 1 = Font names in plain text with CRLF delimiters

GetInstalledFontsByCodePage

Version history

This function was introduced in Quick PDF Library version 7.21.

Description

Returns a list of the names of fonts that are installed. These font names can be used with the **AddTrueTypeFont** and **AddSubsettedFont** functions.

The list is filtered by the specified code page. To show all fonts, set CodePage to 0 (corresponding to DEFAULT_CHARSET).

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetInstalledFontsByCodePage(CodePage,
  Options: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetInstalledFontsByCodePage(
  CodePage As Long, Options As Long) As String
```

DLL

```
wchar_t * DPLGetInstalledFontsByCodePage(int InstanceID, int CodePage,
  int Options);
```

Parameters

CodePage	0 = DEFAULT_CHARSET
	437 = OEM_CHARSET
	850 = OEM_CHARSET
	852 = OEM_CHARSET
	874 = THAI_CHARSET
	932 = SHIFTJIS_CHARSET
	936 = GB2312_CHARSET
	949 = HANGEUL_CHARSET
	950 = CHINESEBIG5_CHARSET
	1250 = EASTEUROPE_CHARSET
	1251 = RUSSIAN_CHARSET
	1252 = ANSI_CHARSET
	1253 = GREEK_CHARSET
	1254 = TURKISH_CHARSET
	1255 = HEBREW_CHARSET
	1256 = ARABIC_CHARSET
	1257 = BALTIC_CHARSET
	1258 = VIETNAMESE_CHARSET
	1361 = JOHAB_CHARSET
Options	0 = Font names enclosed in double quotes with comma delimiters
	1 = Font names in plain text with CRLF delimiters

GetKerning

Description

Returns the amount of kerning for the specified character pair.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetKerning(CharPair: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetKerning(
  CharPair As String) As Long
```

DLL

```
int DPLGetKerning(int InstanceID, wchar_t * CharPair);
```

Parameters

CharPair	A two-character string containing the characters making the kerning pair, for example "AW"

Return values

The amount the space between the kerning pair will be reduced by. This is the same value as shown in graphics programs such as Adobe Illustrator. A value of 1000 is the same as the height of the text.

GetLatestPrinterNames

Version history

This function was introduced in Quick PDF Library version 8.13.

Description

Similar to the **GetPrinterNames** function but returns the latest list of printers rather than the cached list that was enumerated when the app started. This function may take some time to execute depending on the number of network printers installed.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetLatestPrinterNames: WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetLatestPrinterNames As String
```

DLL

```
wchar_t * DPLGetLatestPrinterNames(int InstanceID);
```

GetMaxObjectNumber

Document properties, Miscellaneous functions

Version history

This function was introduced in Quick PDF Library version 7.21.

Description

Returns the highest object number in the selected document. This is for advanced use.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetMaxObjectNumber: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetMaxObjectNumber As Long
```

DLL

```
int DPLGetMaxObjectNumber(int InstanceID);
```

GetMeasureDictBoundsCount

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Returns the number of items in a measurement dictionary Bounds array.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetMeasureDictBoundsCount(
  MeasureDictID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetMeasureDictBoundsCount(
  MeasureDictID As Long) As Long
```

DLL

```
int DPLGetMeasureDictBoundsCount(int InstanceID, int MeasureDictID);
```

Parameters

MeasureDictID   A value returned from the **GetImageMeasureDict** function

GetMeasureDictBoundsItem

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Returns an item from a measurement dictionary Bounds array.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetMeasureDictBoundsItem(MeasureDictID,
  ItemIndex: Integer): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetMeasureDictBoundsItem(
  MeasureDictID As Long, ItemIndex As Long) As Double
```

DLL

```
double DPLGetMeasureDictBoundsItem(int InstanceID, int MeasureDictID,
  int ItemIndex);
```

Parameters

MeasureDictID	A value returned from the **GetImageMeasureDict** function
ItemIndex	The index of the item to return. The first item has an index of 1.

GetMeasureDictCoordinateSystem

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Returns the coordinate system type of a measurement dictionary.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetMeasureDictCoordinateSystem(
  MeasureDictID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetMeasureDictCoordinateSystem(
  MeasureDictID As Long) As Long
```

DLL

```
int DPLGetMeasureDictCoordinateSystem(int InstanceID, int MeasureDictID);
```

Parameters

MeasureDictID	A value returned from the **GetImageMeasureDict** function

Return values

0	The MeasureDictID parameter was incorrect
1	The measurement dictionary is a rectiliniar coordinate system (RL)
2	The measurement dictionary is a geospatial coordinate system (GEO)

GetMeasureDictDCSDict

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Returns the DCS coordinate system dictionary of a measurement dictionary (used for display purposes) as a CSDictID value.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetMeasureDictDCSDict(
  MeasureDictID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetMeasureDictDCSDict(
  MeasureDictID As Long) As Long
```

DLL

```
int DPLGetMeasureDictDCSDict(int InstanceID, int MeasureDictID);
```

Parameters

MeasureDictID	A value returned from the **GetImageMeasureDict** function

Return values

0	The MeasureDictID parameter was incorrect
Non-zero	A CSDictID value

GetMeasureDictGCSDict

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Returns the GCS coordinate system dictionary of a measurement dictionary as a CSDictID value.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetMeasureDictGCSDict(
  MeasureDictID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetMeasureDictGCSDict(
  MeasureDictID As Long) As Long
```

DLL

```
int DPLGetMeasureDictGCSDict(int InstanceID, int MeasureDictID);
```

Parameters

MeasureDictID	A value returned from the **GetImageMeasureDict** function

Return values

0	The MeasureDictID parameter was incorrect
Non-zero	A CSDict value

GetMeasureDictGPTSCount

Measurement and coordinate units

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Returns the number of items in the GPTS array of a measurement dictionary.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetMeasureDictGPTSCount(
  MeasureDictID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetMeasureDictGPTSCount(
  MeasureDictID As Long) As Long
```

DLL

```
int DPLGetMeasureDictGPTSCount(int InstanceID, int MeasureDictID);
```

Parameters

MeasureDictID	A value returned from the **GetImageMeasureDict** function

GetMeasureDictGPTSItem

Measurement and coordinate units

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Returns a value from the GPTS array of a measurement dictionary.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetMeasureDictGPTSItem(MeasureDictID,
  ItemIndex: Integer): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetMeasureDictGPTSItem(
  MeasureDictID As Long, ItemIndex As Long) As Double
```

DLL

```
double DPLGetMeasureDictGPTSItem(int InstanceID, int MeasureDictID,
  int ItemIndex);
```

Parameters

MeasureDictID	A value returned from the **GetImageMeasureDict** function
ItemIndex	The index of the item. The first item has an index of 1.

GetMeasureDictLPTSCount

Measurement and coordinate units

Version history

    This function was introduced in Quick PDF Library version 8.11.

Description

    Returns the number of items in the LPTS array of a measurement dictionary.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetMeasureDictLPTSCount(
  MeasureDictID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetMeasureDictLPTSCount(
  MeasureDictID As Long) As Long
```

DLL

```
int DPLGetMeasureDictLPTSCount(int InstanceID, int MeasureDictID);
```

Parameters

MeasureDictID	A value returned from the **GetImageMeasureDict** function

GetMeasureDictLPTSItem

Measurement and coordinate units

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Returns a value from the CPTS array of a measurement dictionary.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetMeasureDictLPTSItem(MeasureDictID,
  ItemIndex: Integer): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetMeasureDictLPTSItem(
  MeasureDictID As Long, ItemIndex As Long) As Double
```

DLL

```
double DPLGetMeasureDictLPTSItem(int InstanceID, int MeasureDictID,
  int ItemIndex);
```

Parameters

MeasureDictID	A value returned from the **GetImageMeasureDict** function
ItemIndex	The index of the item. The first item has an index of 1.

GetMeasureDictPDU

Measurement and coordinate units

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetMeasureDictPDU(MeasureDictID,
  UnitIndex: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetMeasureDictPDU(
  MeasureDictID As Long, UnitIndex As Long) As Long
```

DLL

```
int DPLGetMeasureDictPDU(int InstanceID, int MeasureDictID, int UnitIndex);
```

Parameters

MeasureDictID	A value returned from the **GetImageMeasureDict** function
UnitIndex	1 = Linear display units 2 = Area display units 3 = Angular display units

Return values

0	The MeasureDictID parameter was incorrect.
1	Linear units: M (a meter) Area units: SQM (a square meter) Angular units: DEG (a degree)
2	Linear units: KM (a kilometer) Area units: HA (a hectare) Angular units: GRD (a grad = 0.9 degrees)
3	Linear units: FT (an international foot) Area units: SQKM (a square kilometer)
4	Linear units: USFT (a U.S. Survey foot) Area units: SQFT (a square foot)
5	Linear units: MI (an international mile) Area units: A (a acre)
6	Linear units: MI (an international nautical mile) Area units: SQMI (a square mile)

GetNamedDestination

Version history

This function was introduced in Quick PDF Library version 7.13.

Description

Locates the named destination with the specified name and returns a DestID that can be used with the **GetDestPage**, **GetDestType** and **GetDestValue** functions.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetNamedDestination(
  DestName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetNamedDestination(
  DestName As String) As Long
```

DLL

```
int DPLGetNamedDestination(int InstanceID, wchar_t * DestName);
```

Parameters

DestName	The name of the named destination to search for

Return values

0	The specified named destination could not be found
Non-zero	A DestID that can be used with the destination functions

GetNextOutline

Description

Returns the ID of the outline that is below the specified outline at the same level.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetNextOutline(OutlineID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetNextOutline(
  OutlineID As Long) As Long
```

DLL

```
int DPLGetNextOutline(int InstanceID, int OutlineID);
```

Parameters

OutlineID	The ID of the outline item to work with. This ID is returned by the **NewOutline** or **NewStaticOutline** functions, or retrieved with the **GetOutlineID** function or Get*Outline functions.

GetObjectCount

Version history

This function was introduced in Quick PDF Library version 7.16.

Description

Returns the number of raw PDF objects in the document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetObjectCount: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetObjectCount As Long
```

DLL

```
int DPLGetObjectCount(int InstanceID);
```

GetObjectDecodeError

Version history

This function was introduced in Quick PDF Library version 9.15.

Description

This function can be used to determine if an error was encountered during decoding of the raw PDF object from the file.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetObjectDecodeError(
  ObjectNumber: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetObjectDecodeError(
  ObjectNumber As Long) As Long
```

DLL

```
int DPLGetObjectDecodeError(int InstanceID, int ObjectNumber);
```

Parameters

ObjectNumber	The number of the object to retrieve. The first object is numbered 1 and the last object has an object number equal to the result of the **GetObjectCount** function.

Return values

0	The object was decoded successfully
1	The object could not be decoded

GetObjectToString

Version history

This function was renamed in Quick PDF Library version 8.11.
The function name in earlier versions was GetObjectSource.

Description

Returns the raw PDF object data for the specified object number. This is for advanced use only.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetObjectToString(
  ObjectNumber: Integer): AnsiString;
```

DLL

```
char * DPLGetObjectToString(int InstanceID, int ObjectNumber);
```

Parameters

ObjectNumber	The number of the object to retrieve. The first object is numbered 1 and the last object has an object number equal to the result of the **GetObjectCount** function.

GetObjectToVariant

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Returns the raw PDF object data for the specified object number as a variant byte array. This is for advanced use only.

Syntax

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetObjectToVariant(
  ObjectNumber As Long) As Variant
```

Parameters

ObjectNumber The number of the object to retrieve. The first object is numbered 1 and the last object has an object number equal to the result of the **GetObjectCount** function.

GetOpenActionDestination

Description

Retrieves the ID of the open action destination, if any. This ID can be used with the **GetDestPage**, **GetDestType** and **GetDestValue** functions to obtain information about the open action destination.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetOpenActionDestination: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetOpenActionDestination As Long
```

DLL

```
int DPLGetOpenActionDestination(int InstanceID);
```

Return values

0	The document does not have an open action destination
Non-zero	A DestID that can be used with the **GetDestPage**, **GetDestType** and **GetDestValue** functions

GetOpenActionJavaScript

Description

Retrieves the JavaScript linked to the document's open action.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetOpenActionJavaScript: WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetOpenActionJavaScript As String
```

DLL

```
wchar_t * DPLGetOpenActionJavaScript(int InstanceID);
```

GetOptionalContentConfigCount

Version history

This function was introduced in Quick PDF Library version 7.21.

Description

Returns the number of optional content configuration dictionaries in the selected document.

The first optional content configuration dictionary is used to specify the initial state of the optional content groups when the document is first opened by a PDF viewer. Other configuration dictionaries are used in other circumstances.

The **GetOptionalContentConfigState** function can be used to determine the state of the optional content groups as defined by a particular optional content configuration dictionary.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetOptionalContentConfigCount: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetOptionalContentConfigCount As Long
```

DLL

```
int DPLGetOptionalContentConfigCount(int InstanceID);
```

Return values

0	The document does not have any optional content configuration dictionaries.
Non-zero	The number of optional content configuration dictionaries in the document.

GetOptionalContentConfigLocked

Version history

This function was introduced in Quick PDF Library version 8.15.

Description

This function is used to determine if an optional content group is locked as defined by the specified optional content configuration dictionary.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetOptionalContentConfigLocked(
  OptionalContentConfigID, OptionalContentGroupID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetOptionalContentConfigLocked(
  OptionalContentConfigID As Long,
  OptionalContentGroupID As Long) As Long
```

DLL

```
int DPLGetOptionalContentConfigLocked(int InstanceID,
  int OptionalContentConfigID, int OptionalContentGroupID);
```

Parameters

OptionalContentConfigID	The first default optional content configuration dictionary has an ID of 1. Higher numbers are used for other optional content configuration dictionaries.
OptionalContentGroupID	An ID returned by the **NewOptionalContentGroup**, **GetOptionalContentGroupID** or **GetOptionalContentConfigOrderItemID** functions

Return values

0	The optional content group is unlocked
1	The optional content group is locked

GetOptionalContentConfigOrderCount

Version history

This function was introduced in Quick PDF Library version 7.21.

Description

Returns the number of items in the order array of the specified optional content configuration dictionary.

The order array defines a tree structure with labels and optional content group items that can be used in the user interface of the PDF viewer application.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetOptionalContentConfigOrderCount(
  OptionalContentConfigID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetOptionalContentConfigOrderCount(
  OptionalContentConfigID As Long) As Long
```

DLL

```
int DPLGetOptionalContentConfigOrderCount(int InstanceID,
  int OptionalContentConfigID);
```

Parameters

OptionalContentConfigID	The first default optional content configuration dictionary has an ID of 1. Higher numbers are used for other optional content configuration dictionaries.

GetOptionalContentConfigOrderItemID

Version history

This function was introduced in Quick PDF Library version 7.21.

Description

Returns the OptionalContentGroupID for an item in the order array of the specified optional content configuration dictionary.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetOptionalContentConfigOrderItemID(
  OptionalContentConfigID, ItemIndex: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetOptionalContentConfigOrderItemID(
  OptionalContentConfigID As Long, ItemIndex As Long) As Long
```

DLL

```
int DPLGetOptionalContentConfigOrderItemID(int InstanceID,
  int OptionalContentConfigID, int ItemIndex);
```

Parameters

OptionalContentConfigID	The first default optional content configuration dictionary has an ID of 1. Higher numbers are used for other optional content configuration dictionaries.
ItemIndex	The index number of the item in the order array. The first item has an index number of 1 and the last item has an index equal to the value returned by the **GetOptionalContentConfigOrderCount** function.

Return values

0	The specified item could not be found or it is a label item and does not have an associated optional content group.
Non-zero	The OptionalContentGroupID of the item

GetOptionalContentConfigOrderItemLabel

Version history

This function was introduced in Quick PDF Library version 7.21.

Description

Returns the label text for an item in the order array of the specified optional content configuration dictionary.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetOptionalContentConfigOrderItemLabel(
  OptionalContentConfigID, ItemIndex: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetOptionalContentConfigOrderItemLabel(
  OptionalContentConfigID As Long, ItemIndex As Long) As String
```

DLL

```
wchar_t * DPLGetOptionalContentConfigOrderItemLabel(int InstanceID,
  int OptionalContentConfigID, int ItemIndex);
```

Parameters

OptionalContentConfigID	The first default optional content configuration dictionary has an ID of 1. Higher numbers are used for other optional content configuration dictionaries.
ItemIndex	The index number of the item in the order array. The first item has an index number of 1 and the last item has an index equal to the value returned by the **GetOptionalContentConfigOrderCount** function.

GetOptionalContentConfigOrderItemLevel
Content Streams and Optional Content Groups

Version history

This function was introduced in Quick PDF Library version 7.21.

Description

Returns the hierarchical level for an item in the order array of the specified optional content configuration dictionary.
The first item has a level of 1.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetOptionalContentConfigOrderItemLevel(
  OptionalContentConfigID, ItemIndex: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetOptionalContentConfigOrderItemLevel(
  OptionalContentConfigID As Long, ItemIndex As Long) As Long
```

DLL

```
int DPLGetOptionalContentConfigOrderItemLevel(int InstanceID,
  int OptionalContentConfigID, int ItemIndex);
```

Parameters

OptionalContentConfigID	The first default optional content configuration dictionary has an ID of 1. Higher numbers are used for other optional content configuration dictionaries.
ItemIndex	The index number of the item in the order array. The first item has an index number of 1 and the last item has an index equal to the value returned by the **GetOptionalContentConfigOrderCount** function.

Return values

0	The specified item could not be found
Non-zero	The level of the specified item

GetOptionalContentConfigOrderItemType

Version history

This function was introduced in Quick PDF Library version 7.21.

Description

Returns the item type for an item in the order array of the specified optional content configuration dictionary.
Items are either optional content groups or text labels.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetOptionalContentConfigOrderItemType(
  OptionalContentConfigID, ItemIndex: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetOptionalContentConfigOrderItemType(
  OptionalContentConfigID As Long, ItemIndex As Long) As Long
```

DLL

```
int DPLGetOptionalContentConfigOrderItemType(int InstanceID,
  int OptionalContentConfigID, int ItemIndex);
```

Parameters

OptionalContentConfigID	The first default optional content configuration dictionary has an ID of 1. Higher numbers are used for other optional content configuration dictionaries.
ItemIndex	The index number of the item in the order array. The first item has an index number of 1 and the last item has an index equal to the value returned by the **GetOptionalContentConfigOrderCount** function.

Return values

0	The specified item could not be found
1	The specified item is an optional content group. The **GetOptionalContentConfigOrderItemID** function can be used to detemine the OptionalContentGroupID.
2	The specified item is a text label.

GetOptionalContentConfigState

Version history

This function was introduced in Quick PDF Library version 7.21.

Description

This function is used to determine the state of an optional content group as defined by the specified optional content configuration dictionary.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetOptionalContentConfigState(
  OptionalContentConfigID, OptionalContentGroupID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetOptionalContentConfigState(
  OptionalContentConfigID As Long,
  OptionalContentGroupID As Long) As Long
```

DLL

```
int DPLGetOptionalContentConfigState(int InstanceID,
  int OptionalContentConfigID, int OptionalContentGroupID);
```

Parameters

OptionalContentConfigID	The first default optional content configuration dictionary has an ID of 1. Higher numbers are used for other optional content configuration dictionaries.
OptionalContentGroupID	An ID returned by the **NewOptionalContentGroup**, **GetOptionalContentGroupID** or **GetOptionalContentConfigOrderItemID** functions

Return values

0	The OptionalContentConfigID parameter or the OptionalContentGroupID parameter is not valid.
1	The state of the optional content group is set to ON when this optional content configuration dictionary is applied.
2	The state of the optional content group is set to OFF when this optional content configuration dictionary is applied.
3	The state of the optional content group is not changed when this optional content configuration dictionary is applied.

GetOptionalContentGroupID

Description

Returns the ID of the optional content group with the specified index.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetOptionalContentGroupID(
  Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetOptionalContentGroupID(
  Index As Long) As Long
```

DLL

```
int DPLGetOptionalContentGroupID(int InstanceID, int Index);
```

Parameters

Index	The index of the optional content group. The first group has an index of 1. Use the **OptionalContentGroupCount** function to determine the number of optional content groups in the document.

Return values

0	The Index parameter was out of range

GetOptionalContentGroupName

Description

Returns the name of the specified optional content group.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetOptionalContentGroupName(
  OptionalContentGroupID: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetOptionalContentGroupName(
  OptionalContentGroupID As Long) As String
```

DLL

```
wchar_t * DPLGetOptionalContentGroupName(int InstanceID,
  int OptionalContentGroupID);
```

Parameters

OptionalContentGroupID	An ID returned by the **NewOptionalContentGroup**, **GetOptionalContentGroupID** or **GetOptionalContentConfigOrderItemID** functions

GetOptionalContentGroupPrintable

Version history

This function was introduced in Quick PDF Library version 7.26.

Description

Returns the printable state of the specified optional content group.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetOptionalContentGroupPrintable(
  OptionalContentGroupID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetOptionalContentGroupPrintable(
  OptionalContentGroupID As Long) As Long
```

DLL

```
int DPLGetOptionalContentGroupPrintable(int InstanceID,
  int OptionalContentGroupID);
```

Parameters

OptionalContentGroupID	An ID returned by the **NewOptionalContentGroup**, **GetOptionalContentGroupID** or **GetOptionalContentConfigOrderItemID** functions

Return values

0	The specified optional content group is not printable
1	The specified optional content group is printable

GetOptionalContentGroupVisible

Version history

This function was introduced in Quick PDF Library version 7.26.

Description

Returns the visible state of the specified optional content group.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetOptionalContentGroupVisible(
  OptionalContentGroupID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetOptionalContentGroupVisible(
  OptionalContentGroupID As Long) As Long
```

DLL

```
int DPLGetOptionalContentGroupVisible(int InstanceID,
  int OptionalContentGroupID);
```

Parameters

OptionalContentGroupID	An ID returned by the **NewOptionalContentGroup**, **GetOptionalContentGroupID** or **GetOptionalContentConfigOrderItemID** functions

Return values

0	The specified optional content group is not visible
1	The specified optional content group is visible

GetOrigin

Measurement and coordinate units

Description

Returns the co-ordinate system origin as set with the **SetOrigin** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetOrigin: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetOrigin As Long
```

DLL

```
int DPLGetOrigin(int InstanceID);
```

GetOutlineActionID

Version history

This function was introduced in Quick PDF Library version 7.16.

Description

This function will return an ActionID if the specified outline has an action dictionary.
The ActionID can be used with the **GetActionType** function and can also be compared to the values returned by **GetAnnotActionID** to determine if an outline action is shared with an annotation action.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetOutlineActionID(
  OutlineID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetOutlineActionID(
  OutlineID As Long) As Long
```

DLL

```
int DPLGetOutlineActionID(int InstanceID, int OutlineID);
```

Parameters

OutlineID	The ID of the outline as returned by the **NewOutline** function. Alternatively, use the **GetOutlineID** function to get a valid outline ID.

GetOutlineColor

Version history

This function was introduced in Quick PDF Library version 7.12.

Description

Returns the color component of the outline as a value between 0 and 1.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetOutlineColor(OutlineID,
  ColorComponent: Integer): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetOutlineColor(
  OutlineID As Long, ColorComponent As Long) As Double
```

DLL

```
double DPLGetOutlineColor(int InstanceID, int OutlineID,
  int ColorComponent);
```

Parameters

OutlineID	The ID of the outline as returned by the **NewOutline** function. Alternatively, use the **GetOutlineID** function to get a valid outline ID.
ColorComponent	The component of the color: 0 = Red 1 = Green 2 = Blue

GetOutlineDest

Outlines

Description

Retrieves information about the destination the specified outline links to.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetOutlineDest(OutlineID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetOutlineDest(
  OutlineID As Long) As Long
```

DLL

```
int DPLGetOutlineDest(int InstanceID, int OutlineID);
```

Parameters

OutlineID	The ID of the outline as returned by the **NewOutline** function. Alternatively, use the **GetOutlineID** function to get a valid outline ID.

Return values

0	The outline does not have a valid destination or the outline could not be found
Non-zero	A destination ID (or DestID) that can be used with the **GetDestPage**, **GetDestType** and **GetDestValue** functions

GetOutlineID

Version history

This function was renamed in Quick PDF Library version 7.11.
The function name in earlier versions was OutlineID.

Description

Returns the Outline ID of the outline item (bookmark) with the specified index. The first outline item has an index of 1.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetOutlineID(Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetOutlineID(
  Index As Long) As Long
```

DLL

```
int DPLGetOutlineID(int InstanceID, int Index);
```

Parameters

Index	The index of the outline item to retrieve the ID of. The first outline item has an index of 1.

GetOutlineJavaScript

Version history

This function was introduced in Quick PDF Library version 7.12.

Description

Returns the JavaScript associated with the outline, if any.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetOutlineJavaScript(
  OutlineID: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetOutlineJavaScript(
  OutlineID As Long) As String
```
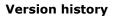
DLL

```
wchar_t * DPLGetOutlineJavaScript(int InstanceID, int OutlineID);
```

Parameters

OutlineID	The ID of the outline as returned by the **NewOutline** function. Alternatively, use the **GetOutlineID** function to get a valid outline ID.

GetOutlineObjectNumber

Version history

This function was introduced in Quick PDF Library version 7.22.

Description

Returns the PDF object number of the specified outline item.

This function is for advanced use only.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetOutlineObjectNumber(
  OutlineID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetOutlineObjectNumber(
  OutlineID As Long) As Long
```

DLL

```
int DPLGetOutlineObjectNumber(int InstanceID, int OutlineID);
```

Parameters

OutlineID	The ID of the outline as returned by the **NewOutline** function. Alternatively, use the **GetOutlineID** function to get a valid outline ID.

GetOutlineOpenFile

Description

Returns the file name that the outline links to, if any.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetOutlineOpenFile(
  OutlineID: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetOutlineOpenFile(
  OutlineID As Long) As String
```

DLL

```
wchar_t * DPLGetOutlineOpenFile(int InstanceID, int OutlineID);
```

Parameters

OutlineID	The ID of the outline as returned by the **NewOutline** function. Alternatively, use the **GetOutlineID** function to get a valid outline ID.

GetOutlinePage

Description

Returns the page number that the outline links to.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetOutlinePage(OutlineID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetOutlinePage(
  OutlineID As Long) As Long
```

DLL

```
int DPLGetOutlinePage(int InstanceID, int OutlineID);
```

Parameters

OutlineID	The ID of the outline as returned by the **NewOutline** function. Alternatively, use the **GetOutlineID** function to get a valid outline ID.

GetOutlineStyle

Version history

This function was introduced in Quick PDF Library version 7.12.

Description

Returns the style of the outline.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetOutlineStyle(
  OutlineID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetOutlineStyle(
  OutlineID As Long) As Long
```

DLL

```
int DPLGetOutlineStyle(int InstanceID, int OutlineID);
```

Parameters

OutlineID	The ID of the outline as returned by the **NewOutline** function. Alternatively, use the **GetOutlineID** function to get a valid outline ID.

Return values

0	Normal
1	Italic
2	Bold
3	Bold Italic

GetOutlineWebLink

Description

Returns the web link (internet URL) that the outline links to, if any.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetOutlineWebLink(
  OutlineID: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetOutlineWebLink(
  OutlineID As Long) As String
```

DLL

```
wchar_t * DPLGetOutlineWebLink(int InstanceID, int OutlineID);
```

Parameters

OutlineID	The ID of the outline as returned by the **NewOutline** function. Alternatively, use the **GetOutlineID** function to get a valid outline ID.

GetPageBox

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Returns the dimensions of the selected page's boundary rectangles.

The MediaBox represents the physical medium of the page.

The CropBox represents the visible region of the page, the contents will be clipped to this region.

The BleedBox is similar to the CropBox, but is the rectangle used in a production environment.

The TrimBox indicates the intended dimensions of the finished page after trimming, and the ArtBox defines the extent of the page's meaningful content as intended by the page's creator.

If the document does not have a CropBox but it does have a MediaBox then the CropBox will be the same as the MediaBox. If the document does not have any of the other boxes this function will return the values from the CropBox.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetPageBox(BoxType,
  Dimension: Integer): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetPageBox(BoxType As Long,
  Dimension As Long) As Double
```

DLL

```
double DPLGetPageBox(int InstanceID, int BoxType, int Dimension);
```

Parameters

BoxType	1 = MediaBox
	2 = CropBox
	3 = BleedBox
	4 = TrimBox
	5 = ArtBox
Dimension	0 = Left
	1 = Top
	2 = Width
	3 = Height
	4 = Right
	5 = Bottom

GetPageColorSpaces

Version history

This function was introduced in Quick PDF Library version 8.14.

Description

Returns a CSV string containing the list of color spaces defined in the resource tree of the selected page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetPageColorSpaces(
  Options: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetPageColorSpaces(
  Options As Long) As String
```

DLL

```
wchar_t * DPLGetPageColorSpaces(int InstanceID, int Options);
```

Parameters

Options	This parameter should be set to 0.

GetPageContentToString

Version history

This function was renamed in Quick PDF Library version 8.11.
The function name in earlier versions was GetPageContent.

Description

This function returns the PDF page description commands which make up the content of the selected page. This is for advanced use only, and will probably be meaningless unless you have an understanding of the Adobe PDF specification.

Previous versions of Quick PDF Library only returned the content of the selected content stream part.

From version 8.11 this function returns the content of the entire page and the **GetContentStreamToString** function can be used to retrieve the PDF page description commands of the content stream part selected with the **SelectContentStream** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetPageContentToString: AnsiString;
```

DLL

```
char * DPLGetPageContentToString(int InstanceID);
```

GetPageContentToVariant

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

This function returns the PDF page description commands which make up the content of the selected page. This is for advanced use only, and will probably be meaningless unless you have an understanding of the Adobe PDF specification.

This function returns the content of the entire page regardless of the number of content stream parts.

Syntax

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetPageContentToVariant As Variant
```

GetPageImageList

Image handling, Page properties

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Version history

This function was introduced in Quick PDF Library version 8.13.

Description

This function finds all the images on the selected page and returns an ImageListID that can be used with the **GetImageListCount**, **GetImageListItemIntProperty**, **GetImageListItemDblProperty**, **GetImageListItemDataToString**, **GetImageListItemDataToVariant** and **SaveImageListItemDataToFile** functions.

As of version 10.13 will include Inline images but the ImageID will be 0 for any inline image which means that any inline images cannot used with ReplaceImage or ClearImage functions.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetPageImageList(Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetPageImageList(
  Options As Long) As Long
```

DLL

```
int DPLGetPageImageList(int InstanceID, int Options);
```

Parameters

Options	Reserved for future use, should be set to 0.

Return values

0	The images on the page could not be enumerated.
Non-zero	An ImageListID value

GetPageJavaScript

Description

Retrieves the JavaScript linked to the specified page event.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetPageJavaScript(
  ActionType: WideString): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetPageJavaScript(
  ActionType As String) As String
```

DLL

```
wchar_t * DPLGetPageJavaScript(int InstanceID, wchar_t * ActionType);
```

Parameters

ActionType	Retrieves the JavaScript linked to this action: "O" = (capital letter O) This event occurs when the page is opened "C" = This event occurs when the page is closed

GetPageLGIDictContent

Version history

This function was introduced in Quick PDF Library version 7.15.

Description

Returns the content of the specified LGIDict dictionary on the selected page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetPageLGIDictContent(
  DictIndex: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetPageLGIDictContent(
  DictIndex As Long) As String
```

DLL

```
wchar_t * DPLGetPageLGIDictContent(int InstanceID, int DictIndex);
```

Parameters

DictIndex   The index of the dictionary. The first dictionary has an index of 1. Use the **LGIDictCount** function to determine the total number of LGIDict dictionaries attached to the selected page.

GetPageLGIDictCount

Version history

This function was introduced in Quick PDF Library version 7.15.

Description

Returns the number of LGIDict dictionaries attached to the selected page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetPageLGIDictCount: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetPageLGIDictCount As Long
```

DLL

```
int DPLGetPageLGIDictCount(int InstanceID);
```

GetPageLabel

Description

Returns the page label for the specified page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetPageLabel(Page: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetPageLabel(
  Page As Long) As String
```

DLL

```
wchar_t * DPLGetPageLabel(int InstanceID, int Page);
```

Parameters

Page	The number of the page to retrieve the page number of

GetPageLayout

Description

Returns the initial page layout of the selected document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetPageLayout: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetPageLayout As Long
```

DLL

```
int DPLGetPageLayout(int InstanceID);
```

Return values

0	Single page
1	One column
2	Two columns, odd-numbered pages on the left
3	Two columns, odd-numbered pages on the right
4	Two pages, odd-numbered pages on the left
5	Two pages, odd-numbered pages on the right
6	No preference set in document

GetPageMetricsToString

Version history

This function was introduced in Quick PDF Library version 9.14.

Description

Returns the dimensions (MediaBox and CropBox) and rotation of the specified page range in the document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetPageMetricsToString(StartPage, EndPage,
  Options: Integer): AnsiString;
```

DLL

```
char * DPLGetPageMetricsToString(int InstanceID, int StartPage,
  int EndPage, int Options);
```

Parameters

StartPage	The first page in the range
EndPage	The last page in the range
Options	1 = Binary output Nine double values per page are streamed in a continuous array. For each page there are the elements of the MediaBox, the elements of the CropBox and the value of the Rotation entry. 2 = Text output The values are displayed in text format, separated by tab (char 9) characters and with CRLF after each page.

GetPageMode

Description

Returns the initial page mode of the document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetPageMode: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetPageMode As Long
```

DLL

```
int DPLGetPageMode(int InstanceID);
```

Return values

0	Normal view
1	Show the outlines pane
2	Show the thumbnails pane
3	Show the document in full screen mode
4	Optional content group panel visible
5	Attachments panel visible

GetPageText

Description

This function provides two different methods for extracting text from the selected page, and presents the results in a variety of formats.

The **SetTextExtractionWordGap**, **SetTextExtractionOptions** and **SetTextExtractionArea** functions can be used to adjust the text extraction process.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetPageText(
   ExtractOptions: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetPageText(
   ExtractOptions As Long) As String
```

DLL

```
wchar_t * DPLGetPageText(int InstanceID, int ExtractOptions);
```

Parameters

ExtractOptions	Using the standard text extraction algorithm:
	0 = Extract text in human readable format
	1 = Deprecated
	2 = Return a CSV string including font, color, size and position of each piece of text on the page
	Using the more accurate but slower text extraction algorithm:
	3 = Return a CSV string for each piece of text on the page with the following format:
	Font Name, Text Color, Text Size, X1, Y1, X2, Y2, X3, Y3, X4, Y4, Text
	The co-ordinates are the four points bounding the text, measured using the units set with the **SetMeasurementUnits** function and the origin set with the **SetOrigin** function. Co-ordinate order is anti-clockwise with the bottom left corner first.
	4 = Similar to option 3, but individual words are returned, making searching for words easier
	5 = Similar to option 3 but character widths are output after each block of text
	6 = Similar to option 4 but character widths are output after each line of text
	7 = Extract text in human readable format with improved accuracy compared to option 0
	8 = Similar output format as option 0 but using the more accurate algorithm. Returns unformatted lines.

Return values

The text of the selected page, or an empty string if a problem occurred. Lines are separated with CR-LF characters.

GetPageUserUnit

Description

Returns the UserUnit for the page scaling. See **SetPageUserUnit** for a description of this value.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetPageUserUnit: Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetPageUserUnit As Double
```

DLL

```
double DPLGetPageUserUnit(int InstanceID);
```

Return values

UserUnit   The value of UserUnit set in the PDF. Default = 1.0

GetPageViewPortCount

Version history

This function was introduced in Quick PDF Library version 8.13.

Description

Returns the number of viewports defined for the selected page.

The **GetPageViewPortID** function can be used to obtain a ViewPortID that can be used with the **GetViewPortName** and **GetViewPortMeasureDict** functions.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetPageViewPortCount: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetPageViewPortCount As Long
```

DLL

```
int DPLGetPageViewPortCount(int InstanceID);
```

GetPageViewPortID

Version history

This function was introduced in Quick PDF Library version 8.13.

Description

Returns a ViewPortID value for the specified viewport of the selected page.

This value can be used with the **GetViewPortName** and **GetViewPortMeasureDict** functions.

Use the **GetPageViewPortCount** function to determine the number of viewports on the page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetPageViewPortID(Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetPageViewPortID(
  Index As Long) As Long
```

DLL

```
int DPLGetPageViewPortID(int InstanceID, int Index);
```

Parameters

Index	The index of the viewport. The first viewport on the page has an index value of 1.

Return values

0	The view port at the specified index could not be found
Non-zero	A ViewPortID value

GetParentOutline

Description

Returns the ID of the outline that is the parent of the specified outline.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetParentOutline(
  OutlineID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetParentOutline(
  OutlineID As Long) As Long
```

DLL

```
int DPLGetParentOutline(int InstanceID, int OutlineID);
```

Parameters

OutlineID	The ID of the outline item to work with. This ID is returned by the **NewOutline** or **NewStaticOutline** functions, or retrieved with the **GetOutlineID** function or Get*Outline functions.

GetPrevOutline

Description

Returns the ID of the outline that is above the specified outline at the same level.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetPrevOutline(OutlineID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetPrevOutline(
   OutlineID As Long) As Long
```

DLL

```
int DPLGetPrevOutline(int InstanceID, int OutlineID);
```

Parameters

OutlineID	The ID of the outline item to work with. This ID is returned by the **NewOutline** or **NewStaticOutline** functions, or retrieved with the **GetOutlineID** function or Get*Outline functions.

GetPrintPreviewBitmapToString

Version history

This function was introduced in Quick PDF Library version 9.16.

Description

Returns a binary string containing a BMP image representing a preview of how printing will look.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetPrintPreviewBitmapToString(
  PrinterName: WideString; PreviewPage, PrintOptions, MaxDimension,
  PreviewOptions: Integer): AnsiString;
```

DLL

```
char * DPLGetPrintPreviewBitmapToString(int InstanceID,
  wchar_t * PrinterName, int PreviewPage, int PrintOptions,
  int MaxDimension, int PreviewOptions);
```

Parameters

PrinterName	The name of the printer to use for printing. This is the name that appears in the Windows Print Manager. Use the **GetPrinterNames** function to return a list of valid printers on the system. A value returned by the **NewCustomPrinter** function can also be used here.
PreviewPage	The page number to preview
PrintOptions	Use the **PrintOptions** function to obtain a value for this parameter
MaxDimension	The maximum width or height of the preview bitmap
PreviewOptions	Reserved for future use, should be set to zero.

GetPrintPreviewBitmapToVariant

Version history

This function was introduced in Quick PDF Library version 9.16.

Description

Returns a byte array containing a BMP image representing a preview of how printing will look.

Syntax

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetPrintPreviewBitmapToVariant(
  PrinterName As String, PreviewPage As Long,
  PrintOptions As Long, MaxDimension As Long,
  PreviewOptions As Long) As Variant
```

Parameters

PrinterName	The name of the printer to use for printing. This is the name that appears in the Windows Print Manager. Use the **GetPrinterNames** function to return a list of valid printers on the system. A value returned by the **NewCustomPrinter** function can also be used here.
PreviewPage	The page number to preview
PrintOptions	Use the **PrintOptions** function to obtain a value for this parameter
MaxDimension	The maximum width or height of the preview bitmap
PreviewOptions	Reserved for future use, should be set to zero.

GetPrinterBins

Description

This function returns a string containing the bin numbers and names for all the bins (paper trays) available for the specified printer. The string returned contains a line of text for each bin, the lines of text are separated with CR/LF characters. Each line contains a numeric bin number, a comma, and the name of the bin, in double quotes. The bin numbers can be used with the **SetupCustomPrinter** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetPrinterBins(
  PrinterName: WideString): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetPrinterBins(
  PrinterName As String) As String
```

DLL

```
wchar_t * DPLGetPrinterBins(int InstanceID, wchar_t * PrinterName);
```

Parameters

PrinterName	The name of the printer to query. This is the name that appears in the Windows Print Manager. Use the **GetPrinterNames** function to return a list of valid printers on the system. A value returned by the **NewCustomPrinter** function can also be used here.

GetPrinterDevModeToString

Version history

This function was introduced in Quick PDF Library version 8.12.

Description

Returns a binary string containing the DEVMODE structure for the specified printer.

Use the **SetPrinterDevModeFromString** function to apply this DEVMODE structure during the printing process.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetPrinterDevModeToString(
    PrinterName: WideString): AnsiString;
```

DLL

```
char * DPLGetPrinterDevModeToString(int InstanceID, wchar_t * PrinterName);
```

Parameters

PrinterName	The name of the printer to use for printing. This is the name that appears in the Windows Print Manager. Use the **GetPrinterNames** function to return a list of valid printers on the system. A value returned by the **NewCustomPrinter** function can also be used here.

GetPrinterDevModeToVariant

Version history

> This function was introduced in Quick PDF Library version 8.12.

Description

> Returns a variant byte array containing the DEVMODE structure for the specified printer.
>
> Use the **SetPrinterDevModeFromVariant** function to apply this DEVMODE structure during the printing process.

Syntax

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetPrinterDevModeToVariant(
  PrinterName As String) As Variant
```

Parameters

PrinterName	The name of the printer to use for printing. This is the name that appears in the Windows Print Manager. Use the **GetPrinterNames** function to return a list of valid printers on the system. A value returned by the **NewCustomPrinter** function can also be used here.

GetPrinterMediaTypes

Version history

This function was introduced in Quick PDF Library version 8.14.

Description

This function returns a string containing the media type numbers and names for all the media types available for the specified printer. The string returned contains a line of text for each media type, the lines of text are separated with CR/LF characters. Each line contains a numeric media type number, a comma, and the name of the media type, in double quotes.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetPrinterMediaTypes(
  PrinterName: WideString): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetPrinterMediaTypes(
  PrinterName As String) As String
```

DLL

```
wchar_t * DPLGetPrinterMediaTypes(int InstanceID, wchar_t * PrinterName);
```

Parameters

PrinterName	The name of the printer to query. This is the name that appears in the Windows Print Manager. Use the **GetPrinterNames** function to return a list of valid printers on the system. A value returned by the **NewCustomPrinter** function can also be used here.

GetPrinterNames

Description

Returns a CSV string containing the names of all the available printers on the system. The result is the cached list that was enumerated when the app was started. The new **GetLatestPrinterNames** function returns the latest list of printers.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetPrinterNames: WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetPrinterNames As String
```

DLL

```
wchar_t * DPLGetPrinterNames(int InstanceID);
```

GetRenderScale

Rendering and printing

Version history

This function was introduced in Quick PDF Library version 7.22.

Description

Returns the render scale as set by the **SetRenderScale** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetRenderScale: Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetRenderScale As Double
```

DLL

```
double DPLGetRenderScale(int InstanceID);
```

GetSignProcessByteRange

Version history

This function was introduced in Quick PDF Library version 9.15.

Description

Returns an element of the byte range array of a passthough digital signature.

The values should be handled as 32-bit unsigned integers with two values combined to form a 64-bit file position.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetSignProcessByteRange(SignProcessID,
  ArrayPosition: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetSignProcessByteRange(
  SignProcessID As Long, ArrayPosition As Long) As Long
```

DLL

```
int DPLGetSignProcessByteRange(int InstanceID, int SignProcessID,
  int ArrayPosition);
```

Parameters

SignProcessID	A value returned by the **NewSignProcessFromFile**, **NewSignProcessFromStream** or **NewSignProcessFromString** functions.
ArrayPosition	1 = ByteArray[0] low 32-bits 2 = ByteArray[1] low 32-bits 3 = ByteArray[2] low 32-bits 4 = ByteArray[3] low 32-bits 5 = ByteArray[0] high 32-bits 6 = ByteArray[1] high32-bits 7 = ByteArray[2] high 32-bits 8 = ByteArray[3] high 32-bits

GetSignProcessResult

Version history

This function was introduced in Quick PDF Library version 9.14.

Description

Returns the signing result of a digital signature process.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetSignProcessResult(
  SignProcessID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetSignProcessResult(
  SignProcessID As Long) As Long
```

DLL

```
int DPLGetSignProcessResult(int InstanceID, int SignProcessID);
```

Parameters

SignProcessID	A value returned by the **NewSignProcessFromFile**, **NewSignProcessFromStream** or **NewSignProcessFromString** functions.

Return values

1	The file was signed successfully
2	Input PDF not found
3	Input PDF cannot be read
4	Input PDF password incorrect
5	Certificate file not found
6	Certificate file is invalid
7	Incorrect certificate password
8	Unknown certificate format
9	No private key found in certificate file
10	Could not write output file
11	Could not apply signature
12	The signature field name was blank

GetStringListCount

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Returns the number of strings in the specified string list.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetStringListCount(
  StringListID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetStringListCount(
  StringListID As Long) As Long
```

DLL

```
int DPLGetStringListCount(int InstanceID, int StringListID);
```

Parameters

StringListID	The ID of the string list as returned by the **CheckFileCompliance** function.

Return values

0	There are no strings in the specified string list.
Non-zero	The number of strings in the list.

GetStringListItem

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Returns an item from the specified string list.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetStringListItem(StringListID,
  ItemIndex: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetStringListItem(
  StringListID As Long, ItemIndex As Long) As String
```

DLL

```
wchar_t * DPLGetStringListItem(int InstanceID, int StringListID,
  int ItemIndex);
```

Parameters

StringListID	The ID of the string list as returned by the **CheckFileCompliance** function.
ItemIndex	The index of the item to return. The first item in the list has an index value of 1. The last item in the list has an index value equal to the return value of the **GetStringListCount** function.

GetTabOrderMode

Form fields, Annotations and hotspot links

Version history

This function was introduced in Quick PDF Library version 9.16.

Description

This function returns the current tabbing order for all the annotations and formfields on the currently selected page.

If you use **SetFormFieldTabOrder** then you should set the tabbing order to 'S'tructure mode for the required pages.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetTabOrderMode: WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetTabOrderMode As String
```

DLL

```
wchar_t * DPLGetTabOrderMode(int InstanceID);
```

Return values

'S'	Structure mode (The order the Annots are defined)
'R'	Row mode (Left to right, top to bottom order)
'C'	Column mode (Top to bottom, left to right order)
'' (Empty String)	No tabbing order has been defined

GetTableCellDblProperty

Version history

This function was introduced in Quick PDF Library version 7.14.

Description

Returns a numeric property of the specified table cell.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetTableCellDblProperty(TableID, RowNumber,
  ColumnNumber, Tag: Integer): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetTableCellDblProperty(
  TableID As Long, RowNumber As Long, ColumnNumber As Long,
  Tag As Long) As Double
```

DLL

```
double DPLGetTableCellDblProperty(int InstanceID, int TableID,
  int RowNumber, int ColumnNumber, int Tag);
```

Parameters

TableID	A TableID returned by the **CreateTable** function
RowNumber	The the row number of the cell. Top row is row number 1.
ColumnNumber	The the column number of the cell. Left most column is column number 1.
Tag	101 to 104 = Left, top, width and height of cell 105 = Text size 106 = Red or cyan component of the background color 107 = Green or magenta component of the background color 108 = Blue or yellow component of the background color 109 = Black component of the background color 110 = Red or cyan component of the text color 111 = Green or magenta component of the text color 112 = Blue or yellow component of the text color 113 = Black component of the text color 114 to 117 = Red or cyan component of the left, top, right and bottom border 118 to 121 = Green or magenta component of the left, top, right and bottom border 122 to 125 = Blue or yellow component of the left, top, right and bottom border 126 to 129 = Black component of the left, top, right and bottom border 130 to 133 = Padding of the edge next to the left, top, right and bottom border 134 to 137 = Width of the left, top, right and bottom border

GetTableCellIntProperty

Version history

This function was introduced in Quick PDF Library version 7.14.

Description

Returns an integer property of the specified table cell.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetTableCellIntProperty(TableID, RowNumber,
  ColumnNumber, Tag: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetTableCellIntProperty(
  TableID As Long, RowNumber As Long, ColumnNumber As Long,
  Tag As Long) As Long
```

DLL

```
int DPLGetTableCellIntProperty(int InstanceID, int TableID, int RowNumber,
  int ColumnNumber, int Tag);
```

Parameters

TableID	A TableID returned by the **CreateTable** function
RowNumber	The the row number of the cell. Top row is row number 1.
ColumnNumber	The the column number of the cell. Left most column is column number 1.
Tag	201 = Cell alignment (see the **SetTableCellAlignment** function) 202 = Merged cell row span 203 = Merged cell column span 204 = Number of color components in the background color (3 for RGB, 4 for CMYK) 205 = Number of color components in the text color (3 for RGB, 4 for CMYK) 206 to 209 = Number of color components in the left, top, right and bottom border color (3 for RGB, 4 for CMYK)

GetTableCellStrProperty

Page layout

Version history

This function was introduced in Quick PDF Library version 7.14.

Description

Returns a string property of the specified table cell.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetTableCellStrProperty(TableID, RowNumber,
  ColumnNumber, Tag: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetTableCellStrProperty(
  TableID As Long, RowNumber As Long, ColumnNumber As Long,
  Tag As Long) As String
```

DLL

```
wchar_t * DPLGetTableCellStrProperty(int InstanceID, int TableID,
  int RowNumber, int ColumnNumber, int Tag);
```

Parameters

TableID	A TableID returned by the **CreateTable** function
RowNumber	The the row number of the cell. Top row is row number 1.
ColumnNumber	The the column number of the cell. Left most column is column number 1.
Tag	301 = Cell contents

GetTableColumnCount

Version history

This function was introduced in Quick PDF Library version 7.16.

Description

Returns the number of columns in the specified table.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetTableColumnCount(
  TableID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetTableColumnCount(
  TableID As Long) As Long
```

DLL

```
int DPLGetTableColumnCount(int InstanceID, int TableID);
```

Parameters

TableID   A TableID returned by the **CreateTable** function

GetTableLastDrawnRow

Version history

This function was introduced in Quick PDF Library version 7.14.

Description

Returns the row number of the last row that was drawn onto the page by the **DrawTableRows** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetTableLastDrawnRow(
  TableID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetTableLastDrawnRow(
  TableID As Long) As Long
```

DLL

```
int DPLGetTableLastDrawnRow(int InstanceID, int TableID);
```

Parameters

TableID	A TableID returned by the **CreateTable** function

Return values

0	No rows from the specified table have been drawn
Non-zero	The row number of the last drawn row. The top row is row number 1.

GetTableRowCount

Version history

This function was introduced in Quick PDF Library version 7.16.

Description

Returns the number of rows in the specified table.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetTableRowCount(TableID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetTableRowCount(
  TableID As Long) As Long
```

DLL

```
int DPLGetTableRowCount(int InstanceID, int TableID);
```

Parameters

TableID   A TableID returned by the **CreateTable** function

GetTempPath

Description

Retrieves the current setting for the folder that will be used to store temporary files generated by functions such as **MergeFileList**.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetTempPath: WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetTempPath As String
```

DLL

```
wchar_t * DPLGetTempPath(int InstanceID);
```

GetTextAscent

Text, Fonts, Page layout

Description

Returns the size of the selected font, measured from the baseline to the top of capital letters (without any accents).

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetTextAscent: Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetTextAscent As Double
```

DLL

```
double DPLGetTextAscent(int InstanceID);
```

Return values

The ascent of the selected font

GetTextBlockAsString

Version history

This function was introduced in Quick PDF Library version 11.12.

Description

Returns all the text block entries for a single text block as a formatted string delimited by CR/LF

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetTextBlockAsString(TextBlockListID,
  Index: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetTextBlockAsString(
  TextBlockListID As Long, Index As Long) As String
```

DLL

```
wchar_t * DPLGetTextBlockAsString(int InstanceID, int TextBlockListID,
  int Index);
```

Parameters

TextBlockListID	A value returned by the **ExtractPageTextBlocks** function
Index	The index of the text block. The first text block in the list has an index of 1.

Return values

TextBlockAsString	A formatted string of all available text block fields where each line is separate by a CR/LF. Here is a sample output string

CNT:4
FNT:Arial
SIZ:12
CLR:#000000
TX1:20
TY1:769.516
TX2:48.02
TY2:769.516
TX3:48.02
TY3:780.616
TX4:20
TY4:780.616
WID:8.004,6.672,6.672,6.672
TXT:Page

where CNT = char count, FNT = fontname, SIZ = Fontsize, CLR = color, TXx = X value for bounds point x, TYy = Y value for bounds y, WID = comma separated character widths, TXT = extracted text.

GetTextBlockBound

Version history

This function was introduced in Quick PDF Library version 9.13.

Description

Returns one of the bounds of the specified text block.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetTextBlockBound(TextBlockListID, Index,
  BoundIndex: Integer): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetTextBlockBound(
  TextBlockListID As Long, Index As Long,
  BoundIndex As Long) As Double
```

DLL

```
double DPLGetTextBlockBound(int InstanceID, int TextBlockListID,
  int Index, int BoundIndex);
```

Parameters

TextBlockListID	A value returned by the **ExtractPageTextBlocks** function
Index	The index of the text block. The first text block in the list has an index of 1.
BoundIndex	1 = Bottom left horizontal coordinate 2 = Bottom left vertical coordinate 3 = Bottom right horizontal coordinate 4 = Bottom right vertical coordinate 5 = Top right horizontal coordinate 6 = Top right vertical coordinate 7 = Top left horizontal coordinate 8 = Top left vertical coordinate

GetTextBlockCharWidth

Version history

This function was introduced in Quick PDF Library version 9.13.

Description

Returns the width of a particular character within the specified text block.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetTextBlockCharWidth(TextBlockListID,
  Index, CharIndex: Integer): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetTextBlockCharWidth(
  TextBlockListID As Long, Index As Long,
  CharIndex As Long) As Double
```

DLL

```
double DPLGetTextBlockCharWidth(int InstanceID, int TextBlockListID,
  int Index, int CharIndex);
```

Parameters

TextBlockListID	A value returned by the **ExtractPageTextBlocks** function
Index	The index of the text block. The first text block in the list has an index of 1.
CharIndex	The index of the character to retrieve the width of. The first character has an index of 1.

GetTextBlockColor

Version history

This function was introduced in Quick PDF Library version 9.13.

Description

Returns one component of the color of the text in the specified text block.

The color component value is returned as a value between 0 and 1.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetTextBlockColor(TextBlockListID, Index,
   ColorComponent: Integer): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetTextBlockColor(
   TextBlockListID As Long, Index As Long,
   ColorComponent As Long) As Double
```

DLL

```
double DPLGetTextBlockColor(int InstanceID, int TextBlockListID,
   int Index, int ColorComponent);
```

Parameters

TextBlockListID	A value returned by the **ExtractPageTextBlocks** function
Index	The index of the text block. The first text block in the list has an index of 1.
ColorComponent	For RGB: 1 = Red 2 = Green 3 = Blue For CMYK: 1 = Cyan 2 = Magenta 3 = Yellow 4 = Black

GetTextBlockColorType

Version history

This function was introduced in Quick PDF Library version 9.13.

Description

Returns the type of color of the text in the specified text block.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetTextBlockColorType(TextBlockListID,
  Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetTextBlockColorType(
  TextBlockListID As Long, Index As Long) As Long
```

DLL

```
int DPLGetTextBlockColorType(int InstanceID, int TextBlockListID,
  int Index);
```

Parameters

TextBlockListID	A value returned by the **ExtractPageTextBlocks** function
Index	The index of the text block. The first text block in the list has an index of 1.

Return values

3	RGB
4	CMYK

GetTextBlockCount

Text, Extraction

Version history

This function was introduced in Quick PDF Library version 9.13.

Description

Returns the number of text blocks in the specified text block list.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetTextBlockCount(
  TextBlockListID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetTextBlockCount(
  TextBlockListID As Long) As Long
```

DLL

```
int DPLGetTextBlockCount(int InstanceID, int TextBlockListID);
```

Parameters

TextBlockListID   A value returned by the **ExtractPageTextBlocks** function

GetTextBlockFontName

Version history

This function was introduced in Quick PDF Library version 9.13.

Description

Returns the font name of the text in the specified text block.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetTextBlockFontName(TextBlockListID,
  Index: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetTextBlockFontName(
  TextBlockListID As Long, Index As Long) As String
```

DLL

```
wchar_t * DPLGetTextBlockFontName(int InstanceID, int TextBlockListID,
  int Index);
```

Parameters

TextBlockListID	A value returned by the **ExtractPageTextBlocks** function
Index	The index of the text block. The first text block in the list has an index of 1.

GetTextBlockFontSize

Version history

This function was introduced in Quick PDF Library version 9.13.

Description

Returns the font size of the text in the specified text block.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetTextBlockFontSize(TextBlockListID,
  Index: Integer): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetTextBlockFontSize(
  TextBlockListID As Long, Index As Long) As Double
```

DLL

```
double DPLGetTextBlockFontSize(int InstanceID, int TextBlockListID,
  int Index);
```

Parameters

TextBlockListID	A value returned by the **ExtractPageTextBlocks** function
Index	The index of the text block. The first text block in the list has an index of 1.

GetTextBlockText

Version history

This function was introduced in Quick PDF Library version 9.13.

Description

Returns the text in the specified text block.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetTextBlockText(TextBlockListID,
  Index: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetTextBlockText(
  TextBlockListID As Long, Index As Long) As String
```

DLL

```
wchar_t * DPLGetTextBlockText(int InstanceID, int TextBlockListID,
  int Index);
```

Parameters

TextBlockListID	A value returned by the **ExtractPageTextBlocks** function
Index	The index of the text block. The first text block in the list has an index of 1.

GetTextBound

Description

Returns the bounding box of the font. This is the largest rectangle which can enclose every character of the font. The top and bottom are measured from the baseline of the font.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetTextBound(Edge: Integer): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetTextBound(
  Edge As Long) As Double
```

DLL

```
double DPLGetTextBound(int InstanceID, int Edge);
```

Parameters

Edge	The edge measurement to retrieve: 1 = Left 2 = Top 3 = Right 4 = Bottom

Return values

0	The edge specified was not valid
Non-zero	The specified edge measurement

GetTextDescent

Description

Returns the size of the selected font, measured from the baseline to the bottom of the tails of lowercase letters such as g and y.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetTextDescent: Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetTextDescent As Double
```

DLL

```
double DPLGetTextDescent(int InstanceID);
```

Return values

The descent of the selected font

GetTextHeight

Description

Returns the height of the selected font. This is the sum of **GetTextBound**(2) and -**GetTextBound** (4).

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetTextHeight: Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetTextHeight As Double
```

DLL

```
double DPLGetTextHeight(int InstanceID);
```

Return values

The height of the selected font

GetTextSize

Description

Retrieves the current text size.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetTextSize: Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetTextSize As Double
```

DLL

```
double DPLGetTextSize(int InstanceID);
```

GetTextWidth

Description

Calculate the width of the specified text, based on the selected font and font size.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetTextWidth(Text: WideString): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetTextWidth(
  Text As String) As Double
```

DLL

```
double DPLGetTextWidth(int InstanceID, wchar_t * Text);
```

Parameters

Text	The text to determine the width for

Return values

	The width of the specified text

GetUnicodeCharactersFromEncoding

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Returns a string containing all the Unicode characters from the specified encoding.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetUnicodeCharactersFromEncoding(
  Encoding: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetUnicodeCharactersFromEncoding(
  Encoding As Long) As String
```

DLL

```
wchar_t * DPLGetUnicodeCharactersFromEncoding(int InstanceID,
  int Encoding);
```

Parameters

Encoding	2 = WinAnsiEncoding

GetViewPortBBox

Version history

This function was introduced in Quick PDF Library version 8.14.

Description

Returns details of the BBox entry of a viewport dictionary.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetViewPortBBox(ViewPortID,
  Dimension: Integer): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetViewPortBBox(
  ViewPortID As Long, Dimension As Long) As Double
```

DLL

```
double DPLGetViewPortBBox(int InstanceID, int ViewPortID, int Dimension);
```

Parameters

ViewPortID	A value returned by the **GetPageViewPortID** function
Dimension	0 = Left 1 = Top 2 = Width 3 = Height 4 = Right 5 = Bottom

GetViewPortMeasureDict

Version history

This function was introduced in Quick PDF Library version 8.13.

Description

Returns the measurement dictionary of the specified viewport as a MeasureDictID value.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetViewPortMeasureDict(
  ViewPortID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetViewPortMeasureDict(
  ViewPortID As Long) As Long
```

DLL

```
int DPLGetViewPortMeasureDict(int InstanceID, int ViewPortID);
```

Parameters

ViewPortID	A value returned by the **GetPageViewPortID** function

Return values

0	The specified ViewPortID was invalid or the viewport does not have a measurement dictionary
Non-zero	A MeasureDictID value

GetViewPortName

Version history

This function was introduced in Quick PDF Library version 8.13.

Description

Returns the name of the specified viewport.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetViewPortName(
  ViewPortID: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetViewPortName(
  ViewPortID As Long) As String
```

DLL

```
wchar_t * DPLGetViewPortName(int InstanceID, int ViewPortID);
```

Parameters

ViewPortID	A value returned by the **GetPageViewPortID** function

GetViewPortPtDataDict

Version history

This function was introduced in Quick PDF Library version 8.13.

Description

Returns the PtData dictionary of the specified viewport.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetViewPortPtDataDict(
  ViewPortID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetViewPortPtDataDict(
  ViewPortID As Long) As Long
```

DLL

```
int DPLGetViewPortPtDataDict(int InstanceID, int ViewPortID);
```

Parameters

ViewPortID	A value returned by the **GetPageViewPortID** function

Return values

0	The ViewPortID parameter was incorrect or the viewport does not have a PtData
Non-zero	A PtDataID value

GetViewerPreferences

Description

Returns the viewer preferences for the document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetViewerPreferences(
  Option: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetViewerPreferences(
  Option As Long) As Long
```

DLL

```
int DPLGetViewerPreferences(int InstanceID, int Option);
```

Parameters

Option
1 = Hide toolbar
2 = Hide menubar
3 = Hide window user interface
4 = Resize window to first page size
5 = Center window
6 = Display document title
7 = Page mode after full screen
8 = Predominant text reading order
9 = Display boundary for viewing
10 = Clipping boundary for viewing
11 = Display boundary for printing
12 = Clipping boundary for printing
13 = Default print dialog: scaling
14 = Default print dialog: duplex
15 = Default print dialog: auto paper tray
16 = Default print dialog: number of copies

Return values

See the **SetViewerPreferences** function to determine possible return values for each Option value.

GetWrappedText

Description

Get the positions where text will wrap, based on the current font and text size. The
SetBreakString function can be used to set the delimiter for the linebreak. The default is a CR/LF
pair. On some systems a LineFeed may be default.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetWrappedText(Width: Double; Delimiter,
  Text: WideString): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetWrappedText(Width As Double,
  Delimiter As String, Text As String) As String
```

DLL

```
wchar_t * DPLGetWrappedText(int InstanceID, double Width,
  wchar_t * Delimiter, wchar_t * Text);
```

Parameters

Width	The width of the block to wrap the text to
Delimiter	The string to place between each line
Text	The text to wrap

Return values

Returns the lines of the text block, separated by the Delimiter string

GetWrappedTextBreakString

Description

Similar to the **GetWrappedText** function, but preserves the break strings originally in the text. This is useful for splitting text into different areas on the page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetWrappedTextBreakString(Width: Double;
  Delimiter, Text: WideString): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetWrappedTextBreakString(
  Width As Double, Delimiter As String, Text As String) As String
```

DLL

```
wchar_t * DPLGetWrappedTextBreakString(int InstanceID, double Width,
  wchar_t * Delimiter, wchar_t * Text);
```

Parameters

Width	The width that the text should be wrapped to
Delimiter	The delimiter to use between wrapped lines
Text	The text to wrap

GetWrappedTextHeight

Text, Page layout

Description

Get the height of a block of text wrapped to a certain width, based on the current font and text size. The **SetBreakString** function can be used to set the delimiter for the linebreak. The default is a CR/LF pair. On some systems a LineFeed may be default.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetWrappedTextHeight(Width: Double;
  Text: WideString): Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetWrappedTextHeight(
  Width As Double, Text As String) As Double
```

DLL

```
double DPLGetWrappedTextHeight(int InstanceID, double Width,
  wchar_t * Text);
```

Parameters

Width	The width of the block to wrap the text to
Text	The text to wrap

Return values

Returns the height of the text block

GetWrappedTextLineCount

Description

Determine the number of lines a block of text wrapped to a certain width will take up, based on the current font and text size. The **SetBreakString** function can be used to set the delimiter for the linebreak. The default is CR / LF pair. On some systems a LineFeed may be default.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetWrappedTextLineCount(Width: Double;
  Text: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetWrappedTextLineCount(
  Width As Double, Text As String) As Long
```

DLL

```
int DPLGetWrappedTextLineCount(int InstanceID, double Width,
  wchar_t * Text);
```

Parameters

Width	The width of the block to wrap the text to
Text	The text to wrap

Return values

	The number of lines

GetXFAFormFieldCount

Version history

This function was introduced in Quick PDF Library version 7.14.

Description

Returns the number of XFA form fields in the selected document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetXFAFormFieldCount: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetXFAFormFieldCount As Long
```

DLL

```
int DPLGetXFAFormFieldCount(int InstanceID);
```

GetXFAFormFieldName

Version history

This function was introduced in Quick PDF Library version 7.14.

Description

Returns the name of the specified form field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetXFAFormFieldName(
  Index: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetXFAFormFieldName(
  Index As Long) As String
```

DLL

```
wchar_t * DPLGetXFAFormFieldName(int InstanceID, int Index);
```

Parameters

Index	The index of the XFA form field. The first XFA form field has an index of 1 and the last XFA form field has a value as returned by the **GetXFAFormFieldCount** function.

GetXFAFormFieldNames

Version history

This function was introduced in Quick PDF Library version 7.14.

Description

Returns a list of the names of the XFA form fields separated by the specified delimiter.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetXFAFormFieldNames(
  Delimiter: WideString): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetXFAFormFieldNames(
  Delimiter As String) As String
```

DLL

```
wchar_t * DPLGetXFAFormFieldNames(int InstanceID, wchar_t * Delimiter);
```

Parameters

Delimiter	The delimiter to use to separate the XFA form field names.

GetXFAFormFieldValue

Description

Returns the value of the specified XFA form field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetXFAFormFieldValue(
  XFAFieldName: WideString): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GetXFAFormFieldValue(
  XFAFieldName As String) As String
```

DLL

```
wchar_t * DPLGetXFAFormFieldValue(int InstanceID, wchar_t * XFAFieldName);
```

Parameters

XFAFieldName	The name of the XFA field to work with.

GetXFAToString

Version history

This function was introduced in Quick PDF Library version 8.16.

Description

Returns the complete XFA form contents as an XML string.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GetXFAToString(
  Options: Integer): AnsiString;
```

DLL

```
char * DPLGetXFAToString(int InstanceID, int Options);
```

Parameters

Options	Reserved for future use. Should be set to zero.

GlobalJavaScriptCount

Description

Returns the number of global JavaScript packages in the selected document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GlobalJavaScriptCount: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GlobalJavaScriptCount As Long
```

DLL

```
int DPLGlobalJavaScriptCount(int InstanceID);
```

GlobalJavaScriptPackageName

Description

Returns the name of the JavaScript package with the specified index. This package name can be used with the **GetGlobalJavaScript** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.GlobalJavaScriptPackageName(
   Index: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::GlobalJavaScriptPackageName(
   Index As Long) As String
```

DLL

```
wchar_t * DPLGlobalJavaScriptPackageName(int InstanceID, int Index);
```

Parameters

Index	The index of the global JavaScript package. The first package has an index of 1. The last package has an index equal to the value returned by the **GlobalJavaScriptCount** function.

HasFontResources

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Determines if the selected document has font resources. If the document does not have font resources it can be assumed to be an image only PDF.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.HasFontResources: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::HasFontResources As Long
```

DLL

```
int DPLHasFontResources(int InstanceID);
```

Return values

0	The selected document does not have font resources
Non-zero	The selected document has font resources

HasPageBox

Description

Indicates whether the selected page has the specified boundary rectangle.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.HasPageBox(BoxType: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::HasPageBox(
  BoxType As Long) As Long
```

DLL

```
int DPLHasPageBox(int InstanceID, int BoxType);
```

Parameters

BoxType	1 = MediaBox
	2 = CropBox
	3 = BleedBox
	4 = TrimBox
	5 = ArtBox

Return values

0	The page does not have the specified boundary rectangle
1	The page has the specified boundary rectangle
2	The page does not have the specified boundary rectangle, but there is a value in a parent page tree node that is being inherited by the page

HidePage

Description

Hides the selected page. This is similar to deleting the page, but the page contents are not removed from the PDF document. This is sometimes useful when used in conjunction with the **ClonePages** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.HidePage: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::HidePage As Long
```

DLL

```
int DPLHidePage(int InstanceID);
```

ImageCount

Image handling, Document properties

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Returns the total number of images added to the PDF file. This function does not take into account the images that may have already been in an existing PDF document which was loaded with the **LoadFromFile** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ImageCount: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ImageCount As Long
```

DLL

```
int DPLImageCount(int InstanceID);
```

ImageFillColor

Description

Returns the color of the center pixel in the selected image. This could be used to identify a placeholder image for later replacement.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ImageFillColor: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ImageFillColor As Long
```

DLL

```
int DPLImageFillColor(int InstanceID);
```

ImageHeight

Image handling

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

The height of the selected image.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ImageHeight: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ImageHeight As Long
```

DLL

```
int DPLImageHeight(int InstanceID);
```

Return values

0	No image has been selected
Non-zero	The height in pixels of the selected image

ImageHorizontalResolution

Description

Returns the horizontal resolution of the selected image, if it is available. Presently only the resolution of JFIF/JPEG, Exif/JPEG, TIFF and BMP images can be retrieved. Use the **ImageResolutionUnits** function to determine if this measurement is in dots-per-inch (DPI) or dots-per-centimetre (DPCM).

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ImageHorizontalResolution: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ImageHorizontalResolution As Long
```

DLL

```
int DPLImageHorizontalResolution(int InstanceID);
```

ImageResolutionUnits

Image handling

Description

Use this function to determine the units of the **ImageHorizontalResolution** and **ImageVerticalResolution** results.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ImageResolutionUnits: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ImageResolutionUnits As Long
```

DLL

```
int DPLImageResolutionUnits(int InstanceID);
```

Return values

0	Unknown
1	No units, values specify the aspect ratio
2	Dots per inch (DPI)
3	Dots per centimetre (DPCM)

ImageType

Description

The type of the selected image.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ImageType: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ImageType As Long
```

DLL

```
int DPLImageType(int InstanceID);
```

Return values

0	No image is selected
1	The selected image is a JPEG image
2	The selected image is a BMP image
3	The selected image is a TIFF image

ImageVerticalResolution

Image handling

Description

Returns the vertical resolution of the selected image, if it is available. Presently only the resolution of JFIF/JPEG, Exif/JPEG, TIFF and BMP images can be retrieved. Use the **ImageResolutionUnits** function to determine if this measurement is in dots-per-inch (DPI) or dots-per-centimetre (DPCM).

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ImageVerticalResolution: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ImageVerticalResolution As Long
```

DLL

```
int DPLImageVerticalResolution(int InstanceID);
```

ImageWidth

Image handling

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

The width of the selected image.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ImageWidth: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ImageWidth As Long
```

DLL

```
int DPLImageWidth(int InstanceID);
```

Return values

0	No image has been selected
Non-zero	The width in pixels of the selected image

ImportEMFFromFile

Vector graphics, Image handling

Version history

This function was introduced in Quick PDF Library version 7.15.

Description

Adds a WMF or EMF image from a file to the selected document.

Once an image has been added to the document it can be drawn on any page multiple times without further increasing the size of the PDF file.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ImportEMFFromFile(FileName: WideString;
  FontOptions, GeneralOptions: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ImportEMFFromFile(
  FileName As String, FontOptions As Long,
  GeneralOptions As Long) As Long
```

DLL

```
int DPLImportEMFFromFile(int InstanceID, wchar_t * FileName,
  int FontOptions, int GeneralOptions);
```

Parameters

FileName	The file name of the image to add.
FontOptions	If GeneralOptions is 1 this parameter is ignored, otherwise the following values take effect: 0 = Use the first font added to the PDF 1 = Automatically add fonts as non-embedded TrueType fonts
GeneralOptions	0 = Import as a vector image 1 = Import as a bitmap image

Return values

0	The image could not be added
Non-zero	The image was added successfully. The ImageID is returned which can be passed to functions like **SelectImage** and **DrawImage**.

ImportEMFFromStream

Vector graphics, Image handling

Version history

This function was introduced in Quick PDF Library version 7.21.

Description

Adds a WMF or EMF image from a TStream to the selected document.

Once an image has been added to the document it can be drawn on any page multiple times without further increasing the size of the PDF file.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ImportEMFFromStream(InStream: TStream;
  FontOptions, GeneralOptions: Integer): Integer;
```

Parameters

InStream	The TStream object containing the EMF/WMF data
FontOptions	If GeneralOptions is 1 this parameter is ignored, otherwise the following values take effect: 0 = Use the first font added to the PDF 1 = Automatically add fonts as non-embedded TrueType fonts
GeneralOptions	0 = Import as a vector image 1 = Import as a bitmap image

InsertPages

Description

Inserts one or more blank pages into the document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.InsertPages(StartPage,
  PageCount: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::InsertPages(StartPage As Long,
  PageCount As Long) As Long
```

DLL

```
int DPLInsertPages(int InstanceID, int StartPage, int PageCount);
```

Parameters

StartPage	The page number of the first page to insert
PageCount	The total number of pages to insert

Return values

0	Failed
Non-zero	The new total number of pages in the document

InsertTableColumns

Version history

This function was introduced in Quick PDF Library version 7.16.

Description

Adds columns to the specified table at any position

Syntax

Delphi

```
function TDebenuPDFLibrary1115.InsertTableColumns(TableID, Position,
  NewColumnCount: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::InsertTableColumns(
  TableID As Long, Position As Long,
  NewColumnCount As Long) As Long
```

DLL

```
int DPLInsertTableColumns(int InstanceID, int TableID, int Position,
  int NewColumnCount);
```

Parameters

TableID	A TableID returned by the **CreateTable** function
Position	The position to insert the new culumns. Minimum value is 1. Maximum value is one greater than the value returned by the **GetTableColumnCount** function.
NewColumnCount	The number of columns to add to the table

Return values

0	Columns could not be added. Check the TableID parameter and make sure NewColumnCount is greater than or equal to 1. The Position parameter must also be within range.
Non-zero	The total number of columns in the table after adding the new columns.

InsertTableRows

Version history

This function was introduced in Quick PDF Library version 7.16.

Description

Adds rows to the specified table at any position

Syntax

Delphi

```
function TDebenuPDFLibrary1115.InsertTableRows(TableID, Position,
  NewRowCount: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::InsertTableRows(TableID As Long,
  Position As Long, NewRowCount As Long) As Long
```

DLL

```
int DPLInsertTableRows(int InstanceID, int TableID, int Position,
  int NewRowCount);
```

Parameters

TableID	A TableID returned by the **CreateTable** function
Position	The position to insert the new rows. Minimum value is 1. Maximum value is one greater than the value returned by the **GetTableRowCount** function.
NewRowCount	The number of rows to add to the table

Return values

0	Rows could not be added. Check the TableID parameter and make sure NewRowCount is greater than or equal to 1. The Position parameter must also be within range.
Non-zero	The total number of rows in the table after adding the new rows.

IsAnnotFormField

Version history

This function was introduced in Quick PDF Library version 7.18.

Description

For an annotation to be a form field it must be attached to the document form.

This function checks if the specified annotation is allowed to be attached to the document form and whether it is currently attached.

For an annotation to be attached to the document form it must be a Widget annotation and it cannot be a child of another annotation.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.IsAnnotFormField(Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::IsAnnotFormField(
  Index As Long) As Long
```

DLL

```
int DPLIsAnnotFormField(int InstanceID, int Index);
```

Parameters

Index	The index of the annotation. The first annotation on the page has an index of 1.

Return values

0	The specified annotation is not a Widget annotation or it is the child of another annotation.
1	The specified annotation is a form field and is currently attached to the document form.
2	The specified annotation is in the correct format to be a form field but it is not currently attached to the document form. Use the **AttachAnnotToForm** function to attach it.

IsLinearized

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

This function was called Linearized in v10 and earlier. Reports whether the selected document was loaded from a linearized file. This is for informational purposes only. If the file is resaved it will no longer be linearized.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.IsLinearized: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::IsLinearized As Long
```

DLL

```
int DPLIsLinearized(int InstanceID);
```

Return values

0	The original file was not linearized
1	The original file was linearized

IsTaggedPDF

Version history

This function was introduced in Quick PDF Library version 9.14.

Description

Determines if the selected document has the MarkInfo/Marked property set.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.IsTaggedPDF: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::IsTaggedPDF As Long
```

DLL

```
int DPLIsTaggedPDF(int InstanceID);
```

Return values

0	The document is not tagged
1	The document is tagged

LastErrorCode

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Use this function to determine the reason certain functions failed.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.LastErrorCode: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::LastErrorCode As Long
```

DLL

```
int DPLLastErrorCode(int InstanceID);
```

Return values

101	The Strength parameter passed to the **Encrypt** function was invalid
102	The Permissions parameter passed to the **Encrypt** function was invalid. Use the **EncodePermissions** function to construct a value for this parameter
103	The **Encrypt** function was used on a document that was already encrypted
104	The **Encrypt** function failed for an unknown reason
201	The **SetInformation** function failed because the document is encrypted
202	The Key parameter passed to the **SetInformation** function was out of range
301	An invalid combination of barcode and option was sent to the **DrawBarcode** function
302	Non-numeric characters were sent to **DrawBarcode** using EAN-13
303	The EAN-13 barcode has an invalid checksum character
401	Could not open input file
402	Output file already exists and could not be deleted
403	Could not open output file
404	Invalid password
405	Document is not encrypted
406	Document is already encrypted
407	Invalid encryption strength
408	Invalid permissions
409	Invalid file structure, file is damaged
410	One of the input files is encrypted
411	File not found
412	Invalid page range list
501	The specified FileHandle was invalid
999	The function could not be used because the library is not unlocked

LastRenderError

Version history

This function was introduced in Quick PDF Library version 7.13.

Description

Returns the exception information in cases where the renderer encountered an error.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.LastRenderError: WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::LastRenderError As String
```

DLL

```
wchar_t * DPLLastRenderError(int InstanceID);
```

LibraryVersion

Description

Returns the version of the library, for example "11.14".

Syntax

Delphi

```
function TDebenuPDFLibrary1115.LibraryVersion: WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::LibraryVersion As String
```

DLL

```
wchar_t * DPLLibraryVersion(int InstanceID);
```

LibraryVersionEx

Version history

This function was introduced in Quick PDF Library version 11.14.

Description

Returns the full version of the library, including subversion. For example "11.14.1.0".

Syntax

Delphi

```
function TDebenuPDFLibrary1115.LibraryVersionEx: WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::LibraryVersionEx As String
```

DLL

```
wchar_t * DPLLibraryVersionEx(int InstanceID);
```

LicenseInfo

Miscellaneous functions

Description

Returns information about the unlock license used.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.LicenseInfo: WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::LicenseInfo As String
```

DLL

```
wchar_t * DPLLicenseInfo(int InstanceID);
```

LinearizeFile

Document manipulation, Miscellaneous functions

Version history

This function was introduced in Quick PDF Library version 11.11.

Description

Linearizes the specified PDF file for fast web view.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.LinearizeFile(InputFileName, Password,
  OutputFileName: WideString; Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::LinearizeFile(
  InputFileName As String, Password As String,
  OutputFileName As String, Options As Long) As Long
```

DLL

```
int DPLLinearizeFile(int InstanceID, wchar_t * InputFileName,
  wchar_t * Password, wchar_t * OutputFileName, int Options);
```

Parameters

InputFileName	The path and file name of the input PDF to transform.
Password	The optional password to open the input PDF if it is encrypted
OutputFileName	The path and file name of the signed PDF that should be created. This should be different to InputFileName.
Options	Reserved for future use, should be set to zero.

Return values

1	Success
2	Input PDF not found
3	Input PDF cannot be read
4	Input PDF password incorrect
5	Could not write output file

LoadFromCanvasDC

Vector graphics, Document management

Version history

This function was introduced in Quick PDF Library version 7.14.

Description

Creates a new document from the drawing operations applied to the DC returned by the **GetCanvasDC** function.

When the Options parameter is set to 3, use the **NoEmbedFontListAdd** function to add fonts to the no embed font list.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.LoadFromCanvasDC(DPI: Double;
  Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::LoadFromCanvasDC(DPI As Double,
  Options As Long) As Long
```

DLL

```
int DPLLoadFromCanvasDC(int InstanceID, double DPI, int Options);
```

Parameters

DPI	The DPI to use for the new document. For example, if the canvas was created with a width and height of 96 and the DPI is specified as 192, the resulting document will be 0.5 inches in width and height.
Options	-1 = Convert the drawing commands to a single image using GDI+ 0 = Process the drawing commands as vector graphics, fonts are not embedded 1 = Process the drawing commands as vector graphics, fonts are embedded but not compressed 2 = Process the drawing commands as vector graphics, fonts are embedded and compressed 3 = Process the drawing commands as vector graphics, fonts not in the no embed font list are embedded and compressed

Return values

0	A canvas has not been created
1	The canvas DC was processed correctly and a new document has been created

LoadFromFile

Document management

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Loads a PDF document from a file on disk. If the function succeeds, the loaded document will be selected and its DocumentID can be retrieved using the **SelectedDocument** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.LoadFromFile(FileName,
  Password: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::LoadFromFile(FileName As String,
  Password As String) As Long
```

DLL

```
int DPLLoadFromFile(int InstanceID, wchar_t * FileName,
  wchar_t * Password);
```

Parameters

FileName	The path and file name of the file to load.
Password	The password to open the file

Return values

0	The file could not be read or processed. Use the **LastErrorCode** function to determine the cause of the failure.
1	The file was loaded successfully

LoadFromStream

Description

This function, only available in the Delphi versions of the library, allows a PDF document to be loaded from a TStream object. If the function succeeds, the loaded document will be selected and its DocumentID can be retrieved using the **SelectedDocument** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.LoadFromStream(InStream: TStream;
  Password: WideString): Integer;
```

Parameters

InStream	The TStream object containing the PDF document data
Password	The password to load the file

Return values

0	A PDF document could not be read from the stream. Use the **LastErrorCode** function to determine the reason this function failed.
1	A PDF document was successfully read from the stream. Use the **SelectedDocument** function to obtain the Document ID which can be used later to select this specific document.

LoadFromString

Description

Similar to the **LoadFromFile** function, except the data for the PDF document is passed in as a string. If the function succeeds, the loaded document will be selected and its DocumentID can be retrieved using the **SelectedDocument** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.LoadFromString(const Source: AnsiString;
  Password: WideString): Integer;
```

DLL

```
int DPLLoadFromString(int InstanceID, char * Source, wchar_t * Password);
```

Parameters

Source	The source data to load the PDF document from
Password	The password to load the file

Return values

0	The PDF could not be loaded
1	The PDF was loaded from the string successfully

LoadFromVariant

Description

Loads a PDF document from a byte array stored as a Variant type. This function is only available in the ActiveX editions of the library. If the function succeeds, the loaded document will be selected and its DocumentID can be retrieved using the **SelectedDocument** function.

Syntax

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::LoadFromVariant(
    Source As Variant, Password As String) As Long
```

Parameters

Source	The byte array to load the PDF document from
Password	The password to load the file

Return values

0	The document could not be loaded. Check the result of the **LastErrorCode** function for more information.
1	The document was loaded successfully

LoadState

Description

Loads the graphics state previously stored with **SaveState**.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.LoadState: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::LoadState As Long
```

DLL

```
int DPLLoadState(int InstanceID);
```

MergeDocument

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Use this function to join another document to the selected document. After merging, the second document is deleted.

Form fields and annotations from the second document are preserved but outlines (bookmarks) are not.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.MergeDocument(DocumentID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::MergeDocument(
  DocumentID As Long) As Long
```

DLL

```
int DPLMergeDocument(int InstanceID, int DocumentID);
```

Parameters

DocumentID	The ID of the document to join to the selected document

Return values

0	The documents could not be merged together
1	The merging was successful

MergeFileList

Description

Merges all the files in a named file list and saves the resulting merged document to the specified file. Use the **ClearFileList**, **FileListCount** and **AddToFileList** functions to construct the named file list. There must be two or more files in the file list in order for the merging to succeed.

Outlines (bookmarks), form fields and annotations from all the documents will be present in the merged document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.MergeFileList(ListName,
  OutputFileName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::MergeFileList(
  ListName As String, OutputFileName As String) As Long
```

DLL

```
int DPLMergeFileList(int InstanceID, wchar_t * ListName,
  wchar_t * OutputFileName);
```

Parameters

ListName	The name of the list of files to merge together
OutputFileName	The path and file name of the file to create which will contain the merged files.

Return values

	The number of documents which were successfully merged together. If this is less than the intended number use the **FileListItem** function to find the file which caused the merge process to end prematurely.

MergeFileListFast

Description

Similar to the **MergeFileList** function, but uses an advanced algorithm to improve speed.

A new file list will be created during merging that will contain the result of the merge process for each of the items in the specified file list. The new file list will have the same name as the original file list with the word Result appended. For example, if the original file list was called "MyFiles", then the new file list will be called "MyFilesResult". This new file list will not contain file names, but will contain a text description of the status of the matching file during the merge process.

There must be two or more files in the file list in order for the merging to succeed.

Form fields and annotations from all the documents will be present in the merged document but only outlines (bookmarks) from the first document will be in the merged document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.MergeFileListFast(ListName,
  OutputFileName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::MergeFileListFast(
  ListName As String, OutputFileName As String) As Long
```

DLL

```
int DPLMergeFileListFast(int InstanceID, wchar_t * ListName,
  wchar_t * OutputFileName);
```

Parameters

ListName	The name of the file list to use. All the files in this list will be merged together.
OutputFileName	The path and file name of the output file to create. This file will contain all the files from the file list.

Return values

0	The merge process could not be completed. Use the GetLastError function to determine the cause of the error.
Non-zero	The number of files that were successfully merged

MergeFiles

Description

Merges two files on disk and saves the merged document to a new file. The files are accessed directly on disk, the entire file does not have to be loaded into memory so this function can be used with huge documents. The files must not be encrypted. Monitor the size of the output file while this function runs to work out the progress.

Outlines (bookmarks), form fields and annotations from the both documents will be present in the merged document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.MergeFiles(FirstFileName, SecondFileName,
  OutputFileName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::MergeFiles(
  FirstFileName As String, SecondFileName As String,
  OutputFileName As String) As Long
```

DLL

```
int DPLMergeFiles(int InstanceID, wchar_t * FirstFileName,
  wchar_t * SecondFileName, wchar_t * OutputFileName);
```

Parameters

FirstFileName	The name of the first file to merge.
SecondFileName	The name of the second file to merge.
OutputFileName	The name of the file to create which will contain the merged document.

Return values

0	The files could not be merged. Use the **LastErrorCode** function to determine the cause of the failure.
1	The files were merged successfully and the new merged document was created

MergeStreams

Description

This function is similar to the **MergeFiles** function, however instead of working with files on disk, it merges two PDF documents stored in different TStream objects and saves the merged document into a third stream.

Outlines (bookmarks), form fields and annotations from the both documents will be present in the merged document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.MergeStreams(FirstStream, SecondStream,
  OutputStream: TStream): Integer;
```

Parameters

FirstStream	The stream containing the first document
SecondStream	The stream containing the second document
OutputStream	The merged document is written into this stream

Return values

0	The documents could not be merged. Use the **LastErrorCode** function to determine the cause of the failure.
1	The merge process was successful

MergeTableCells

Version history

This function was introduced in Quick PDF Library version 7.14.

Description

Merges multiple cells from the specified table into one cell.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.MergeTableCells(TableID, FirstRow,
  FirstColumn, LastRow, LastColumn: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::MergeTableCells(TableID As Long,
  FirstRow As Long, FirstColumn As Long, LastRow As Long,
  LastColumn As Long) As Long
```

DLL

```
int DPLMergeTableCells(int InstanceID, int TableID, int FirstRow,
  int FirstColumn, int LastRow, int LastColumn);
```

Parameters

TableID	A TableID returned by the **CreateTable** function
FirstRow	The the number of the first row to set. Top row is row number 1.
FirstColumn	The the number of the first column to set. Left most column is column number 1.
LastRow	The number of the final row to set
LastColumn	The number of the final column to set

MoveContentStream

Version history

This function was renamed in Quick PDF Library version 8.11.
The function name in earlier versions was MoveLayer.

Description

A page in a PDF document has one or more content stream parts that together contain all the PDF page description commands for the page.

This function can be used to change the order in which the content stream parts are drawn onto the page to bring certain information to the front or push it to the back.

Content stream parts that you want placed at the back should be drawn first (index of 1).

Syntax

Delphi

```
function TDebenuPDFLibrary1115.MoveContentStream(FromPosition,
  ToPosition: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::MoveContentStream(
  FromPosition As Long, ToPosition As Long) As Long
```

DLL

```
int DPLMoveContentStream(int InstanceID, int FromPosition, int ToPosition);
```

Parameters

FromPosition	The current content stream part index. The first content stream part has an index of 1. The last content stream part has an index equal to the value returned by the **ContentStreamCount** function.
ToPosition	The new content stream part index.

Return values

0	The content stream part could not be moved
1	Success

MoveOutlineAfter

Description

Moves an outline item to appear directly after another outline item. The outline will be moved along with all children nodes.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.MoveOutlineAfter(OutlineID,
  SiblingID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::MoveOutlineAfter(
  OutlineID As Long, SiblingID As Long) As Long
```

DLL

```
int DPLMoveOutlineAfter(int InstanceID, int OutlineID, int SiblingID);
```

Parameters

OutlineID	The ID of the outline as returned by the **NewOutline** function. Alternatively, use the **GetOutlineID** function to get a valid outline ID.
SiblingID	The outline will be moved to a position after the outline with this ID

Return values

0	The outline was not moved, the OutlineID or SiblingID parameters were invalid or were the same value
1	The outline was moved successfully

MoveOutlineBefore

Description

Moves an outline item to appear directly before another outline item. The outline will be moved along with all children nodes.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.MoveOutlineBefore(OutlineID,
  SiblingID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::MoveOutlineBefore(
  OutlineID As Long, SiblingID As Long) As Long
```

DLL

```
int DPLMoveOutlineBefore(int InstanceID, int OutlineID, int SiblingID);
```

Parameters

OutlineID	The ID of the outline as returned by the **NewOutline** function. Alternatively, use the **GetOutlineID** function to get a valid outline ID.
SiblingID	The outline will be moved to a position before the outline with this ID

Return values

0	The outline was not moved, the OutlineID or SiblingID parameters were invalid or were the same value
1	The outline was moved successfully

MovePage

Description

Moves the selected page to a new position in the document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.MovePage(NewPosition: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::MovePage(
   NewPosition As Long) As Long
```

DLL

```
int DPLMovePage(int InstanceID, int NewPosition);
```

Parameters

NewPosition	The new position of the page

Return values

0	The page could not be moved. Check the value of the NewPosition parameter.
1	The page was moved successfully

MovePath

Description

Starts a new sub-path within the current path. This allows complex shapes to be created (for example, with pieces cut out).

Syntax

Delphi

```
function TDebenuPDFLibrary1115.MovePath(NewX, NewY: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::MovePath(NewX As Double,
  NewY As Double) As Long
```

DLL

```
int DPLMovePath(int InstanceID, double NewX, double NewY);
```

Parameters

NewX	The new horizontal co-ordinate of the starting point of the new sub-path
NewY	The new vertical co-ordinate of the starting point of the new sub-path

MultiplyScale

Description

Multiplies the drawing scale by a specified factor. For example, multiplying the scale by 0.5 will draw graphics at half their size with the same drawing commands.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.MultiplyScale(MultScaleBy: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::MultiplyScale(
  MultScaleBy As Double) As Long
```

DLL

```
int DPLMultiplyScale(int InstanceID, double MultScaleBy);
```

Parameters

MultScaleBy	The factor to multiply the current drawing scale by

NewChildFormField

Version history

This function was introduced in Quick PDF Library version 7.18.

Description

Adds a new form field to the selected page as a child of another field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.NewChildFormField(Index: Integer;
  Title: WideString; FieldType: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::NewChildFormField(Index As Long,
  Title As String, FieldType As Long) As Long
```

DLL

```
int DPLNewChildFormField(int InstanceID, int Index, wchar_t * Title,
  int FieldType);
```

Parameters

Index	The index of the parent field.
Title	The title of the new form field. The title cannot contain the period "." character.
FieldType	The type of the field to create: 1 = Text 2 = Pushbutton 3 = Checkbox 4 = Radiobutton 5 = Choice 6 = Signature 7 = Parent

Return values

0	The new form field could not be created
Non-zero	The form field was created successfully, and this is the index of the new field

NewContentStream

Version history

> This function was renamed in Quick PDF Library version 8.11.
> The function name in earlier versions was NewLayer.

Description

> A page in a PDF document has one or more content stream parts that together contain all the PDF page description commands for the page.
>
> This function creates a new content stream part on the selected page. If required, the new content stream part can then be moved behind the existing information on the page using the **MoveContentStream** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.NewContentStream: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::NewContentStream As Long
```

DLL

```
int DPLNewContentStream(int InstanceID);
```

Return values

0	The new content stream part could not be created
Non-zero	The index of the new content stream part. The first part has an index of 1.

NewCustomPrinter

Rendering and printing

Description

Creates a custom printer and returns the name of the custom printer. The returned printer name can be used as the PrinterName parameter of the **PrintDocument** function. Before printing, the properties of the printer can be set using the **SetupCustomPrinter** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.NewCustomPrinter(
   OriginalPrinterName: WideString): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::NewCustomPrinter(
   OriginalPrinterName As String) As String
```

DLL

```
wchar_t * DPLNewCustomPrinter(int InstanceID,
   wchar_t * OriginalPrinterName);
```

Parameters

OriginalPrinterName	The name of the printer to use for printing. This is the name that appears in the Windows Print Manager. Use the **GetPrinterNames** function to return a list of valid printers on the system.

NewDestination

Version history

This function was introduced in Quick PDF Library version 7.22.

Description

Creates a new destination object that can be used with the **AddLinkToDestination**, **GetDestPage**, **GetDestType** or **GetDestValue** functions.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.NewDestination(DestPage, Zoom,
  DestType: Integer; Left, Top, Right, Bottom: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::NewDestination(DestPage As Long,
  Zoom As Long, DestType As Long, Left As Double, Top As Double,
  Right As Double, Bottom As Double) As Long
```

DLL

```
int DPLNewDestination(int InstanceID, int DestPage, int Zoom,
  int DestType, double Left, double Top, double Right,
  double Bottom);
```

Parameters

DestPage	The page number that this destination object links to
Zoom	The zoom percentage to use for the destination object, valid values from 0 to 6400. Only used for DestType = 1, should be set to 0 for other DestTypes.
DestType	1 = "XYZ" - the target page is positioned at the point specified by the Left and Top parameters. The Zoom parameter specifies the zoom percentage. 2 = "Fit" - the entire page is zoomed to fit the window. None of the other parameters are used and should be set to zero. 3 = "FitH" - the page is zoomed so that the entire width of the page is visible. The height of the page may be greater or less than the height of the window. The page is positioned at the vertical position specified by the Top parameter. 4 = "FitV" - the page is zoomed so that the entire height of the page can be seen. The width of the page may be greater or less than the width of the window. The page is positioned at the horizontal position specified by the Left parameter. 5 = "FitR" - the page is zoomed so that a certain rectangle on the page is visible. The Left, Top, Right and Bottom parameters define the rectangular area on the page. 6 = "FitB" - the page is zoomed so that it's bounding box is visible. 7 = "FitBH" - the page is positioned vertically at the position specified by the Top parameter. The page is zoomed so that the entire width of the page's bounding box is visible. 8 = "FitBV" - the page is positioned at the horizontal position specified by the Left parameter. The page is zoomed just enough to fit the entire height of the bounding box into the window.
Left	The horizontal position used by DestType = 1, 4, 5 and 8
Top	The vertical position used by DestType = 1, 3, 5 and 7
Right	The horizontal position of the righthand edge of the rectangle. Used by DestType = 5
Bottom	The horizontal position of the bottom of the rectangle. Used by DestType = 5

Return values

0	The DestPage parameter was invalid
Non-zero	A DestID value

NewDocument

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Creates a new blank document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.NewDocument: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::NewDocument As Long
```

DLL

```
int DPLNewDocument(int InstanceID);
```

Return values

0	There was an error while trying to create the new document. This should never occur.
Non-zero	The ID of the new document

NewFormField

Description

Adds a new form field to the selected page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.NewFormField(Title: WideString;
  FieldType: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::NewFormField(Title As String,
  FieldType As Long) As Long
```

DLL

```
int DPLNewFormField(int InstanceID, wchar_t * Title, int FieldType);
```

Parameters

Title	The title of the new form field. The title cannot contain the period "." character.
FieldType	The type of the field to create: 1 = Text 2 = Pushbutton 3 = Checkbox 4 = Radiobutton 5 = Choice 6 = Signature 7 = Parent

Return values

0	The new form field could not be created
Non-zero	The form field was created successfully, and this is the index of the new field

NewInternalPrinterObject

Version history

This function was introduced in Quick PDF Library version 9.16.

Description

Creates a new internal printer object for use by subsequent printing operations.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.NewInternalPrinterObject(
  Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::NewInternalPrinterObject(
  Options As Long) As Long
```

DLL

```
int DPLNewInternalPrinterObject(int InstanceID, int Options);
```

Parameters

Options	Must be set to 0

Return values

0	The options parameter was not zero or the new internal printer object could not be created
1	Success

NewNamedDestination

Version history

This function was introduced in Quick PDF Library version 7.22.

Description

Creates a named destination.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.NewNamedDestination(DestName: WideString;
  DestID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::NewNamedDestination(
  DestName As String, DestID As Long) As Long
```

DLL

```
int DPLNewNamedDestination(int InstanceID, wchar_t * DestName, int DestID);
```

Parameters

DestName	The name of the destination
DestID	The destination to assign a name to

NewOptionalContentGroup

Description

Creates a new optional content group. The group name will appear in the Layers tab in Acrobat 6 or later.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.NewOptionalContentGroup(
  GroupName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::NewOptionalContentGroup(
  GroupName As String) As Long
```

DLL

```
int DPLNewOptionalContentGroup(int InstanceID, wchar_t * GroupName);
```

Parameters

GroupName	The name of the optional content group. This name is displayed in the PDF viewer user interface.

Return values

0	The new optional content group could not be created
Non-zero	An ID that can be used as the OptionalContentGroupID parameter with the other optional content group functions

NewOutline

Description

Adds a new outline item to the document. Outline items can be added in a hierarchical structure. In Acrobat Reader, outlines are referred to as bookmarks.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.NewOutline(Parent: Integer;
  Title: WideString; DestPage: Integer; DestPosition: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::NewOutline(Parent As Long,
  Title As String, DestPage As Long,
  DestPosition As Double) As Long
```

DLL

```
int DPLNewOutline(int InstanceID, int Parent, wchar_t * Title,
  int DestPage, double DestPosition);
```

Parameters

Parent	0 for a root item, or the ID of the parent item if this is a child item (returned by the **NewOutline** function). Alternatively, use the **GetOutlineID** function to get a valid outline ID.
Title	The title of the outline item.
DestPage	The destination page number that this outline item links to
DestPosition	The vertical position on the destination page to link to

Return values

0	The item could not be added
Non-zero	The ID of the item which was added successfully

NewPage

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Create a new page. The new page is added to the end of the document, and will have the same width and height as the selected page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.NewPage: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::NewPage As Long
```

DLL

```
int DPLNewPage(int InstanceID);
```

Return values

0	The page could not be added. This should never occur.
Non-zero	The page number of the page that was added

NewPageFromCanvasDC

Vector graphics, Page manipulation

Version history

This function was introduced in Quick PDF Library version 7.23.

Description

Adds a new page to the selected document from the drawing operations applied to the DC returned by the **GetCanvasDC** function.

When the Options parameter is set to 3 or 4, use the **NoEmbedFontListAdd** function to add fonts to the no embed font list.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.NewPageFromCanvasDC(DPI: Double;
  Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::NewPageFromCanvasDC(
  DPI As Double, Options As Long) As Long
```

DLL

```
int DPLNewPageFromCanvasDC(int InstanceID, double DPI, int Options);
```

Parameters

DPI	The DPI to use for the new document. For example, if the canvas was created with a width and height of 96 and the DPI is specified as 192, the resulting document will be 0.5 inches in width and height.
Options	-1 = Convert the drawing commands to a single image using GDI+ 0 = Process the drawing commands as vector graphics, fonts are not embedded 1 = Process the drawing commands as vector graphics, fonts are embedded but not compressed 2 = Process the drawing commands as vector graphics, fonts are embedded and compressed 3 = Process the drawing commands as vector graphics, fonts not in the no embed font list are embedded and compressed 4 = Same as 3 but fonts already added during previous calls to this function or the **LoadFromCanvasDC** function are reused

Return values

0	A canvas has not been created
1	The canvas DC was processed correctly and a new document has been created

NewPages

Description

This function is similar to the **NewPage** function, but allows you to add more than one new page to the selected document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.NewPages(PageCount: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::NewPages(
  PageCount As Long) As Long
```

DLL

```
int DPLNewPages(int InstanceID, int PageCount);
```

Parameters

PageCount	The number of pages to add to the document

Return values

0	The pages could not be added. This should never occur.
Non-zero	The total number of pages in the document after the new pages were added

NewPostScriptXObject

Description

Adds a PostScript XObject to the document. If the PostScript XObject is drawn onto the page with the **DrawPostScriptXObject** function the contents of the PostScript XObject will be placed into the generated PostScript for the page when printed to a PostScript printer.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.NewPostScriptXObject(
   PS: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::NewPostScriptXObject(
   PS As String) As Long
```

DLL

```
int DPLNewPostScriptXObject(int InstanceID, wchar_t * PS);
```

Parameters

PS	The PostScript that will be inserted

Return values

0	The PostScript XObject could not be added
Non-zero	A reference to the PostScript XObject which can be used with the **DrawPostScriptXObject** function

NewRGBAxialShader

Vector graphics, Color

Version history

This function was introduced in Quick PDF Library version 7.11.

Description

This function adds an axial shader to the current document. The color is varied linearly from one color to another between two points and is used for linear gradient fills.

The shader can be used with the **SetTextShader**, **SetLineShader** and **SetFillShader** functions to set the color of subsequently drawn vector graphics and text.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.NewRGBAxialShader(ShaderName: WideString;
  StartX, StartY, StartRed, StartGreen, StartBlue, EndX, EndY, EndRed,
  EndGreen, EndBlue: Double; Extend: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::NewRGBAxialShader(
  ShaderName As String, StartX As Double, StartY As Double,
  StartRed As Double, StartGreen As Double, StartBlue As Double,
  EndX As Double, EndY As Double, EndRed As Double,
  EndGreen As Double, EndBlue As Double, Extend As Long) As Long
```

DLL

```
int DPLNewRGBAxialShader(int InstanceID, wchar_t * ShaderName,
  double StartX, double StartY, double StartRed,
  double StartGreen, double StartBlue, double EndX, double EndY,
  double EndRed, double EndGreen, double EndBlue, int Extend);
```

Parameters

ShaderName	The name of the shader. Should be a simple string consisting of alphanumeric characters and no whitespace. This name is used with the **SetTextShader**, **SetLineShader** and **SetFillShader** functions.
StartX	The horizontal co-ordinate of the start point
StartY	The vertical co-ordinate of the start point
StartRed	The red component of the start color
StartGreen	The green component of the start color
StartBlue	The blue component of the start color
EndX	The horizontal co-ordinate of the end point
EndY	The vertical co-ordinate of the end point
EndRed	The red component of the end color
EndGreen	The green component of the end color
EndBlue	The blue component of the end color
Extend	0 = do not extend the beyond the start and end points 1 = extend the shader using solid color

Return values

0	The shader could not be added, possibly a shader with this name has already been added
1	The shader was added successfully

NewSignProcessFromFile

Version history

This function was introduced in Quick PDF Library version 9.13.

Description

Creates a new digital signature process using a file as the source document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.NewSignProcessFromFile(InputFile,
  Password: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::NewSignProcessFromFile(
  InputFile As String, Password As String) As Long
```

DLL

```
int DPLNewSignProcessFromFile(int InstanceID, wchar_t * InputFile,
  wchar_t * Password);
```

Parameters

InputFile	The path and name of the file to sign
Password	The password to open the PDF, if any

NewSignProcessFromStream

Version history

This function was introduced in Quick PDF Library version 9.13.

Description

Creates a new digital signature process using a stream as the source.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.NewSignProcessFromStream(
  InputStream: TStream; Password: WideString): Integer;
```

Parameters

InputStream	The stream object containing the PDF to be signed
Password	The password to open the PDF, if any

NewSignProcessFromString

Version history

This function was introduced in Quick PDF Library version 9.13.

Description

Creates a new digital signature process using a string of 8-bit bytes as the source.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.NewSignProcessFromString(
  const Source: AnsiString; Password: WideString): Integer;
```

DLL

```
int DPLNewSignProcessFromString(int InstanceID, char * Source,
  wchar_t * Password);
```

Parameters

Source	A string containing the document to be signed
Password	The password to open the PDF, if any

NewStaticOutline

Description

This function creates a new outline without an action. The action can later be set using the
SetOutlineDestination, **SetOutlineWebLink** or **SetOutlineJavaScript** functions.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.NewStaticOutline(Parent: Integer;
  Title: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::NewStaticOutline(Parent As Long,
  Title As String) As Long
```

DLL

```
int DPLNewStaticOutline(int InstanceID, int Parent, wchar_t * Title);
```

Parameters

Parent	0 for a root item, or the ID of the parent item if this is a child item
Title	The title of the outline item.

Return values

0	The outline item could not be added
Non-zero	The ID of the outline item that was added

NewTilingPatternFromCapturedPage

Vector graphics, Color

Version history

This function was introduced in Quick PDF Library version 8.16.

Description

This function converts a captured page into a tiling pattern and adds the pattern to the current document.

The pattern can be used with the **SetFillTilingPattern** function to set the color of subsequently drawn vector graphics.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.NewTilingPatternFromCapturedPage(
  PatternName: WideString; CaptureID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::NewTilingPatternFromCapturedPage(
  PatternName As String, CaptureID As Long) As Long
```

DLL

```
int DPLNewTilingPatternFromCapturedPage(int InstanceID,
  wchar_t * PatternName, int CaptureID);
```

Parameters

PatternName	The name of the tiling pattern. Should be a simple string consisting of alphanumeric characters and no whitespace. This name is used with the **SetFillTilingPattern** function.
CaptureID	The ID returned by the **CapturePage** or **CapturePageEx** functions.

Return values

0	The captured page could not be converted into a tiling pattern. The CaptureID parameter might be invalid or the PatternName has already been used.
1	Success

NoEmbedFontListAdd

Vector graphics, Fonts, Miscellaneous functions

Version history

This function was introduced in Quick PDF Library version 7.23.

Description

Adds a font name to the no embed font list used by the **LoadFromCanvasDC** and **NewPageFromCanvasDC** functions.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.NoEmbedFontListAdd(
  FontName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::NoEmbedFontListAdd(
  FontName As String) As Long
```

DLL

```
int DPLNoEmbedFontListAdd(int InstanceID, wchar_t * FontName);
```

Parameters

FontName	The font name to add to the list

Return values

0	The font name is already in the list
1	The font name was added to the list successfully

NoEmbedFontListCount

Vector graphics, Fonts, Miscellaneous functions

Version history

This function was introduced in Quick PDF Library version 7.23.

Description

Returns the number of font names in the no embed font list used by the **LoadFromCanvasDC** and **NewPageFromCanvasDC** functions.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.NoEmbedFontListCount: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::NoEmbedFontListCount As Long
```

DLL

```
int DPLNoEmbedFontListCount(int InstanceID);
```

NoEmbedFontListGet

Vector graphics, Fonts, Miscellaneous functions

Version history

This function was introduced in Quick PDF Library version 7.23.

Description

Returns the font name at the specified index in the no embed font list used by the **LoadFromCanvasDC** and **NewPageFromCanvasDC** functions.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.NoEmbedFontListGet(
   Index: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::NoEmbedFontListGet(
   Index As Long) As String
```

DLL

```
wchar_t * DPLNoEmbedFontListGet(int InstanceID, int Index);
```

Parameters

Index	The index of the font name in the list. The first name has an Index value of 1.

NoEmbedFontListRemoveAll

Vector graphics, Fonts, Miscellaneous functions

Version history

This function was introduced in Quick PDF Library version 7.23.

Description

Removes all the font names from the no embed font list used by the **LoadFromCanvasDC** and **NewPageFromCanvasDC** functions.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.NoEmbedFontListRemoveAll: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::NoEmbedFontListRemoveAll As Long
```

DLL

```
int DPLNoEmbedFontListRemoveAll(int InstanceID);
```

NoEmbedFontListRemoveIndex

Vector graphics, Fonts, Miscellaneous functions

Version history

This function was introduced in Quick PDF Library version 7.23.

Description

Removes the font name at the specified index from the no embed font list used by the **LoadFromCanvasDC** and **NewPageFromCanvasDC** functions.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.NoEmbedFontListRemoveIndex(
  Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::NoEmbedFontListRemoveIndex(
  Index As Long) As Long
```

DLL

```
int DPLNoEmbedFontListRemoveIndex(int InstanceID, int Index);
```

Parameters

Index	The index of the font name in the list. The first name has an Index value of 1.

Return values

0	The specified index was out of range
1	The font name was successfully removed from the list

NoEmbedFontListRemoveName

Vector graphics, Fonts, Miscellaneous functions

Version history

This function was introduced in Quick PDF Library version 7.23.

Description

Removes the specified font name from the no embed font list used by the **LoadFromCanvasDC** and **NewPageFromCanvasDC** functions.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.NoEmbedFontListRemoveName(
  FontName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::NoEmbedFontListRemoveName(
  FontName As String) As Long
```

DLL

```
int DPLNoEmbedFontListRemoveName(int InstanceID, wchar_t * FontName);
```

Parameters

FontName	The font name to remove from the list

Return values

0	The specified font name was not found in the list
1	The font name was successfully removed from the list

NormalizePage

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Version history

This function was introduced in Quick PDF Library version 7.25.

Description

Moves and/or rotates the contents of the page so that subsequent drawing operations are at the expected position on the page. All the page boundary boxes are adjusted to the physical size of the page and the page's rotation attribute is reset to zero.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.NormalizePage(
  NormalizeOptions: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::NormalizePage(
  NormalizeOptions As Long) As Long
```

DLL

```
int DPLNormalizePage(int InstanceID, int NormalizeOptions);
```

Parameters

NormalizeOptions	0 = Standard normalization
	1 = Normalize and also balance the graphics state stack
	2 = Maintain existing page structure
	3 = Maintain existing page structure and balance the stack

OpenOutline

Description

Expands an outline item (bookmark).

Syntax

Delphi

```
function TDebenuPDFLibrary1115.OpenOutline(OutlineID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::OpenOutline(
  OutlineID As Long) As Long
```

DLL

```
int DPLOpenOutline(int InstanceID, int OutlineID);
```

Parameters

OutlineID	The ID of the outline item to work with. This ID is returned by the **NewOutline** or **NewStaticOutline** functions, or retrieved with the **GetOutlineID** function or Get*Outline functions.

Return values

0	The Outline ID provided was invalid
1	The outline item was expanded

OptionalContentGroupCount

Content Streams and Optional Content Groups

Description

Returns the number of optional content groups in the selected document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.OptionalContentGroupCount: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::OptionalContentGroupCount As Long
```

DLL

```
int DPLOptionalContentGroupCount(int InstanceID);
```

OutlineCount

Description

Returns the number of outline items (bookmarks) in the selected document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.OutlineCount: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::OutlineCount As Long
```

DLL

```
int DPLOutlineCount(int InstanceID);
```

OutlineTitle

Description

Returns the title of an outline item (bookmark).

Syntax

Delphi

```
function TDebenuPDFLibrary1115.OutlineTitle(
  OutlineID: Integer): WideString;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::OutlineTitle(
  OutlineID As Long) As String
```

DLL

```
wchar_t * DPLOutlineTitle(int InstanceID, int OutlineID);
```

Parameters

OutlineID	The ID of the outline item to work with. This ID is returned by the **NewOutline** or **NewStaticOutline** functions, or retrieved with the **GetOutlineID** function or Get*Outline functions.

PageCount

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Reports the total number of pages in the selected document.

Syntax

Delphi
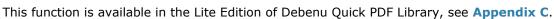
```
function TDebenuPDFLibrary1115.PageCount: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::PageCount As Long
```

DLL

```
int DPLPageCount(int InstanceID);
```

PageHasFontResources

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Version history

This function was introduced in Quick PDF Library version 9.16.

Description

Analyses the specified page to identify font resources.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.PageHasFontResources(
  PageNumber: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::PageHasFontResources(
  PageNumber As Long) As Long
```

DLL

```
int DPLPageHasFontResources(int InstanceID, int PageNumber);
```

Parameters

PageNumber	The number of the page to anaylse

Return values

0	The specified page does not have font resources
1	The specified page has at least one font resource

PageHeight

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Returns the height of the selected page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.PageHeight: Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::PageHeight As Double
```

DLL

```
double DPLPageHeight(int InstanceID);
```

Return values

The height of the selected page (in points, millimetres or inches)

PageJavaScriptAction

JavaScript, Page properties

Description

This function is used to add JavaScript to a page open or page close event.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.PageJavaScriptAction(ActionType,
  JavaScript: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::PageJavaScriptAction(
  ActionType As String, JavaScript As String) As Long
```

DLL

```
int DPLPageJavaScriptAction(int InstanceID, wchar_t * ActionType,
  wchar_t * JavaScript);
```

Parameters

ActionType	The event to add the JavaScript to: "O" = (capital letter O) This event occurs when the page is opened "C" = This event occurs when the page is closed
JavaScript	This is the JavaScript to execute when the event occurs.

Return values

0	The specified ActionType was not valid
1	The JavaScript was added successfully

PageRotation

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Returns the rotation of the selected page. This value should always be a multiple of 90 degrees.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.PageRotation: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::PageRotation As Long
```

DLL

```
int DPLPageRotation(int InstanceID);
```

PageWidth

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Returns the width of the selected page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.PageWidth: Double;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::PageWidth As Double
```

DLL

```
double DPLPageWidth(int InstanceID);
```

Return values

The width of the selected page (in points, millimetres or inches)

PrintDocument

Description

Renders certain pages from the selected document to the specified printer.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.PrintDocument(PrinterName: WideString;
  StartPage, EndPage, Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::PrintDocument(
  PrinterName As String, StartPage As Long, EndPage As Long,
  Options As Long) As Long
```

DLL

```
int DPLPrintDocument(int InstanceID, wchar_t * PrinterName, int StartPage,
  int EndPage, int Options);
```

Parameters

PrinterName	The name of the printer to use for printing. This is the name that appears in the Windows Print Manager. Use the **GetPrinterNames** function to return a list of valid printers on the system. A value returned by the **NewCustomPrinter** function can also be used here.
StartPage	The first page to print
EndPage	The last page to print
Options	Use the **PrintOptions** function to obtain a value for this parameter

Return values

0	The pages could not be printed, usually caused by the StartPage and EndPage parameters being out of range
1	The pages were printed successfully

PrintDocumentToFile

Version history

This function was introduced in Quick PDF Library version 7.18.

Description

Renders certain pages from the selected document to the specified printer. The print output is directed to the specified spool file.

Not all printer drivers support the DocInfo.lpszOutput field so results may vary.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.PrintDocumentToFile(
  PrinterName: WideString; StartPage, EndPage, Options: Integer;
  FileName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::PrintDocumentToFile(
  PrinterName As String, StartPage As Long, EndPage As Long,
  Options As Long, FileName As String) As Long
```

DLL

```
int DPLPrintDocumentToFile(int InstanceID, wchar_t * PrinterName,
  int StartPage, int EndPage, int Options, wchar_t * FileName);
```

Parameters

PrinterName	The name of the printer to use for printing. This is the name that appears in the Windows Print Manager. Use the **GetPrinterNames** function to return a list of valid printers on the system. A value returned by the **NewCustomPrinter** function can also be used here.
StartPage	The first page to print
EndPage	The last page to print
Options	Use the **PrintOptions** function to obtain a value for this parameter
FileName	The file name where print output should be spooled to.

PrintDocumentToPrinterObject

Rendering and printing

Version history

This function was introduced in Quick PDF Library version 7.11.

Description

Renders certain pages from the selected document to the printer specified by the Delphi TPrinter object.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.PrintDocumentToPrinterObject(
  APrinter: TPrinter; StartPage, EndPage, Options: Integer): Integer;
```

Parameters

APrinter	A Delph TPrinter object
StartPage	The first page to print
EndPage	The last page to print
Options	Use the **PrintOptions** function to obtain a value for this parameter

PrintMode

Version history

This function was introduced in Quick PDF Library version 11.11.

Description

This function is used to handle printing process.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.PrintMode(Mode: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::PrintMode(Mode As Long) As Long
```

DLL

```
int DPLPrintMode(int InstanceID, int Mode);
```

Parameters

Mode	0 = Smaller size, normal quality
	1 = Higher size, higher quality
	2 = Lossless quality

Return values

The current printing mode

PrintOptions

Description

This function is used to construct a value that can be used as the Options parameter to the **PrintDocument** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.PrintOptions(PageScaling,
  AutoRotateCenter: Integer; Title: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::PrintOptions(
  PageScaling As Long, AutoRotateCenter As Long,
  Title As String) As Long
```

DLL

```
int DPLPrintOptions(int InstanceID, int PageScaling, int AutoRotateCenter,
  wchar_t * Title);
```

Parameters

PageScaling	0 = None 1 = Fit to paper 2 = Shrink large pages
AutoRotateCenter	0 = Do not rotate pages automatically 1 = Rotate pages to fit on the output medium, and center on the page -1 = Rotate pages to fit on the output medium, and center on the page but rotate anticlockwise instead.
Title	The title of the document. This title is used by Windows in the Print Manager and for network title pages

PrintPages

Version history

This function was introduced in Quick PDF Library version 9.14.

Description

Renders a page range list from the selected document to the specified printer.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.PrintPages(PrinterName,
  PageRanges: WideString; Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::PrintPages(
  PrinterName As String, PageRanges As String,
  Options As Long) As Long
```

DLL

```
int DPLPrintPages(int InstanceID, wchar_t * PrinterName,
  wchar_t * PageRanges, int Options);
```

Parameters

PrinterName	The name of the printer to use for printing. This is the name that appears in the Windows Print Manager. Use the **GetPrinterNames** function to return a list of valid printers on the system. A value returned by the **NewCustomPrinter** function can also be used here.
PageRanges	A list of pages to print, for example "1-10,12,14"
Options	Use the **PrintOptions** function to obtain a value for this parameter

Return values

0	An error occurred
1	The pages were printed successfully

PrintPagesToFile

Version history

This function was introduced in Quick PDF Library version 9.14.

Description

Renders a list of page ranges from the selected document to the specified printer. The print output is directed to the specified spool file.

Not all printer drivers support the DocInfo.lpszOutput field so results may vary.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.PrintPagesToFile(PrinterName,
  PageRanges: WideString; Options: Integer; FileName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::PrintPagesToFile(
  PrinterName As String, PageRanges As String, Options As Long,
  FileName As String) As Long
```

DLL

```
int DPLPrintPagesToFile(int InstanceID, wchar_t * PrinterName,
  wchar_t * PageRanges, int Options, wchar_t * FileName);
```

Parameters

PrinterName	The name of the printer to use for printing. This is the name that appears in the Windows Print Manager. Use the **GetPrinterNames** function to return a list of valid printers on the system. A value returned by the **NewCustomPrinter** function can also be used here.
PageRanges	A list of pages to print, for example "1-10,12,14"
Options	Use the **PrintOptions** function to obtain a value for this parameter
FileName	Use the **PrintOptions** function to obtain a value for this parameter

Return values

0	An error occurred
1	The pages were printed successfully

PrintPagesToPrinterObject

Version history

This function was introduced in Quick PDF Library version 9.14.

Description

Renders a page range list from the selected document to the printer specified by the Delphi TPrinter object.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.PrintPagesToPrinterObject(
  APrinter: TPrinter; PageRanges: WideString; Options: Integer): Integer;
```

Parameters

APrinter	A Delph TPrinter object
PageRanges	A list of pages to print, for example "1-10,12,14"
Options	Use the **PrintOptions** function to obtain a value for this parameter

ReleaseBuffer

Version history

This function was introduced in Quick PDF Library version 7.11.

Description

Releases a buffer created with the **CreateBuffer** function.

Syntax

DLL

```
int DPLReleaseBuffer(int InstanceID, char * Buffer);
```

Parameters

Buffer	A value returned from the **CreateBuffer** function

Return values

0	The InstanceID was invalid, or the Buffer has already been released or is invalid
1	The buffer was released successfully

ReleaseImage

Image handling

Version history

This function was introduced in Quick PDF Library version 8.15.

Description

Releases the temporary memory used by an image that was added to the PDF after the document was opened (using functions such as **AddImageFromFile**) or an image that was found using the **FindImages** function.

Releasing the image does not affect the PDF itself, images that have already been drawn onto the page will not be removed.

After the image has been released the ImageID is no longer valid and cannot be used with functions such as **SelectImage**.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ReleaseImage(ImageID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ReleaseImage(
  ImageID As Long) As Long
```

DLL

```
int DPLReleaseImage(int InstanceID, int ImageID);
```

Parameters

ImageID	The ID of the image to release

Return values

0	The image could not be released. The ImageID parameter could be invalid or the ImageID doesn't reference an image contained in the selected document.
1	The image was released successfully.

ReleaseImageList

Image handling, Page properties

Version history

This function was introduced in Quick PDF Library version 8.15.

Description

Releases the specified image list including all the image data extracted from the images in the list. Releasing the image list does not affect the original images.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ReleaseImageList(
  ImageListID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ReleaseImageList(
  ImageListID As Long) As Long
```

DLL

```
int DPLReleaseImageList(int InstanceID, int ImageListID);
```

Parameters

ImageListID	A value returned by the **GetPageImageList** function

Return values

0	The image list could not be released. The ImageListID parameter could be invalid or the ImageListID doesn't reference an image list from the selected document.
1	The image list was released successfully.

ReleaseLibrary

Miscellaneous functions

Version history

This function was introduced in Quick PDF Library version 7.11.

Description

Frees the object created with the **CreateLibrary** function.

Syntax

DLL

```
int DPLReleaseLibrary(int InstanceID);
```

Return values

0	The library could not be released. The InstanceID value may be incorrect.
1	The library was released successfully

ReleaseSignProcess

Version history

This function was introduced in Quick PDF Library version 9.14.

Description

Releases a signature process from memory.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ReleaseSignProcess(
  SignProcessID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ReleaseSignProcess(
  SignProcessID As Long) As Long
```

DLL

```
int DPLReleaseSignProcess(int InstanceID, int SignProcessID);
```

Parameters

SignProcessID	A value returned by the **NewSignProcessFromFile**, **NewSignProcessFromStream** or **NewSignProcessFromString** functions.

Return values

0	Invalid SignProcessID
1	Successfully deleted the signing process

ReleaseStringList

Version history

This function was introduced in Quick PDF Library version 9.14.

Description

Releases the specified string list.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ReleaseStringList(
  StringListID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ReleaseStringList(
  StringListID As Long) As Long
```

DLL

```
int DPLReleaseStringList(int InstanceID, int StringListID);
```

Parameters

StringListID	The ID of the string list as returned by the **CheckFileCompliance** function.

Return values

0	The string list could not be released, the StringListID parameter is invalid.
1	Success

ReleaseTextBlocks

Version history

This function was introduced in Quick PDF Library version 9.13.

Description

Releases the memory used by a text block list.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ReleaseTextBlocks(
  TextBlockListID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ReleaseTextBlocks(
  TextBlockListID As Long) As Long
```

DLL

```
int DPLReleaseTextBlocks(int InstanceID, int TextBlockListID);
```

Parameters

TextBlockListID   A value returned by the **ExtractPageTextBlocks** function

RemoveAppearanceStream

Form fields

Description

Removes the appearance stream of the specified form field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.RemoveAppearanceStream(
  Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::RemoveAppearanceStream(
  Index As Long) As Long
```

DLL

```
int DPLRemoveAppearanceStream(int InstanceID, int Index);
```

Parameters

Index	The index of the form field to work with. The first form field has an index of 1.

Return values

0	The form field could not be found
1	The appearance stream of the form field was removed successfully

RemoveCustomInformation

Version history

This function was introduced in Quick PDF Library version 7.24.

Description

Removes a custom metadata item from the document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.RemoveCustomInformation(
  Key: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::RemoveCustomInformation(
  Key As String) As Long
```

DLL

```
int DPLRemoveCustomInformation(int InstanceID, wchar_t * Key);
```

Parameters

Key	Specifies which key to remove

RemoveDocument

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Removes the specified document, freeing up memory.

Quick PDF Library will always ensure that there is at least one document loaded at all times.

In version 7.18 and earlier, it was only possible to remove a document if there were at least two documents loaded.

From version 7.19 this function will always succeed. If the specified document was the only loaded document it will be removed and replaced with a new blank document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.RemoveDocument(
  DocumentID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::RemoveDocument(
  DocumentID As Long) As Long
```

DLL

```
int DPLRemoveDocument(int InstanceID, int DocumentID);
```

Parameters

DocumentID	The ID of the document to remove

Return values

0	The specified document does not exist or could not be removed.
1	The specified document was removed successfully

RemoveEmbeddedFile

Version history

This function was introduced in Quick PDF Library version 7.19.

Description

Removes the specified embedded file from the document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.RemoveEmbeddedFile(Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::RemoveEmbeddedFile(
  Index As Long) As Long
```

DLL

```
int DPLRemoveEmbeddedFile(int InstanceID, int Index);
```

Parameters

Index	The index of the embedded file. Must be a value between 1 and the value returned by **EmbeddedFileCount**.

Return values

0	The embedded file could not be removed.
1	The embedded file was successfully removed from the document.

RemoveFormFieldBackgroundColor

Form fields

Version history

This function was introduced in Quick PDF Library version 9.12.

Description

Removes the form field's background color entry

Syntax

Delphi

```
function TDebenuPDFLibrary1115.RemoveFormFieldBackgroundColor(
  Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::RemoveFormFieldBackgroundColor(
  Index As Long) As Long
```

DLL

```
int DPLRemoveFormFieldBackgroundColor(int InstanceID, int Index);
```

Parameters

Index	The index of the form field

Return values

0	The Index parameter was incorrect
1	Success

RemoveFormFieldBorderColor

Form fields

Version history

This function was introduced in Quick PDF Library version 8.14.

Description

Removes the form field's border color entry

Syntax

Delphi

```
function TDebenuPDFLibrary1115.RemoveFormFieldBorderColor(
  Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::RemoveFormFieldBorderColor(
  Index As Long) As Long
```

DLL

```
int DPLRemoveFormFieldBorderColor(int InstanceID, int Index);
```

Parameters

Index	The index of the form field

Return values

0	The Index parameter was incorrect
1	Success

RemoveFormFieldChoiceSub

Form fields

Version history

This function was introduced in Quick PDF Library version 10.12.

Description

Removes a subname entry from a choice based form field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.RemoveFormFieldChoiceSub(Index: Integer;
  SubName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::RemoveFormFieldChoiceSub(
  Index As Long, SubName As String) As Long
```

DLL

```
int DPLRemoveFormFieldChoiceSub(int InstanceID, int Index,
  wchar_t * SubName);
```

Parameters

Index	The index of the form field
SubName	The string value of the subname to delete

Return values

0	The subname was not deleted. The specified form field may not have been a choice form field.
1	The subname was successfully deleted

RemoveGlobalJavaScript

Document properties, JavaScript

Version history

This function was introduced in Quick PDF Library version 7.19.

Description

Removes a block of JavaScript from the global JavaScript store.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.RemoveGlobalJavaScript(
  PackageName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::RemoveGlobalJavaScript(
  PackageName As String) As Long
```

DLL

```
int DPLRemoveGlobalJavaScript(int InstanceID, wchar_t * PackageName);
```

Parameters

PackageName	The name that that JavaScript was stored under.

Return values

0	The specified package name could not be found
1	The JavaScript was removed successfully

RemoveOpenAction

Version history

This function was introduced in Quick PDF Library version 9.12.

Description

Removes any open action from the document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.RemoveOpenAction: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::RemoveOpenAction As Long
```

DLL

```
int DPLRemoveOpenAction(int InstanceID);
```

Return values

0	An unexpected error occurred
1	The open action, if any, was removed from the document successfully

RemoveOutline

Description

Removes an outline from the document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.RemoveOutline(OutlineID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::RemoveOutline(
  OutlineID As Long) As Long
```

DLL

```
int DPLRemoveOutline(int InstanceID, int OutlineID);
```

Parameters

OutlineID	The ID of the outline item to work with. This ID is returned by the **NewOutline** or **NewStaticOutline** functions, or retrieved with the **GetOutlineID** function or Get*Outline functions.

Return values

0	The Outline ID provided was invalid
1	The outline was removed successfully

RemovePageBox

Version history

This function was introduced in Quick PDF Library version 10.13.

Description

Removes the specified boundary rectangle from selected page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.RemovePageBox(BoxType: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::RemovePageBox(
  BoxType As Long) As Long
```

DLL

```
int DPLRemovePageBox(int InstanceID, int BoxType);
```

Parameters

BoxType	1 = MediaBox (disabled for now)
	2 = CropBox
	3 = BleedBox
	4 = TrimBox
	5 = ArtBox

Return values

0	The specified boundary rectangle was not found.
1	The specified boundary rectangle was removed successfully.

RemoveSharedContentStreams

Content Streams and Optional Content Groups

Version history

This function was renamed in Quick PDF Library version 8.11.
The function name in earlier versions was RemoveSharedLayers.

Description

This function ensures that none of the pages in the selected document have shared content streams. This is necessary before imposing a document with the **CapturePage** or **CapturePageEx** functions.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.RemoveSharedContentStreams: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::RemoveSharedContentStreams As Long
```

DLL

```
int DPLRemoveSharedContentStreams(int InstanceID);
```

RemoveStyle

Description

Removes a style that was previously saved using the **SaveStyle** function. The style name is case sensitive, it must exactly match the style name used with the **SaveStyle** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.RemoveStyle(StyleName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::RemoveStyle(
  StyleName As String) As Long
```

DLL

```
int DPLRemoveStyle(int InstanceID, wchar_t * StyleName);
```

Parameters

StyleName	The name to associate with the style. This name is case sensitive.

Return values

0	The specified StyleName could not be found
1	The style was removed successfully

RemoveUsageRights

Version history

This function was introduced in Quick PDF Library version 7.25.

Description

Removes any usage rights from the document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.RemoveUsageRights: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::RemoveUsageRights As Long
```

DLL

```
int DPLRemoveUsageRights(int InstanceID);
```

Return values

0	Usage rights were not found in the document.
1	Usage rights were successfully removed from the document.

RemoveXFAEntries

Version history

This function was introduced in Quick PDF Library version 7.21.

Description

Removes the XFA form field entry from the document's form.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.RemoveXFAEntries(Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::RemoveXFAEntries(
  Options As Long) As Long
```

DLL

```
int DPLRemoveXFAEntries(int InstanceID, int Options);
```

Parameters

Options	Reserved for future use, should be set to 0.

RenderAsMultipageTIFFToFile

Image handling, Rendering and printing

Version history

This function was introduced in Quick PDF Library version 10.11.

Description

Renders the specified pages from the selected document to a multi-page TIFF file.

ImageOptions 1, TIFF (G4) output, is only available on Windows Vista and Windows Server 2008 and later.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.RenderAsMultipageTIFFToFile(DPI: Double;
  PageRanges: WideString; ImageOptions, OutputOptions: Integer;
  FileName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::RenderAsMultipageTIFFToFile(
  DPI As Double, PageRanges As String, ImageOptions As Long,
  OutputOptions As Long, FileName As String) As Long
```

DLL

```
int DPLRenderAsMultipageTIFFToFile(int InstanceID, double DPI,
  wchar_t * PageRanges, int ImageOptions, int OutputOptions,
  wchar_t * FileName);
```

Parameters

DPI	The DPI to render the pages at
PageRanges	A list of pages to render, for example "5-10,3,12".
ImageOptions	0=24-bit RGB TIFF 1=1-bit G4 TIFF
OutputOptions	Reserved for future use, should be set to 0.
FileName	The file name and path of the TIFF file to create

Return values

0	Invalid parameters or cannot create file
1	The multipage TIFF was created successfully

RenderDocumentToFile

Description

Renders certain pages from the selected document to an image file on disk.

By default rendering uses the GDI+ system which is available by default in Windows XP and later.

Option 10, TIFF (G4) output, is only available on Windows Vista and Windows Server 2008 and later.

It is also possible to render using Cairo, use the **SetCairoFileName** and **SelectRenderer** functions.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.RenderDocumentToFile(DPI: Double;
  StartPage, EndPage, Options: Integer; FileName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::RenderDocumentToFile(
  DPI As Double, StartPage As Long, EndPage As Long,
  Options As Long, FileName As String) As Long
```

DLL

```
int DPLRenderDocumentToFile(int InstanceID, double DPI, int StartPage,
  int EndPage, int Options, wchar_t * FileName);
```

Parameters

DPI	The DPI to use for the rendering. A value of 72 will give the same result as Acrobat when the zoom level is 100%.
StartPage	The first page to print
EndPage	The last page to print
Options	0 = BMP output 1 = JPEG output 2 = WMF output 3 = EMF output 4 = EPS output 5 = PNG output 6 = GIF output 7 = TIFF output 8 = EMF+ output 9 = HTML5 output 10 = G4 TIFF output
FileName	The path and filename to use for the file. Each page will be stored in a separate file. If this parameter contains "%p" this will be replaced by the page number, otherwise the page number will be appended to the end of the filename before the extension. For example, if FileName is "output.jpg" and page 10 is rendered the image will be stored in a file called "output10.jpg". If FileName is "page%poutput.bmp" and page 5 is rendered the image will be stored in a file called "page5output.bmp".

Return values

0	The pages were not rendered successfully. This is usually caused by the StartPage or EndPage parameters being out of range.
1	The pages were rendered successfully

RenderPageToDC

Rendering and printing

Version history

This function was introduced in Quick PDF Library version 7.12.

Description

This function renders a page from the selected document directly onto a graphics surface.

On Windows the target surface is a Device Context handle (DC).

By default rendering uses the GDI+ system which is available by default in Windows XP and later.

It is also possible to render using Cairo, use the **SetCairoFileName** and **SelectRenderer** functions.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.RenderPageToDC(DPI: Double; Page: Integer;
  DC: HDC): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::RenderPageToDC(DPI As Double,
  Page As Long, DC As Long) As Long
```

DLL

```
int DPLRenderPageToDC(int InstanceID, double DPI, int Page, HDC DC);
```

Parameters

DPI	The DPI to use when rendering the page
Page	The page number to render
DC	The device context handle

Return values

0	Page could not be rendered
1	Page was rendered successfully

RenderPageToDCClip

Description

This function renders a page from the selected document directly onto a graphics surface with a clip paths. It is possible to use a render offset for clip paths definitions **SetRenderDCOffset**

On Windows the target surface is a Device Context handle (DC).

By default rendering uses the GDI+ system which is available by default in Windows XP and later.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.RenderPageToDCClip(DPI: Double; Page,
  DC: Integer; const Clip: AnsiString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::RenderPageToDCClip(
  DPI As Double, Page As Long, DC As Long,
  Clip As String) As Long
```

DLL

```
int DPLRenderPageToDCClip(int InstanceID, double DPI, int Page, int DC,
  char * Clip);
```

Parameters

DPI	Rendering DPI
Page	Page number to render
DC	The device context handle
Clip	Initial clip rectangle array, defined by left top positions, widths and heights with comma delimiters (L1 T1 W1 H1 L2 T2 W2 H2 ... Ln Tn Wn Hn)

Return values

1	Render successfull
0	Render failed

RenderPageToFile

Description

This function renders a page from the selected document to a file on disk. The data written to disk depends on the Options parameter.

By default rendering uses the GDI+ system which is available by default in Windows XP and later.

Option 10, TIFF (G4) output, is only available on Windows Vista and Windows Server 2008 and later.

It is also possible to render using Cairo, use the **SetCairoFileName** and **SelectRenderer** functions.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.RenderPageToFile(DPI: Double; Page,
  Options: Integer; FileName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::RenderPageToFile(DPI As Double,
  Page As Long, Options As Long, FileName As String) As Long
```

DLL

```
int DPLRenderPageToFile(int InstanceID, double DPI, int Page, int Options,
  wchar_t * FileName);
```

Parameters

DPI	The DPI to use when rendering the page. Values over 300 will cause excessive memory usage.
Page	The page number to render
Options	0 = BMP output 1 = JPEG output 2 = WMF output 3 = EMF output 4 = EPS output 5 = PNG output 6 = GIF output 7 = TIFF (LZW) output 8 = EMF+ output 9 = HTML5 output 10 = TIFF (G4) output
FileName	The path and file name of the file to create to store the rendered page image data in.

Return values

0	The page could not be rendered
1	The page was rendered correctly and the image file was saved to disk
2	The file could not be written to disk

RenderPageToStream

Description

This function is only available in the Delphi edition. It renders a page from the selected document to a TStream object. The data placed into the stream depends on the Options parameter.

By default rendering uses the GDI+ system which is available by default in Windows XP and later.

Option 10, TIFF (G4) output, is only available on Windows Vista and Windows Server 2008 and later.

It is also possible to render using Cairo, use the **SetCairoFileName** and **SelectRenderer** functions.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.RenderPageToStream(DPI: Double; Page,
  Options: Integer; Target: TStream): Integer;
```

Parameters

DPI	The DPI to use when rendering the page. Values over 300 will cause excessive memory usage.
Page	The page number to render
Options	0 = BMP output 1 = JPEG output 2 = WMF output 3 = EMF output 4 = EPS output 5 = PNG output 6 = GIF output 7 = TIFF output 8 = EMF+ output 9 = HTML5 output 10 = TIFF (G4) output
Target	The stream to place the rendered page into

RenderPageToString

Description

This function renders a page from the selected document to a string. The data in the returned string depends on the Options parameter.

By default rendering uses the GDI+ system which is available by default in Windows XP and later.

Option 10, TIFF (G4) output, is only available on Windows Vista and Windows Server 2008 and later.

It is also possible to render using Cairo, use the **SetCairoFileName** and **SelectRenderer** functions.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.RenderPageToString(DPI: Double; Page,
  Options: Integer): AnsiString;
```

DLL

```
char * DPLRenderPageToString(int InstanceID, double DPI, int Page,
  int Options);
```

Parameters

DPI	The DPI to use when rendering the page. Values over 300 will cause excessive memory usage.
Page	The page number to render
Options	0 = BMP output 1 = JPEG output 2 = WMF output 3 = EMF output 4 = EPS output 5 = PNG output 6 = GIF output 7 = TIFF output 8 = EMF+ output 9 = HTML5 output 10 = TIFF (G4) output

RenderPageToVariant

Description

This function is only available in the ActiveX edition. It renders a page from the selected document to a byte array Variant. The data in the byte array depends on the Options parameter.

By default rendering uses the GDI+ system which is available by default in Windows XP and later.

Option 10, TIFF (G4) output, is only available on Windows Vista and Windows Server 2008 and later.

It is also possible to render using Cairo, use the **SetCairoFileName** and **SelectRenderer** functions.

Syntax

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::RenderPageToVariant(
   DPI As Double, Page As Long, Options As Long) As Variant
```

Parameters

DPI	The DPI to use when rendering the page. Values over 300 will cause excessive memory usage.
Page	The page number to render
Options	0 = BMP output 1 = JPEG output 2 = WMF output 3 = EMF output 4 = EPS output 5 = PNG output 6 = GIF output 7 = TIFF output 8 = EMF+ output 9 = HTML5 output 10 = G4 TIFF output

ReplaceFonts

Description

Replaces embedded fonts with equivalent "Standard" fonts, reducing the file size. In version 9.11 and earlier, only Courier and Courier-Bold were replaced. As of version 9.12, fonts are replaced with the one of the 14 "Standard" fonts listed below:

Courier
Courier-Bold
Courier-BoldOblique
Courier-Oblique
Helvetica
Helvetica-Bold
Helvetica-BoldOblique
Helvetica-Oblique
Times-Roman
Times-Bold
Times-Italic
Times-BoldItalic
Symbol
ZapfDingbats

For example, ArialMT font will be replaced with Helvetica "Standard". Note that the "Standard" fonts do not contain the full Unicode character set but only the 229 characters defined by the WinAnsiEncoding table.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ReplaceFonts(Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ReplaceFonts(
  Options As Long) As Long
```

DLL

```
int DPLReplaceFonts(int InstanceID, int Options);
```

Parameters

Options	0 = Default. Standard and much preferrered level of font replacement
	1 = Special option to replace fonts and also remove the subsetted fonts and also the WinAnsiEncoding entry. This will only work if there is no other Encoding entry and is for very specific documents only.
	10 = Do not replace the FontDescriptor but delete the FontFile2 embedded font data. This will turn an embedded font into a non embedded font for fonts that match closely to a Standard font such as ArialMT. Most viewers will use Arial to render ArialMT anyway if it is not embedded.

Return values

0	Nothing was changed
1	A replacement of some type was made.

ReplaceImage

Description

Replaces an image on the selected page with another image.

The original image is not removed from the document and can be reused. If the original image is no longer needed it can be cleared using the **ClearImage** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ReplaceImage(OriginalImageID,
  NewImageID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ReplaceImage(
  OriginalImageID As Long, NewImageID As Long) As Long
```

DLL

```
int DPLReplaceImage(int InstanceID, int OriginalImageID, int NewImageID);
```

Parameters

OriginalImageID	The ImageID of the image to be replaced
NewImageID	The ImageID of the image to replace the existing image

ReplaceTag

Description

This function searches through the contents of the current page, and replaces all occurrences of Tag with NewValue.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ReplaceTag(Tag,
  NewValue: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ReplaceTag(Tag As String,
  NewValue As String) As Long
```

DLL

```
int DPLReplaceTag(int InstanceID, wchar_t * Tag, wchar_t * NewValue);
```

Parameters

Tag	The text to search for
NewValue	The replacement text

Return values

Returns the number of times the text was replaced

RequestPrinterStatus

Rendering and printing

Description

Use this function to activate an alternative printing system that allows the printer status to be returned. Many of the status codes returned are supplied by the printer driver there is no guarantee that values will contain meaningful information for all printers.

The first step is to call this function with StatusCommand=101 to enable printer status monitoring. Optionally, the print job can be started in the paused state by calling this function again with StatusCommand=103. This might be necessary for small print jobs that would otherwise finish before the status can be read.

The print job can then be started as usual with one of the printing functions: **PrintDocument**, **PrintDocumentToFile** or **PrintDocumentToPrinterObject**.

Once the print job has started, this function can be called again repeatedly to obtain the printer status for the print job over time. If the print job was started in the paused state, actual printing will only begin once this function is called with StatusCommand=402.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.RequestPrinterStatus(
  StatusCommand: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::RequestPrinterStatus(
  StatusCommand As Long) As Long
```

DLL

```
int DPLRequestPrinterStatus(int InstanceID, int StatusCommand);
```

Parameters

StatusCommand
100 = Turn off printer status monitoring.
101 = Turn on printer status monitoring.
102 = Returns 1 if printer status monitoring is active.
103 = Start print job in paused state.
104 = Start printing immediately.
105 = Returns 1 if print job will be started in paused state.
200 = Returns 1 if print job data exists.
201 = Returns the Windows Spooler JobID.
202 = Returns the job priority, from 1 to 99.
203 = Returns the job's position in the print queue.
204 = Returns the total page count.
205 = Returns the number of pages that have been printed. This is usually zero if the data type is "RAW".
206 = Returns the number of milliseconds since the print job was started.

207 = Returns the print job data type
Returns 1 if the data type contains "RAW"
Returns 2 if the data type contains "EMF"
Returns 3 if the data type contains "TEXT"
Returns 4 if the data type contains "XPS"

300 = Returns the encoded job status.
301 = Returns 1 if the job is paused.
302 = Returns 1 if there is an error.
303 = Returns 1 if the job is being deleted.
304 = Returns 1 if the job is spooling.
305 = Returns 1 if the job is printing.
306 = Returns 1 if the printer is offline.
307 = Returns 1 if the printer is out of paper.
308 = Returns 1 if the job has printed.
309 = Returns 1 if the job has been deleted.
310 = Returns 1 if the driver cannot print the print job.
311 = Returns 1 if the printer has an error that requires the user to do something.
312 = Returns 1 if the job has been restarted.
313 = For Windows XP and later, returns 1 if the job has been sent to the printer (job may not be printed yet).
314 = For Windows Vista and later, returns 1 if the job has been retained in the print queue and cannot be deleted.

401 = Pause the print job.
402 = Resume a paused print job.
403 = Delete the print job.

RetrieveCustomDataToFile

Description

Retrieves custom data from the PDF that was previously stored with
StoreCustomDataFromString or **StoreCustomDataFromFile**. The retrieved data is written to
the specified file.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.RetrieveCustomDataToFile(Key,
  FileName: WideString; Location: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::RetrieveCustomDataToFile(
  Key As String, FileName As String, Location As Long) As Long
```

DLL

```
int DPLRetrieveCustomDataToFile(int InstanceID, wchar_t * Key,
  wchar_t * FileName, int Location);
```

Parameters

Key	The key that the data was stored under. If the location is the Document Catalog then the key must have a special prefix assigned to you by Adobe to avoid conflicts with other software. If the location is the Document Information Dictionary any key can be used but should be chosen with care so they make sense to the user.
FileName	The path and file name of the file to save the retrieved data to.
Location	1 = Retrieve the data from the Document Information Dictionary 2 = Retrieve the data from the Document Catalog

Return values

0	There was no data stored in the specified key, or the file to save the data to already exists and could not be overwritten
1	The data was retrieved and written to the specified file successfully

RetrieveCustomDataToString

Version history

> This function was renamed in Quick PDF Library version 7.11.
> The function name in earlier versions was RetrieveCustomData.

Description

> Retrieves custom data from the PDF that was previously stored with the StoreCustomData function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.RetrieveCustomDataToString(
  const Key: AnsiString; Location: Integer): AnsiString;
```

DLL

```
char * DPLRetrieveCustomDataToString(int InstanceID, char * Key,
  int Location);
```

Parameters

Key	The key that the data was stored under. If the location is the Document Catalog then the key must have a special prefix assigned to you by Adobe to avoid conflicts with other software. If the location is the Document Information Dictionary any key can be used but should be chosen with care so they make sense to the user.
Location	1 = Retrieve the data from the Document Information Dictionary 2 = Retrieve the data from the Document Catalog

RetrieveCustomDataToVariant

Description

This function is only available in the ActiveX edition. It retrieves custom data that was previously stored with the StoreCustomData function into a variant byte array.

Syntax

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::RetrieveCustomDataToVariant(
  Key As String, Location As Long) As Variant
```

Parameters

Key	The key that the data was stored under. If the location is the Document Catalog then the key must have a special prefix assigned to you by Adobe to avoid conflicts with other software. If the location is the Document Information Dictionary any key can be used but should be chosen with care so they make sense to the user.
Location	1 = Retrieve the data from the Document Information Dictionary 2 = Retrieve the data from the Document Catalog

ReverseImage

Description

This function reverses the interpretation of the color components in the selected image. For example, a green pixel (0, 255, 0) will become a purple pixel (255, 0, 255) and a black pixel will become a while pixel.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.ReverseImage(Reset: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::ReverseImage(
  Reset As Long) As Long
```

DLL

```
int DPLReverseImage(int InstanceID, int Reset);
```

Parameters

Reset	Indicates whether the /Decode parameter in the image dictionary should be removed. This is necessary when the image is used as a stencil mask in Acrobat 4.0, but may give different results for different source image types (BMP, TIFF and PNG). Experimentation will be necessary.
	0 = Keep the /Decode array and reverse the image
	1 = Remove the /Decode array

RotatePage

Page properties, Page manipulation

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Used to rotate the page by a multiple of 90 degrees. This will also rotate the co-ordinate system on the page so that it remains the same with respect to the orientation of the page. The rotation is absolute, for example calling the function twice with a parameter of 90 will result in a page rotated by 90 degrees, not 180 degrees.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.RotatePage(PageRotation: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::RotatePage(
  PageRotation As Long) As Long
```

DLL

```
int DPLRotatePage(int InstanceID, int PageRotation);
```

Parameters

PageRotation	The number of degrees to rotate the page by. Must be a multiple of 90 degrees (90, 180 or 270).

Return values

0	The page could not be rotated, probably because the rotation specified was not a multiple of 90
1	The page was rotated successfully

SaveFontToFile

Description

This function is useful for extracting fonts from a PDF that have been found with the **FindFonts** function. The TTF font data for the currently selected file will be saved. Only embedded TrueType fonts can be saved.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SaveFontToFile(
  FileName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SaveFontToFile(
  FileName As String) As Long
```

DLL

```
int DPLSaveFontToFile(int InstanceID, wchar_t * FileName);
```

Parameters

FileName	The path and file name of the file that should be created to store the font data in.

Return values

0	The font is not embedded so there is no font data to save to the file
1	The embedded font data was written to the file successfully

SaveImageListItemDataToFile

Version history

This function was introduced in Quick PDF Library version 8.13.

Description

Saves the image data of an image list item to a file on disk.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SaveImageListItemDataToFile(ImageListID,
  ImageIndex, Options: Integer; ImageFileName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SaveImageListItemDataToFile(
  ImageListID As Long, ImageIndex As Long, Options As Long,
  ImageFileName As String) As Long
```

DLL

```
int DPLSaveImageListItemDataToFile(int InstanceID, int ImageListID,
  int ImageIndex, int Options, wchar_t * ImageFileName);
```

Parameters

ImageListID	A value returned by the **GetPageImageList** function
ImageIndex	The index of the image in the list. The first image has an index of 1.
Options	Reserved for future use. Should be set to 0.
ImageFileName	The path and filename of the file to create

Return values

0	Image data could not be saved
1	Image data was saved successfully

SaveImageToFile

Description

Saves the selected image to a file on disk. Only certain images can be saved. If the **ImageType** function returns 0 then the image type is in an unsupported format and cannot be saved.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SaveImageToFile(
  FileName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SaveImageToFile(
  FileName As String) As Long
```

DLL

```
int DPLSaveImageToFile(int InstanceID, wchar_t * FileName);
```

Parameters

FileName	The name of the image file to create.

Return values

0	The image could not be saved. Either an image is not selected or the file could not be created.
1	The image was saved successfully

SaveImageToStream

Description

This function is only available in the Delphi editions of the library. Use this function to save the selected image to a stream. Only certain image types can be saved, see the **SaveImageToFile** function for further information.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SaveImageToStream(
  OutStream: TStream): Integer;
```

Parameters

OutStream	The image data will be written into this Delphi TStream object

Return values

0	The image data could not be saved. Either an image was not selected, or the image data was of an unsupported type.
1	The image data was saved to the stream successfully

SaveImageToString

Version history

This function was introduced in Quick PDF Library version 7.23.

Description

Use this function to save the selected image to a string. Only certain image types can be saved, see the **SaveImageToFile** function for further information.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SaveImageToString: AnsiString;
```

DLL

```
char * DPLSaveImageToString(int InstanceID);
```

SaveImageToVariant

Image handling

Version history

This function was introduced in Quick PDF Library version 7.23.

Description

Use this function to save the selected image to a variant byte array. Only certain image types can be saved, see the **SaveImageToFile** function for further information.

Syntax

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SaveImageToVariant As Variant
```

SaveState

Vector graphics, Page layout

Description

Saves the current graphics state, which can be loaded later with the **LoadState** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SaveState: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SaveState As Long
```

DLL

```
int DPLSaveState(int InstanceID);
```

SaveStyle

Description

Saves the current text properties under a named style. This style can then be applied quickly with a single call to the **ApplyStyle** function. The properties that are saved include the font name, font size, text color, alignment, underline and highlight style, spacing and scaling.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SaveStyle(StyleName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SaveStyle(
  StyleName As String) As Long
```

DLL

```
int DPLSaveStyle(int InstanceID, wchar_t * StyleName);
```

Parameters

StyleName	The name to associate with the style. This name is case sensitive.

SaveToFile

Document management

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Saves the selected document to a file on disk.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SaveToFile(FileName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SaveToFile(
  FileName As String) As Long
```

DLL

```
int DPLSaveToFile(int InstanceID, wchar_t * FileName);
```

Parameters

FileName	The name of the file to create.

Return values

0	The file could not be created
1	The file was created successfully

SaveToStream

Description

Similar to the **SaveToFile** function, but allows the PDF document to be written to a stream object.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SaveToStream(OutStream: TStream): Integer;
```

Parameters

OutStream	The stream object to write the document to

Return values

0	The document could not be saved
1	The document was saved to the stream successfully

SaveToString

Description

Similar to the **SaveToFile** function, but instead of creating a file the data for the PDF file is returned as a string.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SaveToString: AnsiString;
```

DLL

```
char * DPLSaveToString(int InstanceID);
```

SaveToVariant

Description

Similar to the **SaveToFile** function, but allows the PDF document to be written to a byte array variant.

Syntax

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SaveToVariant As Variant
```

Return values

Empty array	The document could not be generated
Array	A byte array containing the PDF data

SecurityInfo

Document properties, Security and Signatures

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Returns information about the security settings of the selected document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SecurityInfo(
  SecurityItem: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SecurityInfo(
  SecurityItem As Long) As Long
```

DLL

```
int DPLSecurityInfo(int InstanceID, int SecurityItem);
```

Parameters

SecurityItem	0 = Security Method
	1 = User Password
	2 = Owner Password
	3 = Printing
	4 = Changing the Document
	5 = Content Copying or Extraction
	6 = Authoring Comments and Form Fields
	7 = Form Field Fill-in or Signing
	8 = Content Accessibility Enabled
	9 = Document Assembly
	10 = Encryption Level
	11 = Opened with User password
	12 = Opened with Owner password
	13 = Variable Encryption Strength

Return values

0	None
1	Adobe Standard Security
2	No
3	Yes
4	Fully Allowed
5	Not Allowed
6	Allowed
7	40-bit RC4 (Acrobat 3.x, 4.x)
8	128-bit RC4 (Acrobat 5.x)
9	Unknown
10	Low resolution
11	Blank
12	128-bit AES (Acrobat 7)
13	256-bit AES (Acrobat 9)
14	Variable length RC4 (use SecurityItem=13 to determine the length)
15	256-bit AES (Acrobat X)

SelectContentStream

Version history

This function was renamed in Quick PDF Library version 8.11.
The function name in earlier versions was SelectLayer.

Description

A page in a PDF document has one or more content stream parts that together contain all the PDF page description commands for the page.

This function selects one of the selected page's content stream parts.

All drawing operations are only carried out on the selected content stream part.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SelectContentStream(
  NewIndex: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SelectContentStream(
  NewIndex As Long) As Long
```

DLL

```
int DPLSelectContentStream(int InstanceID, int NewIndex);
```

Parameters

NewIndex	The index of the content stream part to select. The first content stream part has an index of 1.

Return values

0	The specified layer could not be selected
1	The specified layer was selected successfully

SelectDocument

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Selects a document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SelectDocument(
  DocumentID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SelectDocument(
  DocumentID As Long) As Long
```

DLL

```
int DPLSelectDocument(int InstanceID, int DocumentID);
```

Parameters

DocumentID	The ID of the document to select

Return values

0	The document could not be selected, the ID could not be found
1	The specified document was selected successfully

SelectFont

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Select one of the fonts which have been added to the selected document. The FontID must be a valid ID as returned by one of the Add*Font functions or returned by **GetFontID** .

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SelectFont(FontID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SelectFont(
  FontID As Long) As Long
```

DLL

```
int DPLSelectFont(int InstanceID, int FontID);
```

Parameters

FontID	The ID of the font to select

Return values

0	The specified ID could not be found
1	The font was selected successfully

SelectImage

Image handling, Page layout

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Select one of the images that have been added to the selected document with the AddImage* functions or an ImageID returned using **GetImageListItemIntProperty** with property 405.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SelectImage(ImageID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SelectImage(
  ImageID As Long) As Long
```

DLL

```
int DPLSelectImage(int InstanceID, int ImageID);
```

Parameters

ImageID	The ID of the image to select

Return values

0	The specified ID could not be found
1	The image was selected successfully

SelectPage

Page layout, Page manipulation

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Selects a page of the selected document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SelectPage(PageNumber: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SelectPage(
  PageNumber As Long) As Long
```

DLL

```
int DPLSelectPage(int InstanceID, int PageNumber);
```

Parameters

PageNumber	The page to select

Return values

0	The specified page could not be found
1	The page was selected successfully

SelectRenderer

Rendering and printing

Version history

This function was introduced in Quick PDF Library version 8.13.

Description

Select the renderer to use during rendering. By default the GDI+ rendering engine is used.

If DPLR (AGG) is used, the **SetDPLRFileName** function should be used to set the path to the DPLR DLL. All rendering functions support this rendering engine.

The required DPLR rendering DLL for this function can be found in the 'Rendering and Printing Add-On' folder/ directory of the Debenu Quick PDF Library installation folder. Please see the README.TXT in this folder for futher explanation if required.

If Cairo is used, the **SetCairoFileName** function should be used to set the path to the Cairo DLL. This rendering engine is being deprecated, you should switch to using DPLR instead.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SelectRenderer(
  RendererID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SelectRenderer(
  RendererID As Long) As Long
```

DLL

```
int DPLSelectRenderer(int InstanceID, int RendererID);
```

Parameters

RendererID	1 = GDI+ 2 = Cairo 3 = DPLR (AGG)

Return values

0	The specified renderer could not be selected
1	The GDI+ renderer was selected
2	The Cairo renderer was selected
3	The DPLR (AGG) renderer was selected

SelectedDocument

Document management

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Returns the ID of the selected document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SelectedDocument: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SelectedDocument As Long
```

DLL

```
int DPLSelectedDocument(int InstanceID);
```

Return values

0	A document has not been selected. This should never occur.
Non-zero	The ID of the selected document

SelectedFont

Description

Returns the ID of the selected font.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SelectedFont: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SelectedFont As Long
```

DLL

```
int DPLSelectedFont(int InstanceID);
```

Return values

0	No font has been selected
Non-zero	The ID of the selected font

SelectedImage

Image handling, Page layout

Description

Returns the ID of the selected image.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SelectedImage: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SelectedImage As Long
```

DLL

```
int DPLSelectedImage(int InstanceID);
```

Return values

0	No image has been selected
Non-zero	The ID of the selected image

SelectedPage

Page layout, Page manipulation

Description

Returns currently selected page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SelectedPage: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SelectedPage As Long
```

DLL

```
int DPLSelectedPage(int InstanceID);
```

SetActionURL

Version history

This function was introduced in Quick PDF Library version 7.21.

Description

Sets the target URL of the specified action.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetActionURL(ActionID: Integer;
  NewURL: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetActionURL(ActionID As Long,
  NewURL As String) As Long
```

DLL

```
int DPLSetActionURL(int InstanceID, int ActionID, wchar_t * NewURL);
```

Parameters

ActionID	An ActionID as returned by the **GetAnnotActionID**, **GetOutlineActionID** or **GetFormFieldActionID** functions
NewURL	The new URL target

Return values

0	The specified ActionID was not valid
1	The action's target URL was set successfully

SetAnnotBorderColor

Color, Annotations and hotspot links

Version history

This function was introduced in Quick PDF Library version 7.19.

Description

Sets the border color for the specified annotation.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetAnnotBorderColor(Index: Integer; Red,
  Green, Blue: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetAnnotBorderColor(
  Index As Long, Red As Double, Green As Double,
  Blue As Double) As Long
```

DLL

```
int DPLSetAnnotBorderColor(int InstanceID, int Index, double Red,
  double Green, double Blue);
```

Parameters

Index	The index of the annotation. The first annotation on the page has an index of 1.
Red	The red component of the color
Green	The green component of the color
Blue	The blue component of the color

SetAnnotBorderStyle

Annotations and hotspot links

Version history

This function was introduced in Quick PDF Library version 7.24.

Description

Sets the border style of the specified annotation.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetAnnotBorderStyle(Index: Integer;
  Width: Double; Style: Integer; DashOn, DashOff: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetAnnotBorderStyle(
  Index As Long, Width As Double, Style As Long,
  DashOn As Double, DashOff As Double) As Long
```

DLL

```
int DPLSetAnnotBorderStyle(int InstanceID, int Index, double Width,
  int Style, double DashOn, double DashOff);
```

Parameters

Index	The index of the annotation. The first annotation on the page has an index of 1.
Width	The width of the border
Style	The style of the border: 0 = Solid 1 = Dashed 2 = Beveled 3 = Inset Anything else = Solid
DashOn	The length of the dash. Only valid if the border style is "dashed".
DashOff	The length of the spaces betwen the dashes. Only valid if the border style is "dashed".

SetAnnotContents

Description

Changes the contents of an annotation.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetAnnotContents(Index: Integer;
  NewContents: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetAnnotContents(Index As Long,
  NewContents As String) As Long
```

DLL

```
int DPLSetAnnotContents(int InstanceID, int Index, wchar_t * NewContents);
```

Parameters

Index	The index of the annotation. The first annotation on the page has an index of 1.
NewContents	The new contents of the annotation

SetAnnotDblProperty

Version history

This function was introduced in Quick PDF Library version 9.11.

Description

Sets an double property of the specified annotation.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetAnnotDblProperty(Index, Tag: Integer;
  NewValue: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetAnnotDblProperty(
  Index As Long, Tag As Long, NewValue As Double) As Long
```

DLL

```
int DPLSetAnnotDblProperty(int InstanceID, int Index, int Tag,
  double NewValue);
```

Parameters

Index	The index of the annotation. The first annotation on the page has an index of 1.
Tag	105 = Left 106 = Top 107 = Width 108 = Height
NewValue	The new value of the specified annotation and property.

Return values

0	The annotation specified by the Index parameter was out of range or the Tag parameter was not valid
1	The annotation property was set successfully

SetAnnotIntProperty

Version history

This function was introduced in Quick PDF Library version 8.16.

Description

Sets an integer property of the specified annotation.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetAnnotIntProperty(Index, Tag,
  NewValue: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetAnnotIntProperty(
  Index As Long, Tag As Long, NewValue As Long) As Long
```

DLL

```
int DPLSetAnnotIntProperty(int InstanceID, int Index, int Tag,
  int NewValue);
```

Parameters

Index	The index of the annotation. The first annotation on the page has an index of 1.
Tag	116 = Page number of "GoToR" action (1 is first page) 131 = Page number of "GoTo" action
NewValue	The new value of the specified annotation and property.

Return values

0	The annotation specified by the Index parameter was out of range or the Tag parameter was not valid
1	The annotation property was set successfully

SetAnnotOptional

Version history

This function was introduced in Quick PDF Library version 11.14.

Description

Adds an annotation to the specified optional content group. This allows the annotation to be made visible or hidden using the layers functionality within the PDF viewer.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetAnnotOptional(Index,
  OptionalContentGroupID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetAnnotOptional(Index As Long,
  OptionalContentGroupID As Long) As Long
```

DLL

```
int DPLSetAnnotOptional(int InstanceID, int Index,
  int OptionalContentGroupID);
```

Parameters

Index	The index of the annotation. The first annotation on the page has an index of 1.
OptionalContentGroupID	An ID returned by the **NewOptionalContentGroup**, **GetOptionalContentGroupID** or **GetOptionalContentConfigOrderItemID** functions

Return values

0	The annotation could not be added to the specified optional content group
1	Success

SetAnnotQuadPoints

Annotations and hotspot links

Version history

This function was introduced in Quick PDF Library version 7.21.

Description

Sets the co-ordinates of the specified quad (rectangular area) contained within the specified annotation. If the QuadNumber is higher than the number of quads that the annotation already has then a new quad will be added to the annotation.

From version 7.25 the order of the co-ordinates has changed for consistency between **GetPageText** and **GetAnnotQuadPoints**.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetAnnotQuadPoints(Index,
  QuadNumber: Integer; X1, Y1, X2, Y2, X3, Y3, X4, Y4: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetAnnotQuadPoints(
  Index As Long, QuadNumber As Long, X1 As Double, Y1 As Double,
  X2 As Double, Y2 As Double, X3 As Double, Y3 As Double,
  X4 As Double, Y4 As Double) As Long
```

DLL

```
int DPLSetAnnotQuadPoints(int InstanceID, int Index, int QuadNumber,
  double X1, double Y1, double X2, double Y2, double X3,
  double Y3, double X4, double Y4);
```

Parameters

Index	The index of the annotation. The first annotation on the page has an index of 1.
QuadNumber	The index of the annotation's quad to set. The first quad has a QuadNumber of 1. If QuadNumber is greater than the number of existing quads then a new quad will be added to the annotation.
X1	The horizontal co-ordinate of the bottom-left corner.
Y1	The vertical co-ordinate of the bottom-left corner.
X2	The horizontal co-ordinate of the bottom-right corner.
Y2	The vertical co-ordinate of the bottom-right corner.
X3	The horizontal co-ordinate of the top-right corner.
Y3	The vertical co-ordinate of the top-right corner.
X4	The horizontal co-ordinate of the top-left corner.
Y4	The vertical co-ordinate of the top-left corner.

Return values

0	The QuadNumber parameter was less than 1.
1	The quad was changed or a new quad was added.

SetAnnotRect

Annotations and hotspot links

Version history

This function was introduced in Quick PDF Library version 9.11.

Description

Sets the size and position of the specified annotation.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetAnnotRect(Index: Integer; Left, Top,
  Width, Height: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetAnnotRect(Index As Long,
  Left As Double, Top As Double, Width As Double,
  Height As Double) As Long
```

DLL

```
int DPLSetAnnotRect(int InstanceID, int Index, double Left, double Top,
  double Width, double Height);
```

Parameters

Index	The index of the annotation. The first annotation on the page has an index of 1.
Left	The new horizontal co-ordinate of the left edge of the annotation
Top	The new vertical co-ordinate of the top edge of the annotation
Width	The new width of the annotation
Height	The new height of the annotation

SetAnnotStrProperty

Description

Sets a string property of the specified annotation.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetAnnotStrProperty(Index, Tag: Integer;
  NewValue: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetAnnotStrProperty(
  Index As Long, Tag As Long, NewValue As String) As Long
```

DLL

```
int DPLSetAnnotStrProperty(int InstanceID, int Index, int Tag,
  wchar_t * NewValue);
```

Parameters

Index	The index of the annotation. The first annotation on the page has an index of 1.
Tag	102 = Contents 103 = Name 110 = Subject 111 = URL of a link annotation 113 = The "Win" file name of a "Launch" action 114 = The "F" file name of a "Launch" action 115 = The "F" file name of a "GoToR" action 127 = Subject 129 = The "UF" file name of a "Launch" action 130 = The "UF" file name of a "GoToR" action
NewValue	The new value of the specified annotation and property.

Return values

0	The annotation specified by the Index parameter was out of range or the Tag parameter was not valid
1	The annotation property was set successfully

SetAnsiMode

Miscellaneous functions

Version history

This function was introduced in Quick PDF Library version 8.12.

Description

This function sets the mode used by the DLL to convert strings to and from Unicode.

Syntax

```
int DPLSetAnsiMode(int InstanceID, int NewAnsiMode);
```

Parameters

NewAnsiMode	0 = Conversion using the current code page
	1 = Conversion using UTF-8 encoding

SetAppendInputFromString

Version history

This function was introduced in Quick PDF Library version 11.11.

Description

Sets the input for a subsequent call to the **AppendToString** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetAppendInputFromString(
  const Source: AnsiString): Integer;
```

DLL

```
int DPLSetAppendInputFromString(int InstanceID, char * Source);
```

Parameters

Source	The input PDF to base the update section on

Return values

0	Could not set input
1	Input set successfully

SetAppendInputFromVariant

Version history

This function was introduced in Quick PDF Library version 11.11.

Description

Sets the input for a subsequent call to the **SetAppendToVariant** function.

Syntax

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetAppendInputFromVariant(
  Source As Variant) As Long
```

Parameters

Source	A byte array variant containing the input PDF to base the update section on

Return values

0	Could not set input
1	Input set successfully

SetBaseURL

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Version history

This function was introduced in Quick PDF Library version 7.15.

Description

Sets the Base URL for all URL links in the document.

For example, if the Base URL was set to "http://www.example.com/" and a URL link destination was set to "index.html" then the link will point to "http://www.example.com/index.html".

Use the **AddLinkToWeb** function to add a URL link to the current page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetBaseURL(NewBaseURL: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetBaseURL(
  NewBaseURL As String) As Long
```

DLL

```
int DPLSetBaseURL(int InstanceID, wchar_t * NewBaseURL);
```

Parameters

NewBaseURL	The base URL to use for all URL link annotations in the document.

SetBlendMode

Description

Sets the blend mode for subsequently drawn graphics.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetBlendMode(BlendMode: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetBlendMode(
    BlendMode As Long) As Long
```

DLL

```
int DPLSetBlendMode(int InstanceID, int BlendMode);
```

Parameters

BlendMode	The blend mode to use: 0 = Normal 1 = Multiply 2 = Screen 3 = Overlay 4 = Darken 5 = Lighten 6 = Color Dodge 7 = Color Burn 9 = Hard Light 10 = Soft Light 11 = Difference 12 = Exclusion 13 = Hue 14 = Saturation 14 = Color 15 = Luminosity

SetBreakString

Description

Sets the string to use to mark line breaks. This string allows text to be split when using the *WrappedText functions. The breakstring by default is set to CR/LF. ie. '#13#10' in Delphi.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetBreakString(
  NewBreakString: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetBreakString(
  NewBreakString As String) As Long
```

DLL

```
int DPLSetBreakString(int InstanceID, wchar_t * NewBreakString);
```

Parameters

NewBreakString	The string of characters to use as a break character, for example Chr(13) + Chr(10)

SetCSDictEPSG

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Sets the EPSG reference code for a coordinate system dictionary (see www.epsg.org).

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetCSDictEPSG(CSDictID,
  NewEPSG: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetCSDictEPSG(CSDictID As Long,
  NewEPSG As Long) As Long
```

DLL

```
int DPLSetCSDictEPSG(int InstanceID, int CSDictID, int NewEPSG);
```

Parameters

CSDictID	A value returned from the **GetMeasureDictGCSDict** or **GetMeasureDictDCSDict** functions
NewEPSG	The new value for the EPSG reference code

Return values

0	The CSDictID parameter was incorrect
1	Success

SetCSDictType

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Sets the coordinate system type of a coordinate system dictionary.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetCSDictType(CSDictID,
  NewDictType: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetCSDictType(CSDictID As Long,
  NewDictType As Long) As Long
```

DLL

```
int DPLSetCSDictType(int InstanceID, int CSDictID, int NewDictType);
```

Parameters

CSDictID	A value returned from the **GetMeasureDictGCSDict** or **GetMeasureDictDCSDict** functions
NewDictType	1 = Geographic coordinate system (GEOGCS) 2 = Projected coordinate system (PROJCS)

Return values

0	The CSDictID parameter was incorrect or the NewDictType parameter was out of range
1	Success

SetCSDictWKT

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Sets the Well Known Text (WKT) describing a coordinate system dictionary.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetCSDictWKT(CSDictID: Integer;
  NewWKT: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetCSDictWKT(CSDictID As Long,
  NewWKT As String) As Long
```

DLL

```
int DPLSetCSDictWKT(int InstanceID, int CSDictID, wchar_t * NewWKT);
```

Parameters

CSDictID	A value returned from the **GetMeasureDictGCSDict** or **GetMeasureDictDCSDict** functions
NewWKT	The new Well Known Text description

Return values

0	The CSDictID parameter was incorrect
1	Success

SetCairoFileName

Version history

This function was introduced in Quick PDF Library version 8.13.

Description

Sets the path and file name of the Cairo DLL. The **SelectRenderer** function can be used to select the Cairo renderer rather than the default GDI+ renderer.

The Cairo DLL is usually dependent on other DLLs. If these are not all stored in the same directory as the application, or a system directory, the Windows API function SetDllDirectory should be used to add the correct path before calling any rendering functions.

Rendering using Cairo is currently experimental.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetCairoFileName(
  FileName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetCairoFileName(
  FileName As String) As Long
```

DLL

```
int DPLSetCairoFileName(int InstanceID, wchar_t * FileName);
```

Parameters

FileName	The path and file name of the Cairo DLL.

Return values

0	The specified DLL was not a valid Cairo DLL
1	The specified Cairo DLL was valid

SetCapturedPageOptional

Content Streams and Optional Content Groups, Page layout

Description

Links the captured page to an optional content group. This allows the captured page to be selectively shown in Acrobat 6 or later.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetCapturedPageOptional(CaptureID,
  OptionalContentGroupID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetCapturedPageOptional(
  CaptureID As Long, OptionalContentGroupID As Long) As Long
```

DLL

```
int DPLSetCapturedPageOptional(int InstanceID, int CaptureID,
  int OptionalContentGroupID);
```

Parameters

CaptureID	The ID returned by the **CapturePage** function when a page was previously captured
OptionalContentGroupID	An ID returned by the **NewOptionalContentGroup**, **GetOptionalContentGroupID** or **GetOptionalContentConfigOrderItemID** functions

Return values

0	The CaptureID or OptionalContentGroupID parameters were not valid
1	The captured page was linked to the optional content group successfully

SetCapturedPageTransparencyGroup

Content Streams and Optional Content Groups, Page layout

Version history

This function was introduced in Quick PDF Library version 8.14.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetCapturedPageTransparencyGroup(CaptureID,
  CS, Isolate, Knockout: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetCapturedPageTransparencyGroup(
  CaptureID As Long, CS As Long, Isolate As Long,
  Knockout As Long) As Long
```

DLL

```
int DPLSetCapturedPageTransparencyGroup(int InstanceID, int CaptureID,
  int CS, int Isolate, int Knockout);
```

Parameters

CaptureID	The ID returned by the **CapturePage** function when a page was previously captured
CS	The color space to use: 1 = RGB 2 = CMYK
Isolate	This parameter has no effect and is reserved for future use. It should always be set to 0.
Knockout	Indicates whether items added to the page are drawn over each other or "knocked out" of the page. In knockout mode a "hole" is made through existing objects on the page in the shape of the new object. The new object is then drawn against the background. 0 = Do not knockout 1 = Knockout

Return values

0	An error occurred
1	Success

SetCatalogInformation

Description

This function allows you to store custom information in the PDF document. This is similar to the **SetCustomInformation** function, but the information is stored in the Document Catalog instead of the Document Information Dictionary. Metadata should be stored in the Document Information Dictionary using **SetCustomInformation**, private content or structural information should be stored in the Document Catalog using this fuction.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetCatalogInformation(Key,
  NewValue: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetCatalogInformation(
  Key As String, NewValue As String) As Long
```

DLL

```
int DPLSetCatalogInformation(int InstanceID, wchar_t * Key,
  wchar_t * NewValue);
```

Parameters

Key	The name of the key to set. This key must have a special prefix assigned to you by Adobe to avoid conflicts with other software.
NewValue	The new value of the specified key.

Return values

0	The key specified could not be set, it may have been a system key
1	The value of the specified key was set successfully

SetCharWidth

Version history

This function was introduced in Quick PDF Library version 7.11.

Description

Sets the width of a specific character in the selected font.

The width uses is a ratio to the text size. For example, if a value of 750 is used the width of the character when output as 12pt text would be (750 / 1000) * 12.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetCharWidth(CharCode,
  NewWidth: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetCharWidth(CharCode As Long,
  NewWidth As Long) As Long
```

DLL

```
int DPLSetCharWidth(int InstanceID, int CharCode, int NewWidth);
```

Parameters

CharCode	The glyph character code that should be set. For example, 65 for "A".
NewWidth	The new width

Return values

0	A font has not been selected
1	The width was set successfully

SetClippingPath

Description

Uses the current path as a clipping path for subsequent drawing operations.

The current path is combined with the existing clipping path, this means that the clipping area can only be made smaller.

To restore the clipping path, call **SaveState** before calling this function and then **LoadState** to restore the clipping path to its previous state.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetClippingPath: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetClippingPath As Long
```

DLL

```
int DPLSetClippingPath(int InstanceID);
```

SetClippingPathEvenOdd

Description

Similar to the **SetClippingPath** function, but uses the "even odd" method for dealing with situations where parts of the path overlap.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetClippingPathEvenOdd: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetClippingPathEvenOdd As Long
```

DLL

```
int DPLSetClippingPathEvenOdd(int InstanceID);
```

SetCompatibility

Version history

This function was introduced in Quick PDF Library version 7.19.

Description

Sets Quick PDF Library to operate in the same way as previous versions of the library to maintain backwards compatibility.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetCompatibility(CompatibilityItem,
  CompatibilityMode: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetCompatibility(
  CompatibilityItem As Long, CompatibilityMode As Long) As Long
```

DLL

```
int DPLSetCompatibility(int InstanceID, int CompatibilityItem,
  int CompatibilityMode);
```

Parameters

CompatibilityItem	100 = **DrawTableRows** return value scaling (version 7.18)
CompatibilityMode	0 = Turn off compatibility 1 = Turn on compatibility

Return values

0	Either CompatibilityItem or CompatibilityMode was out of range
1	The compatibility mode was set successfully

SetContentStreamFromString

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Sets the PDF page description commands in the content stream part that was selected with the **SelectContentStream** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetContentStreamFromString(
  const Source: AnsiString): Integer;
```

DLL

```
int DPLSetContentStreamFromString(int InstanceID, char * Source);
```

Parameters

Source	The new PDF page description commands for the content stream part

SetContentStreamFromVariant

Page properties, Content Streams and Optional Content Groups, Page manipulation

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Sets the PDF page description commands in the content stream part that was selected with the **SelectContentStream** function.

Syntax

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetContentStreamFromVariant(
  NewValue As Variant) As Long
```

Parameters

NewValue   A variant byte array containing the new PDF page description commands for the content stream part

SetContentStreamOptional

Content Streams and Optional Content Groups

Version history

This function was renamed in Quick PDF Library version 8.11.
The function name in earlier versions was SetLayerOptional.

Description

A page in a PDF document has one or more content stream parts that together contain all the PDF page description commands for the page.

This function links the content stream that was selected using the **SelectContentStream** function to an optional content group. This allows the content stream part to be selectively shown in Acrobat 6 or later.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetContentStreamOptional(
  OptionalContentGroupID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetContentStreamOptional(
  OptionalContentGroupID As Long) As Long
```

DLL

```
int DPLSetContentStreamOptional(int InstanceID,
  int OptionalContentGroupID);
```

Parameters

OptionalContentGroupID	An ID returned by the **NewOptionalContentGroup**, **GetOptionalContentGroupID** or **GetOptionalContentConfigOrderItemID** functions

Return values

0	The OptionalContentGroupID parameter was not valid
1	The content stream part was linked to the optional content group successfully

SetCropBox

Description

Sets the visible area of the selected page. The non-visible area will be "cropped" and will not be displayed or printed.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetCropBox(Left, Top, Width,
  Height: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetCropBox(Left As Double,
  Top As Double, Width As Double, Height As Double) As Long
```

DLL

```
int DPLSetCropBox(int InstanceID, double Left, double Top, double Width,
  double Height);
```

Parameters

Left	The horizontal co-ordinate of the left edge of the cropping rectangle
Top	The vertical co-ordinate of the top edge of the cropping rectangle
Width	The width of the cropping rectangle
Height	The height of the cropping rectangle

SetCustomInformation

Description

This function is used to store custom metadata in the document. These values can later be read from the document with the **GetCustomInformation** function. The data is stored in the Document Information Dictionary. Private content or structural information should rather be stored in the Document Catalog using the **SetCatalogInformation** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetCustomInformation(Key,
  NewValue: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetCustomInformation(
  Key As String, NewValue As String) As Long
```

DLL

```
int DPLSetCustomInformation(int InstanceID, wchar_t * Key,
  wchar_t * NewValue);
```

Parameters

Key	Specifies which key to set
NewValue	The value to set the key to.

Return values

0	The value could not be set. The Key parameter cannot be "Producer", "Creator", "Subject", "Title", "Keywords" or "Author". For these keys use the **SetInformation** function.
1	The value of the key was set successfully

SetCustomLineDash

Description

Sets a custom line dash pattern.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetCustomLineDash(DashPattern: WideString;
  DashPhase: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetCustomLineDash(
  DashPattern As String, DashPhase As Double) As Long
```

DLL

```
int DPLSetCustomLineDash(int InstanceID, wchar_t * DashPattern,
  double DashPhase);
```

Parameters

DashPattern	A list of numeric values separated with commas. Alternate values are used for dashes and spaces. A period must be used for numbers with decimal fractions. For example, to make a dash-dot-dot pattern the following could be used: "20.5,10,11,10,11,10"
DashPhase	The distance within the pattern to start the dashed line. For example, if DashPattern is "20,10,40,10" and DashPhase is set to 5, the dashed line will start with a dash of size 15. The next dash will be 40, then 20, then 40, etc. with spaces of 10 between each dash.

Return values

0	The dash pattern was not valid
1	The custom dash pattern was set successfully

SetDPLRFileName

Version history

This function was introduced in Quick PDF Library version 10.12.

Description

Sets the path and file name of the DPLR DLL. The **SelectRenderer** function can be used to select the DPLR renderer rather than the default GDI+ renderer.

The required DPLR rendering DLL for this function can be found in the 'Rendering and Printing Add-On' folder/ directory of the Debenu Quick PDF Library installation folder. Please see the README.TXT in this folder for futher explanation if required.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetDPLRFileName(
  FileName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetDPLRFileName(
  FileName As String) As Long
```

DLL

```
int DPLSetDPLRFileName(int InstanceID, wchar_t * FileName);
```

Parameters

FileName	The path and file name of the DPLR DLL.

Return values

0	The specified DLL was not a valid DPLR DLL
1	The specified DPLR DLL was valid

SetDecodeMode

Version history

This function was introduced in Quick PDF Library version 11.11.

Description

This function provides a way to select between different object decoding modes.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetDecodeMode(
  NewDecodeMode: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetDecodeMode(
  NewDecodeMode As Long) As Long
```

DLL

```
int DPLSetDecodeMode(int InstanceID, int NewDecodeMode);
```

Parameters

NewDecodeMode	1=Older method 2=Default method

Return values

0	The NewDecodeMode parameter was invalid
1	The decode mode was set successfully

SetDestProperties

Version history

This function was introduced in Quick PDF Library version 8.14.

Description

Changes various properties of an existing destination.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetDestProperties(DestID, Zoom,
  DestType: Integer; Left, Top, Right, Bottom: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetDestProperties(
  DestID As Long, Zoom As Long, DestType As Long,
  Left As Double, Top As Double, Right As Double,
  Bottom As Double) As Long
```

DLL

```
int DPLSetDestProperties(int InstanceID, int DestID, int Zoom,
  int DestType, double Left, double Top, double Right,
  double Bottom);
```

Parameters

DestID	The ID of the destination to analyse. A valid destination ID is returned by the **GetOutlineDest** function.
Zoom	The zoom percentage to use when the outline destination is opened, valid values from 0 to 6400. Only used for DestType = 1, should be set to 0 for other DestTypes.
DestType	1 = "XYZ" - the target page is positioned at the point specified by the Left and Top parameters. The Zoom parameter specifies the zoom percentage. 2 = "Fit" - the entire page is zoomed to fit the window. None of the other parameters are used and should be set to zero. 3 = "FitH" - the page is zoomed so that the entire width of the page is visible. The height of the page may be greater or less than the height of the window. The page is positioned at the vertical position specified by the Top parameter. 4 = "FitV" - the page is zoomed so that the entire height of the page can be seen. The width of the page may be greater or less than the width of the window. The page is positioned at the horizontal position specified by the Left parameter. 5 = "FitR" - the page is zoomed so that a certain rectangle on the page is visible. The Left, Top, Right and Bottom parameters define the rectangular area on the page. 6 = "FitB" - the page is zoomed so that it's bounding box is visible. 7 = "FitBH" - the page is positioned vertically at the position specified by the Top parameter. The page is zoomed so that the entire width of the page's bounding box is visible. 8 = "FitBV" - the page is positioned at the horizontal position specified by the Left parameter. The page is zoomed just enough to fit the entire height of the bounding box into the window.
Left	The horizontal position used by DestType = 1, 4, 5 and 8
Top	The vertical position used by DestType = 1, 3, 5 and 7
Right	The horizontal position of the righthand edge of the rectangle. Used by DestType = 5
Bottom	The horizontal position of the bottom of the rectangle. Used by DestType = 5

Return values

0	The destination properties could not be set. The DestID parameter might be invalid or the Zoom and DestType parameters could be out of range.
1	The destination properties were set successfully

SetDestValue

Annotations and hotspot links

Version history

This function was introduced in Quick PDF Library version 8.14.

Description

Sets one of the properties of the specified destination.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetDestValue(DestID, ValueKey: Integer;
  NewValue: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetDestValue(DestID As Long,
  ValueKey As Long, NewValue As Double) As Long
```

DLL

```
int DPLSetDestValue(int InstanceID, int DestID, int ValueKey,
  double NewValue);
```

Parameters

DestID	The ID of the destination to analyse. A valid destination ID is returned by the **GetOutlineDest** function.
ValueKey	1=Left 2=Top 3=Bottom 4=Right 5=Zoom
NewValue	The new value for the specified destination property

Return values

0	The destination value could not be set. The DestID parameter might be invalid or the DestType parameter could be out of range.
1	The destination type was set successfully

SetDocumentMetadata

Document properties

Description

Set's the document metadata. The metadata must be a valid XMP string, see Adobe's website for XMP documentation.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetDocumentMetadata(
  XMP: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetDocumentMetadata(
  XMP As String) As Long
```

DLL
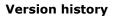
```
int DPLSetDocumentMetadata(int InstanceID, wchar_t * XMP);
```

Parameters

XMP   The XMP metadata

Return values

This function always returns 1

SetEmbeddedFileStrProperty

Version history

This function was introduced in Quick PDF Library version 7.19.

Description

Sets a property of the specified embedded file.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetEmbeddedFileStrProperty(Index,
  Tag: Integer; NewValue: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetEmbeddedFileStrProperty(
  Index As Long, Tag As Long, NewValue As String) As Long
```

DLL

```
int DPLSetEmbeddedFileStrProperty(int InstanceID, int Index, int Tag,
  wchar_t * NewValue);
```

Parameters

Index	The index of the embedded file. Must be a value between 1 and the value returned by **EmbeddedFileCount**.
Tag	1 = File name 2 = MIME type 3 = Creation date 4 = Modification date 5 = Title 7 = Description
NewValue	The new value of the specified property.

SetFillColor

Description

Sets the fill color for any subsequently drawn graphics. The values for Red, Green and Blue range from 0 to 1, where 0 indicates 0% and 1 indicates 100% of the color.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFillColor(Red, Green,
  Blue: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFillColor(Red As Double,
  Green As Double, Blue As Double) As Long
```

DLL

```
int DPLSetFillColor(int InstanceID, double Red, double Green, double Blue);
```

Parameters

Red	The red component of the color
Green	The green component of the color
Blue	The blue component of the color

SetFillColorCMYK

Description

Sets the fill color of subsequently drawn graphics. Similar to the **SetFillColor** function, but allows a color in the CMYK color space to be used. The values of the color parameters range from 0 to 1, with 0 indicating 0% and 1 indicating 100% of the color.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFillColorCMYK(C, M, Y,
  K: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFillColorCMYK(C As Double,
  M As Double, Y As Double, K As Double) As Long
```

DLL

```
int DPLSetFillColorCMYK(int InstanceID, double C, double M, double Y,
  double K);
```

Parameters

C	The cyan component of the color
M	The magenta component of the color
Y	The yellow component of the color
K	The black component of the color

SetFillColorSep

Description

Sets the fill color of subsequently drawn graphics. Similar to the **SetFillColor** function, but a tint of a separation color added with the **AddSeparationColor** function is used.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFillColorSep(ColorName: WideString;
  Tint: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFillColorSep(
  ColorName As String, Tint As Double) As Long
```

DLL

```
int DPLSetFillColorSep(int InstanceID, wchar_t * ColorName, double Tint);
```

Parameters

ColorName	The name of the separation color that was used with the **AddSeparationColor** function
Tint	The amount of color to use. 0 indicates no color (white), 1 indicates maximum color.

Return values

0	The separation color name could not be found
1	The fill color was set successfully

SetFillShader

Version history

This function was introduced in Quick PDF Library version 7.11.

Description

Sets the fill to the specified shader for subsequently drawn graphics.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFillShader(
  ShaderName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFillShader(
  ShaderName As String) As Long
```

DLL

```
int DPLSetFillShader(int InstanceID, wchar_t * ShaderName);
```

Parameters

ShaderName	The shader name that was used when the shader was created.

Return values

0	The shader could not be found
1	The shader fill was setup correctly

SetFillTilingPattern

Vector graphics, Color

Version history

This function was introduced in Quick PDF Library version 8.16.

Description

Sets the current fill to the specified tiling pattern.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFillTilingPattern(
  PatternName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFillTilingPattern(
  PatternName As String) As Long
```

DLL

```
int DPLSetFillTilingPattern(int InstanceID, wchar_t * PatternName);
```

Parameters

PatternName	The pattern name that was used with the **NewTilingPatternFromCapturedPage** function

Return values

0	The PatternName parameter was invalid
1	Success

SetFindImagesMode

Image handling, Document management, Page properties

Version history

This function was introduced in Quick PDF Library version 7.22.

Description

Sets the search mode used by the **FindImages** function.

The default search mode runs a recursive search in the resources of all the pages and annotations in the document. This is the fastest method and requires the least amount of memory, however unused images will not be found.

The full search mode examines each object in the document. This takes more time and requires more memory, however all images will be located even if they are not used by any of the pages or annotations in the document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFindImagesMode(
  NewFindImagesMode: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFindImagesMode(
  NewFindImagesMode As Long) As Long
```

DLL

```
int DPLSetFindImagesMode(int InstanceID, int NewFindImagesMode);
```

Parameters

NewFindImagesMode	1 = Default search mode 2 = Full search mode 3 = Default search mode, full convert 4 = Full search mode, full convert

Return values

0	An invalid value for the NewFindImagesMode parameter was used
1	The search mode was changed successfully

SetFontEncoding

Description

Sets the encoding for the selected font.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFontEncoding(Encoding: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFontEncoding(
  Encoding As Long) As Long
```

DLL

```
int DPLSetFontEncoding(int InstanceID, int Encoding);
```

Parameters

Encoding	The encoding to use for the font: 0 = StandardEncoding 1 = MacRomanEncoding 2 = WinAnsiEncoding 3 = Deprecated (was PDFDocEncoding) 4 = MacExpertEncoding 5 = Do not specify encoding

Return values

0	No font was selected, or the encoding could not be set
1	The encoding for the selected font was set successfully

SetFontFlags

Description

Sets the flags for the selected font. Usually these flags are set automatically when the font is added, but in some circumstance (for example with symbolic Type1 fonts) the flags cannot be automatically set. This function allows you to ensure the fonts have the correct flags.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFontFlags(Fixed, Serif, Symbolic,
    Script, Italic, AllCap, SmallCap, ForceBold: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFontFlags(Fixed As Long,
    Serif As Long, Symbolic As Long, Script As Long,
    Italic As Long, AllCap As Long, SmallCap As Long,
    ForceBold As Long) As Long
```

DLL

```
int DPLSetFontFlags(int InstanceID, int Fixed, int Serif, int Symbolic,
    int Script, int Italic, int AllCap, int SmallCap,
    int ForceBold);
```

Parameters

Fixed	0 = Font is proportional or variable width 1 = Font is fixed width, all glyphs have the same width
Serif	0 = Glyphs do not have serifs (short strokes drawn at an angle on the top and bottom of glyph stems) 1 = Glyphs have serifs
Symbolic	0 = Font contains glyphs in the standard Latin character set 1 = Font contains symbols
Script	0 = Font contains regular glyphs 1 = Glyphs resemble cursive handwriting
Italic	0 = Regular font 1 = Glyphs have dominant vertical strokes that are slanted
AllCap	0 = Font contains lowercase letters 1 = Font contains only uppercase letters
SmallCap	0 = Regular font 1 = Lowercase glyphs look like the corresponding uppercase glyphs but are smaller in size
ForceBold	0 = Regular font 1 = Force font to be rendered with a bold effect even at small sizes

Return values

0	A font has not been selected
1	The font flags were set successfully

SetFormFieldAlignment

Description

Sets the alignment for the specified form field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldAlignment(Index,
  Alignment: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldAlignment(
  Index As Long, Alignment As Long) As Long
```

DLL

```
int DPLSetFormFieldAlignment(int InstanceID, int Index, int Alignment);
```

Parameters

Index	The index of the form field to work with. The first form field has an index of 1.
Alignment	The alignment to use for the form field: 0 = Left alignment 1 = Centered 2 = Right alignment

Return values

0	The form field index was invalid
1	The alignment of the form field was set successfully

SetFormFieldAnnotFlags

Form fields

Description

Set the "annotation" flags for the specified form field. This is for advanced use, see the PDF specification for details.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldAnnotFlags(Index,
  NewFlags: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldAnnotFlags(
  Index As Long, NewFlags As Long) As Long
```

DLL

```
int DPLSetFormFieldAnnotFlags(int InstanceID, int Index, int NewFlags);
```

Parameters

Index	The index of the form field to change
NewFlags	The new flags value to apply

Return values

0	The specified form field could not be found
1	The "annotation" flags for the specified form field were set successfully

SetFormFieldBackgroundColor

Form fields, Color

Description

Sets the background color of the specified form field. The values of the color parameters range from 0 to 1, with 0 indicating 0% and 1 indicating 100% of the color.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldBackgroundColor(Index: Integer;
  Red, Green, Blue: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldBackgroundColor(
  Index As Long, Red As Double, Green As Double,
  Blue As Double) As Long
```

DLL

```
int DPLSetFormFieldBackgroundColor(int InstanceID, int Index, double Red,
  double Green, double Blue);
```

Parameters

Index	The index of the form field
Red	The red component of the color
Green	The green component of the color
Blue	The blue component of the color

Return values

0	The form field could not be found or the parameters were invalid.
1	The background color of the form field was set successfully

SetFormFieldBackgroundColorCMYK

Form fields, Color

Description

Sets the background color of the specified form field. Similar to the **SetFormFieldBorderColor** function, but the color components are specified in the CMYK color space (Cyan, Magenta, Yellow, Black). The values of the color parameters range from 0 to 1, with 0 indicating 0% and 1 indicating 100% of the color.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldBackgroundColorCMYK(
    Index: Integer; C, M, Y, K: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldBackgroundColorCMYK(
    Index As Long, C As Double, M As Double, Y As Double,
    K As Double) As Long
```

DLL

```
int DPLSetFormFieldBackgroundColorCMYK(int InstanceID, int Index,
    double C, double M, double Y, double K);
```

Parameters

Index	The index of the form field
C	The cyan component of the color
M	The magenta component of the color
Y	The yellow component of the color
K	The black component of the color

Return values

0	The form field could not be found
1	The background color of the specified form field was set successfully

SetFormFieldBackgroundColorGray

Form fields, Color

Version history

This function was introduced in Quick PDF Library version 9.12.

Description

Sets the background color of the specified form field. Similar to the
SetFormFieldBackgroundColor function, but a single color component is specified in the Gray
color space. Possible values are in the range 0 to 1.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldBackgroundColorGray(
  Index: Integer; Gray: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldBackgroundColorGray(
  Index As Long, Gray As Double) As Long
```

DLL

```
int DPLSetFormFieldBackgroundColorGray(int InstanceID, int Index,
  double Gray);
```

Parameters

Index	The index of the form field
Gray	The gray component

Return values

0	The form field could not be found
1	The background color of the specified form field was set successfully

SetFormFieldBackgroundColorSep

Description

Sets the background color of the specified form field. Similar to the **SetFormFieldBorderColor** function, but a tint of a separation color added with the **AddSeparationColor** function is used. The PDF specification does not support separation color spaces for form fields, so the results may not always work, especially if the form field is later edited in Acrobat. This feature has been added for situations where the form field will be flattened.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldBackgroundColorSep(
  Index: Integer; ColorName: WideString; Tint: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldBackgroundColorSep(
  Index As Long, ColorName As String, Tint As Double) As Long
```

DLL

```
int DPLSetFormFieldBackgroundColorSep(int InstanceID, int Index,
  wchar_t * ColorName, double Tint);
```

Parameters

Index	The index of the form field
ColorName	The name of the separation color that was used with the **AddSeparationColor** function
Tint	The amount of color to use. 0 indicates no color (white), 1 indicates maximum color.

Return values

0	The form field could not be found, or the separation color name could not be found
1	The background color of the specified form field was set successfully

SetFormFieldBorderColor

Description

Sets the border color of the specified form field. The values of the color parameters range from 0 to 1, with 0 indicating 0% and 1 indicating 100% of the color.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldBorderColor(Index: Integer;
  Red, Green, Blue: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldBorderColor(
  Index As Long, Red As Double, Green As Double,
  Blue As Double) As Long
```

DLL

```
int DPLSetFormFieldBorderColor(int InstanceID, int Index, double Red,
  double Green, double Blue);
```

Parameters

Index	The index of the form field
Red	The red component of the color
Green	The green component of the color
Blue	The blue component of the color

Return values

0	The form field could not be found or the parameters were invalid
1	The border color of the form field was set successfully

SetFormFieldBorderColorCMYK

Description

Sets the border color of the specified form field. Similar to the **SetFormFieldBorderColor** function, but the color components are specified in the CMYK color space (Cyan, Magenta, Yellow, Black). The values of the color parameters range from 0 to 1, with 0 indicating 0% and 1 indicating 100% of the color.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldBorderColorCMYK(Index: Integer;
  C, M, Y, K: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldBorderColorCMYK(
  Index As Long, C As Double, M As Double, Y As Double,
  K As Double) As Long
```

DLL

```
int DPLSetFormFieldBorderColorCMYK(int InstanceID, int Index, double C,
  double M, double Y, double K);
```

Parameters

Index	The index of the form field
C	The amount of cyan for the color. 0 indicates no cyan, 1 indicates maximum cyan.
M	The amount of magenta for the color. equivalent to the separation color. 0 indicates no magenta, 1 indicates maximum magenta.
Y	The amount of yellow for the color. 0 indicates no yellow, 1 indicates maximum yellow.
K	The amount of black for the color. 0 indicates no black, 1 indicates maximum black.

Return values

0	The form field could not be found
1	The border color of the specified form field was set successfully

SetFormFieldBorderColorGray

Version history

This function was introduced in Quick PDF Library version 9.12.

Description

Sets the border color of the specified form field. Similar to the **SetFormFieldBorderColor** function, but a single color component is specified in the Gray color space. Possible values are in the range 0 to 1.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldBorderColorGray(Index: Integer;
  Gray: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldBorderColorGray(
  Index As Long, Gray As Double) As Long
```

DLL

```
int DPLSetFormFieldBorderColorGray(int InstanceID, int Index, double Gray);
```

Parameters

Index	The index of the form field
Gray	The gray component

Return values

0	The form field could not be found
1	The background color of the specified form field was set successfully

SetFormFieldBorderColorSep

Form fields, Color

Description

Sets the border color of the specified form field. Similar to the **SetFormFieldBorderColor** function, but a tint of a separation color added with the **AddSeparationColor** function is used. The PDF specification does not support separation color spaces for form fields, so the results may not always work, especially if the form field is later edited in Acrobat. This feature has been added for situations where the form field will be flattened.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldBorderColorSep(Index: Integer;
  ColorName: WideString; Tint: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldBorderColorSep(
  Index As Long, ColorName As String, Tint As Double) As Long
```

DLL

```
int DPLSetFormFieldBorderColorSep(int InstanceID, int Index,
  wchar_t * ColorName, double Tint);
```

Parameters

Index	The index of the form field
ColorName	The name of the separation color that was used with the **AddSeparationColor** function
Tint	The amount of color to use. 0 indicates no color (white), 1 indicates maximum color.

Return values

0	The form field could not be found, or the separation color name could not be found
1	The border color of the specified form field was set successfully

SetFormFieldBorderStyle

Form fields

Description

Sets the width and line style of the specified form field's border.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldBorderStyle(Index: Integer;
  Width: Double; Style: Integer; DashOn, DashOff: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldBorderStyle(
  Index As Long, Width As Double, Style As Long,
  DashOn As Double, DashOff As Double) As Long
```

DLL

```
int DPLSetFormFieldBorderStyle(int InstanceID, int Index, double Width,
  int Style, double DashOn, double DashOff);
```

Parameters

Index	The index of the form field
Width	The width of the border
Style	The style of the border: 0 = Solid 1 = Dashed 2 = Beveled 3 = Inset Anything else = Solid
DashOn	The length of the dash. Only valid if the border style is "dashed".
DashOff	The length of the space between dashes. Only valid if the border style is "dashed".

Return values

0	The form field could not be found or the parameters were invalid
1	The border style of the form field was set successfully

SetFormFieldBounds

Form fields

Description

Changes the physical size and position of the specified form field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldBounds(Index: Integer; Left,
  Top, Width, Height: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldBounds(
  Index As Long, Left As Double, Top As Double, Width As Double,
  Height As Double) As Long
```

DLL

```
int DPLSetFormFieldBounds(int InstanceID, int Index, double Left,
  double Top, double Width, double Height);
```

Parameters

Index	The index of the form field to adjust
Left	The new co-ordinate of the left edge of the form field
Top	The new co-ordinate of the top of the form field
Width	The new width of the form field
Height	The new height of the form field

Return values

0	The form field could not be found
1	The form field was resized and moved successfully

SetFormFieldCalcOrder

Description

Sets or changes the calculation order for form fields.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldCalcOrder(Index,
  Order: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldCalcOrder(
  Index As Long, Order As Long) As Long
```

DLL

```
int DPLSetFormFieldCalcOrder(int InstanceID, int Index, int Order);
```

Parameters

Index	The index of the form field to add to the list of calculated field
Order	The order this field should be calculated in. A value of 0 means this field is the first field to be calculated. A value of 1 means this field is the second field to be calculated. Use a value of -1 to specify this field should be calculated last out of the fields which have already been added to the calculation order list.

Return values

0	The specified form field could not be found
1	The specified form field was added to the calculation order list, or moved to the new position if it was already in the list

SetFormFieldCaption

Description

Sets the caption of the form field. This applies to buttons, checkboxes and radiobutton form fields only.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldCaption(Index: Integer;
  NewCaption: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldCaption(
  Index As Long, NewCaption As String) As Long
```

DLL

```
int DPLSetFormFieldCaption(int InstanceID, int Index,
  wchar_t * NewCaption);
```

Parameters

Index	The index of the form field
NewCaption	The new caption for the form field.

Return values

0	The form field could not be found or the parameters were invalid
1	The caption of the form field was set successfully

SetFormFieldCheckStyle

Description

Sets the check style for checkbox fields or radio-button sub-fields.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldCheckStyle(Index, CheckStyle,
  Position: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldCheckStyle(
  Index As Long, CheckStyle As Long, Position As Long) As Long
```

DLL

```
int DPLSetFormFieldCheckStyle(int InstanceID, int Index, int CheckStyle,
  int Position);
```

Parameters

Index	The index of the form field to work with. The first form field has an index of 1.
CheckStyle	0 = Cross 1 = Check (Tick) 2 = Dot (Radio) 3 = XP check 4 = XP Radio 5 = Diamond 6 = Square 7 = Star
Position	0 = Left align 1 = Center 2 = Right align

Return values

0	One of the parameters was invalid
1	The check style was set successfully

SetFormFieldChildTitle

Form fields

Description

Sets the title of the specified form field. For form fields arranged in a hierarchy this function only sets the last part of the field name. For example, a field with the name "Address.ZipCode" can be changed to "Address.PostalCode".

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldChildTitle(Index: Integer;
  NewTitle: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldChildTitle(
  Index As Long, NewTitle As String) As Long
```

DLL

```
int DPLSetFormFieldChildTitle(int InstanceID, int Index,
  wchar_t * NewTitle);
```

Parameters

Index	The index of the form field to set the title of
NewTitle	The new value for the last part of the title for the specified field.

Return values

0	The form field could not be found
1	The title of the specified form field was changed successfully

SetFormFieldChoiceSub

Version history

This function was introduced in Quick PDF Library version 9.16.

Description

Sets the export and display values of an existing sub-field that is part of a choice form field. If the display value is an empty string then it will be set to the same string as the export value.
The **AddFormFieldChoiceSub** function can be used to change a sub-field entry in an existing choice form field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldChoiceSub(Index,
  SubIndex: Integer; SubName, DisplayName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldChoiceSub(
  Index As Long, SubIndex As Long, SubName As String,
  DisplayName As String) As Long
```

DLL

```
int DPLSetFormFieldChoiceSub(int InstanceID, int Index, int SubIndex,
  wchar_t * SubName, wchar_t * DisplayName);
```

Parameters

Index	The index of the choice form field
SubIndex	The index of the sub-field. The first sub-field has an index of 1.
SubName	The export value of the new sub-field.
DisplayName	The display value of the new sub-field.

Return values

0	The sub-field was not added. The specified form field may not have been a choice form field.
1	The form field was updated successfully.

SetFormFieldChoiceType

Version history

This function was introduced in Quick PDF Library version 7.24.

Description

Sets a choice form field to be a combo box or list box.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldChoiceType(Index,
  ChoiceType: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldChoiceType(
  Index As Long, ChoiceType As Long) As Long
```

DLL

```
int DPLSetFormFieldChoiceType(int InstanceID, int Index, int ChoiceType);
```

Parameters

Index	The index of the form field
ChoiceType	1 = Set the form field to be a scrollable list box 2 = Set the form field to be a drop-down combo box 3 = Set the form field to be a multiselect scrollable list box 4 = Set the form field to be a drop-down combo box with edit box

Return values

0	The field was not changed
1	The field was changed successfully

SetFormFieldColor

Description

Sets the color of the text in the form field. The values of the color parameters range from 0 to 1, with 0 indicating 0% and 1 indicating 100% of the color.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldColor(Index: Integer; Red,
  Green, Blue: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldColor(Index As Long,
  Red As Double, Green As Double, Blue As Double) As Long
```

DLL

```
int DPLSetFormFieldColor(int InstanceID, int Index, double Red,
  double Green, double Blue);
```

Parameters

Index	The index of the form field to work with. The first form field has an index of 1.
Red	The red component of the color
Green	The green component of the color
Blue	The blue component of the color

Return values

0	The form field could not be found
1	The form field text color was set successfully

SetFormFieldColorCMYK

Description

Sets the color of the text in the specified form field. Similar to the **SetFormFieldBorderColor** function, but the color components are specified in the CMYK color space (Cyan, Magenta, Yellow, Black). The values of the color parameters range from 0 to 1, with 0 indicating 0% and 1 indicating 100% of the color.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldColorCMYK(Index: Integer; C, M,
  Y, K: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldColorCMYK(
  Index As Long, C As Double, M As Double, Y As Double,
  K As Double) As Long
```

DLL

```
int DPLSetFormFieldColorCMYK(int InstanceID, int Index, double C,
  double M, double Y, double K);
```

Parameters

Index	The index of the form field
C	The cyan component of the color
M	The magenta component of the color
Y	The yellow component of the color
K	The black component of the color

Return values

0	The form field could not be found
1	The text color of the specified form field was set successfully

SetFormFieldColorSep

Description

Sets the color of the text in the specified form field. Similar to the **SetFormFieldBorderColor** function, but a tint of a separation color added with the **AddSeparationColor** function is used. The PDF specification does not support separation color spaces for form fields, so the results may not always work, especially if the form field is later edited in Acrobat. This feature has been added for situations where the form field will be flattened.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldColorSep(Index: Integer;
  ColorName: WideString; Tint: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldColorSep(
  Index As Long, ColorName As String, Tint As Double) As Long
```

DLL

```
int DPLSetFormFieldColorSep(int InstanceID, int Index,
  wchar_t * ColorName, double Tint);
```

Parameters

Index	The index of the form field
ColorName	The name of the separation color that was used with the [f:AddSeparationColor] function
Tint	The amount of color to use. 0 indicates no color (white), 1 indicates maximum color.

Return values

0	The form field could not be found, or the separation color name could not be found
1	The text color of the specified form field was set successfully

SetFormFieldComb

Description

Marks a form field as a comb field, where each character in the value occupies the same space in the field. The field must be a test field, and the **SetFormFieldMaxLen** function must be called to specify the number of characters in the field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldComb(Index,
   Comb: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldComb(Index As Long,
   Comb As Long) As Long
```

DLL

```
int DPLSetFormFieldComb(int InstanceID, int Index, int Comb);
```

Parameters

Index	The index of the form field
Comb	0 = Regular field 1 = Comb field

SetFormFieldDefaultValue

Form fields

Description

Sets the default value of the field. This is the value which is shown when the reset button is pressed, if one is on the form.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldDefaultValue(Index: Integer;
  NewDefaultValue: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldDefaultValue(
  Index As Long, NewDefaultValue As String) As Long
```

DLL

```
int DPLSetFormFieldDefaultValue(int InstanceID, int Index,
  wchar_t * NewDefaultValue);
```

Parameters

Index	The index of the form field to change
NewDefaultValue	The new default value for the form field. For multi-line text fields you can use Chr(13) or Chr(13) + Chr(10) to force a line feed between lines.

Return values

0	The form field could not be found
1	The default value of the specified form field was set successfully

SetFormFieldDescription

Description

Sets the description of the specified form field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldDescription(Index: Integer;
  NewDescription: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldDescription(
  Index As Long, NewDescription As String) As Long
```

DLL

```
int DPLSetFormFieldDescription(int InstanceID, int Index,
  wchar_t * NewDescription);
```

Parameters

Index	The index of the form field to change
NewDescription	The new description.

Return values

0	The form field could not be found
1	The specified form field's description was set successfully

SetFormFieldFlags

Description

Sets the internal flags for the form field. This setting is for advanced purposes and most users will not need to use it.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldFlags(Index,
  NewFlags: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldFlags(Index As Long,
  NewFlags As Long) As Long
```

DLL

```
int DPLSetFormFieldFlags(int InstanceID, int Index, int NewFlags);
```

Parameters

Index	The index of the form field
NewFlags	The new value of the flags. Consult the PDF specification for further details.

Return values

0	Cannot find the form field
1	The flags were set successfully

SetFormFieldFont

Description

Sets the font that the specified form field must use.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldFont(Index,
  FontIndex: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldFont(Index As Long,
  FontIndex As Long) As Long
```

DLL

```
int DPLSetFormFieldFont(int InstanceID, int Index, int FontIndex);
```

Parameters

Index	The index of the form field to work with. The first form field has an index of 1.
FontIndex	The index of the font to use. The first font in the form has an index of 1. Use **GetFormFontCount** to determine the number of fonts available in the form.

Return values

0	Bad font index or form field not found
1	Font was set successfully

SetFormFieldHighlightMode

Form fields

Description

Sets the highlight mode for the specified form field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldHighlightMode(Index,
  NewMode: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldHighlightMode(
  Index As Long, NewMode As Long) As Long
```

DLL

```
int DPLSetFormFieldHighlightMode(int InstanceID, int Index, int NewMode);
```

Parameters

Index	The index of the form field
NewMode	The highlighting mode: 0 = None 1 = Invert 2 = Outline 3 = Push

Return values

0	The form field could not be found or the parameters were invalid
1	The highlight mode of the form field was set successfully

SetFormFieldIcon

Description

Sets the icon of a button form field. To create an icon: add a new page to the document, set the size and draw images or text onto the page, and then capture the page using the **CapturePage** function. For a "down" or "rollover" icon to be displayed correctly the form field's hightlight mode must be set to "push", see the **SetFormFieldHighlightMode** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldIcon(Index, IconType,
  CaptureID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldIcon(Index As Long,
  IconType As Long, CaptureID As Long) As Long
```

DLL

```
int DPLSetFormFieldIcon(int InstanceID, int Index, int IconType,
  int CaptureID);
```

Parameters

Index	The index of the form field
IconType	The type of icon to assign: 0 = Normal icon 1 = Rollover icon 2 = Down icon
CaptureID	The ID returned by the **CapturePage** function

Return values

0	The form field could not be found or the parameters were invalid
1	The specified icon of the form field was set successfully

SetFormFieldIconStyle

Form fields

Description

Sets the position, scaling and layout of a button form field's icon. These parameters apply to all the icons assigned to a button (up, down and rollover).

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldIconStyle(Index, Placement,
  Scale, ScaleType, HorizontalShift, VerticalShift: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldIconStyle(
  Index As Long, Placement As Long, Scale As Long,
  ScaleType As Long, HorizontalShift As Long,
  VerticalShift As Long) As Long
```

DLL

```
int DPLSetFormFieldIconStyle(int InstanceID, int Index, int Placement,
  int Scale, int ScaleType, int HorizontalShift,
  int VerticalShift);
```

Parameters

Index	The index of the form field
Placement	The icon placement: 0 = No icon; caption only 1 = No caption; icon only 2 = Caption below the icon 3 = Caption above the icon 4 = Caption to the right of the icon 5 = Caption to the left of the icon 6 = Caption overlaid directly on the icon
Scale	The conditions under which to scale the icon: 0 = Always scale 1 = Only scale when the icon is bigger than the button 2 = Only scale when the icon is smaller than the button 3 = Never scale
ScaleType	The type of scaling to use: 0 = Ignore aspect ratio 1 = Maintain aspect ratio
HorizontalShift	The percentage of space placed to the left of the icon, for example: 0 = Align left 50 = Center horizontally 100 = Align right
VerticalShift	The percentage of space placed beneath the icon, for example: 0 = Align bottom 50 = Center vertically 100 = Align top

Return values

0	The form field could not be found or the parameters were invalid
1	The icon style of the form field was set successfully

SetFormFieldMaxLen

Description

Sets the maximum number of characters that will be accepted for the specified text form field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldMaxLen(Index,
  NewMaxLen: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldMaxLen(
  Index As Long, NewMaxLen As Long) As Long
```

DLL

```
int DPLSetFormFieldMaxLen(int InstanceID, int Index, int NewMaxLen);
```

Parameters

Index	The index of the form field to work with. The first form field has an index of 1.
NewMaxLen	The new maximum length to use for the form field

Return values

0	The form field index was invalid
1	The maximum length of the form field was set successfully

SetFormFieldNoExport

Form fields

Version history

This function was introduced in Quick PDF Library version 7.24.

Description

Sets the state of a field's NoExport flag.

The field will not be exported by a submit-form action if the NoExport flag is set.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldNoExport(Index,
  NoExport: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldNoExport(
  Index As Long, NoExport As Long) As Long
```

DLL

```
int DPLSetFormFieldNoExport(int InstanceID, int Index, int NoExport);
```

Parameters

Index	The index of the form field
NoExport	0 = Clear the field's NoExport flag 1 = Set the field's NoExport flag

Return values

0	Could not find the specified form field
1	The NoExport flag was set successfully

SetFormFieldOptional

Form fields, Content Streams and Optional Content Groups

Description

Adds a form field to an optional content group.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldOptional(Index,
  OptionalContentGroupID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldOptional(
  Index As Long, OptionalContentGroupID As Long) As Long
```

DLL

```
int DPLSetFormFieldOptional(int InstanceID, int Index,
  int OptionalContentGroupID);
```

Parameters

Index	The index of the form field
OptionalContentGroupID	An ID returned by the **NewOptionalContentGroup**, **GetOptionalContentGroupID** or **GetOptionalContentConfigOrderItemID** functions

Return values

0	The OptionalContentGroupID or Index parameter was invalid
1	The field was added to the optional content group successfully

SetFormFieldPage

Description

Moves the specified form field onto another page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldPage(Index,
  NewPage: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldPage(Index As Long,
  NewPage As Long) As Long
```

DLL

```
int DPLSetFormFieldPage(int InstanceID, int Index, int NewPage);
```

Parameters

Index	The index of the form field to move
NewPage	The page number to move the form field to

Return values

0	Can't find the form field or the new destination page is invalid
1	Form field moved successfully

SetFormFieldPrintable

Description

Set whether the specified form field should be printed or not.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldPrintable(Index,
  Printable: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldPrintable(
  Index As Long, Printable As Long) As Long
```

DLL

```
int DPLSetFormFieldPrintable(int InstanceID, int Index, int Printable);
```

Parameters

Index	The index of the form field to change
Printable	0 = Do not print 1 = Print

Return values

0	The specified form field could not be found
1	The printable flag of the specified form field was set successfully

SetFormFieldReadOnly

Form fields

Description

Sets the state of a field's ReadOnly flag.

The user cannot change the value of a form field if the ReadOnly flag is set.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldReadOnly(Index,
  ReadOnly: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldReadOnly(
  Index As Long, ReadOnly As Long) As Long
```

DLL

```
int DPLSetFormFieldReadOnly(int InstanceID, int Index, int ReadOnly);
```

Parameters

Index	The index of the form field
ReadOnly	0 = Clear the field's ReadOnly flag 1 = Set the field's ReadOnly flag

Return values

0	Could not find the specified form field
1	The ReadOnly flag was set successfully

SetFormFieldRequired

Version history

This function was introduced in Quick PDF Library version 7.24.

Description

Sets the state of a field's is Required flag.

If this flag is set the field must have a value when the form is exported by a submit-form action.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldRequired(Index,
  Required: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldRequired(
  Index As Long, Required As Long) As Long
```

DLL

```
int DPLSetFormFieldRequired(int InstanceID, int Index, int Required);
```

Parameters

Index	The index of the form field
Required	0 = Clear the field's Required flag 1 = Set the field's Required flag

Return values

0	Could not find the specified form field
1	The Required flag was set successfully

SetFormFieldResetAction

Version history

This function was introduced in Quick PDF Library version 9.15.

Description

Adds a reset action to a button form field. When actioned all formfields will be reset to their default values.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldResetAction(Index: Integer;
  ActionType: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldResetAction(
  Index As Long, ActionType As String) As Long
```

DLL

```
int DPLSetFormFieldResetAction(int InstanceID, int Index,
  wchar_t * ActionType);
```

Parameters

Index	The index of the form field
ActionType	The action type:E = An action to be performed when the cursor enters the annotation's active areaX = An action to be performed when the cursor exits the annotation's active areaD = An action to be performed when the mouse button is pressed inside the annotation's active areaU = An action to be performed when the mouse button is released inside the annotation's active areaFo = An action to be performed when the annotation receives the input focusBl = An action to be performed when the annotation loses the input focus (blurred)K = An action to be performed when the user types a keystroke into a text field or combo box or modifies the selection in a scrollable list box. This allows the keystroke to be checked for validity and rejected or modified.F = An action to be performed before the field is formatted to display its current value. This allows the field's value to be modified before formatting.V = An action to be performed when the field's value is changed. This allows the new value to be checked for validity.C = An action to be performed in order to recalculate the value of this field when that of another field changes

Return values

0	Could not set the field action
1	Success

SetFormFieldRichTextString

Version history

This function was introduced in Quick PDF Library version 9.15.

Description

Sets the rich text (RV) or default style (DS) string of the specified form field using the given key.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldRichTextString(Index: Integer;
  Key, NewValue: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldRichTextString(
  Index As Long, Key As String, NewValue As String) As Long
```

DLL

```
int DPLSetFormFieldRichTextString(int InstanceID, int Index,
  wchar_t * Key, wchar_t * NewValue);
```

Parameters

Index	The index of the required form field. The first form field has an index of 1.
Key	The Key used to set the required string "RV" = sets the rich text string "DS" = sets the default style string
NewValue	The new value for the specified key. The required format of the input string is defined in the PDF Specification under the section titled "Field Dictionaries".

Return values

0	Could not find the specified form field
1	The specified value of the form field was set successfully

SetFormFieldRotation

Form fields

Description

Sets the rotation of a form field anti-clockwise relative to the page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldRotation(Index,
  Angle: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldRotation(
  Index As Long, Angle As Long) As Long
```

DLL

```
int DPLSetFormFieldRotation(int InstanceID, int Index, int Angle);
```

Parameters

Index	The index of the form field to work with. The first form field has an index of 1.
Angle	The angle to rotate the field by. Must be one of the following values: 0, 90, 180 or 270.

Return values

0	The form field could not be found or the specified angle was not valid
1	The rotation of the specified form field was set successfully

SetFormFieldSignatureImage

Image handling, Form fields, Security and Signatures

Version history

This function was introduced in Quick PDF Library version 9.13.

Description

Sets the visual appearance of a signature form field to use the specified image.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldSignatureImage(Index, ImageID,
  Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldSignatureImage(
  Index As Long, ImageID As Long, Options As Long) As Long
```

DLL

```
int DPLSetFormFieldSignatureImage(int InstanceID, int Index, int ImageID,
  int Options);
```

Parameters

Index	The index of the signature form field to work with. The first form field has an index of 1.
ImageID	A valid image ID as returned by the **SelectedImage** or **GetImageID** functions.
Options	0 = The image is stretched in both directions to fill the field size without any rotation

Return values

0	The form field was not a signature field, the the Index parameter was out of range or the ImageID parameter was invalid.
1	Success

SetFormFieldStandardFont

Fonts, Form fields

Description

Sets a form field to use a standard font. A standard font must be used in Acrobat 4 and earlier if the form field contains a border or is rotated.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldStandardFont(Index,
  StandardFontID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldStandardFont(
  Index As Long, StandardFontID As Long) As Long
```

DLL

```
int DPLSetFormFieldStandardFont(int InstanceID, int Index,
  int StandardFontID);
```

Parameters

Index	The index of the form field
StandardFontID	The ID of the font to add: 0 = Courier 1 = CourierBold 2 = CourierBoldOblique 3 = CourierOblique 4 = Helvetica 5 = HelveticaBold 6 = HelveticaBoldOblique 7 = HelveticaOblique 8 = TimesRoman 9 = TimesBold 10 = TimesItalic 11 = TimesBoldItalic 12 = Symbol 13 = ZapfDingbats

SetFormFieldSubmitAction

Version history

This function was introduced in Quick PDF Library version 7.25.

Description

Adds a submit action to a button form field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldSubmitAction(Index: Integer;
  ActionType, Link: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldSubmitAction(
  Index As Long, ActionType As String, Link As String) As Long
```

DLL

```
int DPLSetFormFieldSubmitAction(int InstanceID, int Index,
  wchar_t * ActionType, wchar_t * Link);
```

Parameters

Index	The index of the form field
ActionType	The action type: E = An action to be performed when the cursor enters the annotation's active area X = An action to be performed when the cursor exits the annotation's active area D = An action to be performed when the mouse button is pressed inside the annotation's active area U = An action to be performed when the mouse button is released inside the annotation's active area Fo = An action to be performed when the annotation receives the input focus Bl = An action to be performed when the annotation loses the input focus (blurred) K = An action to be performed when the user types a keystroke into a text field or combo box or modifies the selection in a scrollable list box. This allows the keystroke to be checked for validity and rejected or modified. F = An action to be performed before the field is formatted to display its current value. This allows the field's value to be modified before formatting. V = An action to be performed when the field's value is changed. This allows the new value to be checked for validity. C = An action to be performed in order to recalculate the value of this field when that of another field changes
Link	The URL of the server script that will process the form submission.

Return values

0	Could not set the field action
1	Success

SetFormFieldSubmitActionEx

Version history

This function was introduced in Quick PDF Library version 9.15.

Description

Adds a submit action to a button form field with a flags parameter for setting various submit options. Please refer to section "Form Actions" of the official PDF Specifications.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldSubmitActionEx(Index: Integer;
  ActionType, Link: WideString; Flags: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldSubmitActionEx(
  Index As Long, ActionType As String, Link As String,
  Flags As Long) As Long
```

DLL

```
int DPLSetFormFieldSubmitActionEx(int InstanceID, int Index,
  wchar_t * ActionType, wchar_t * Link, int Flags);
```

Parameters

Index	The index of the form field
ActionType	The action type: E = An action to be performed when the cursor enters the annotation's active area X = An action to be performed when the cursor exits the annotation's active area D = An action to be performed when the mouse button is pressed inside the annotation's active area U = An action to be performed when the mouse button is released inside the annotation's active area Fo = An action to be performed when the annotation receives the input focus Bl = An action to be performed when the annotation loses the input focus (blurred) K = An action to be performed when the user types a keystroke into a text field or combo box or modifies the selection in a scrollable list box. This allows the keystroke to be checked for validity and rejected or modified. F = An action to be performed before the field is formatted to display its current value. This allows the field's value to be modified before formatting. V = An action to be performed when the field's value is changed. This allows the new value to be checked for validity. C = An action to be performed in order to recalculate the value of this field when that of another field changes
Link	The URL of the server script that will process the form submission.
Flags	Adobe defined flags value for the formfield submit action.

Return values

0	Could not set the field action
1	Success

SetFormFieldTabOrder

Description

Sets the tab order of the specified form field. A position of 1 indicates that the form field is the first field on the page.

If you use this function then you should call **SetTabOrderMode** with 'S' to set the tabbing mode to Structure mode.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldTabOrder(Index,
  Order: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldTabOrder(
  Index As Long, Order As Long) As Long
```

DLL

```
int DPLSetFormFieldTabOrder(int InstanceID, int Index, int Order);
```

Parameters

Index	The index of the form field that should be moved to a new position in the tab order
Order	The new position this form field should be in the tab order. The first position in the tab order has a value of 1.

Return values

0	The form field could not be found or the new tab order was out of range
1	The tab order of the specified form field was updated successfully

SetFormFieldTextFlags

Description

Sets various options for text form fields.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldTextFlags(Index, Multiline,
  Password, FileSelect, DoNotSpellCheck, DoNotScroll: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldTextFlags(
  Index As Long, Multiline As Long, Password As Long,
  FileSelect As Long, DoNotSpellCheck As Long,
  DoNotScroll As Long) As Long
```

DLL

```
int DPLSetFormFieldTextFlags(int InstanceID, int Index, int Multiline,
  int Password, int FileSelect, int DoNotSpellCheck,
  int DoNotScroll);
```

Parameters

Index	The index of the form field to work with. The first form field has an index of 1.
Multiline	0 = Field's text is restricted to one line 1 = Field may contain multiple lines of text
Password	0 = The field is not a password field 1 = The field is a password, characters will be displayed as asterisks
FileSelect	0 = The field is not a file select field 1 = The contents of the file specified by the text entered in this field will be submitted as the value of the form field
DoNotSpellCheck	0 = The field will be spell checked 1 = The field will not be spell checked
DoNotScroll	0 = Field can scroll 1 = Field is not allowed to scroll

Return values

0	The form field could not be found
1	The options for the text field were set successfully

SetFormFieldTextSize

Text, Form fields

Description

Sets the size of the text in the specified form field. A value of 0 indicates that the form field autosizes the text to fit into the available space.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldTextSize(Index: Integer;
  NewTextSize: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldTextSize(
  Index As Long, NewTextSize As Double) As Long
```

DLL

```
int DPLSetFormFieldTextSize(int InstanceID, int Index, double NewTextSize);
```

Parameters

Index	The index of the form field to work with. The first form field has an index of 1.
NewTextSize	The new size in points of the form field's font

Return values

0	The form field could not be found
1	The form field font size was set successfully

SetFormFieldTitle

Version history

This function was introduced in Quick PDF Library version 10.12.

Description

Renames the title of a parent form field. No validation is performed so you should make sure the title is unique.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldTitle(Index: Integer;
  NewTitle: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldTitle(Index As Long,
  NewTitle As String) As Long
```

DLL

```
int DPLSetFormFieldTitle(int InstanceID, int Index, wchar_t * NewTitle);
```

Parameters

Index	The index of the formfield
NewTitle	The new title name for the formfield

Return values

0	The form field title cound not be changed
1	The field field title was changed successfully

SetFormFieldValue

Description

Sets the initial value of a form field. The appearance stream for the form field is generated if **SetNeedAppearances**(1) has been called.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldValue(Index: Integer;
  NewValue: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldValue(Index As Long,
  NewValue As String) As Long
```

DLL

```
int DPLSetFormFieldValue(int InstanceID, int Index, wchar_t * NewValue);
```

Parameters

Index	The index of the required form field. The first form field has an index of 1.
NewValue	The new value of the form field. For multi-line text fields you can use Chr(13) or Chr(13) + Chr(10) to force a line break.

Return values

0	Could not find the specified form field
1	The default value of the form field was set successfully

SetFormFieldValueByTitle

Form fields

Description

Sets the value of all the form fields with the specified title. The appearance streams for the form fields are generated if **SetNeedAppearances**(1) has been called.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldValueByTitle(Title,
  NewValue: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldValueByTitle(
  Title As String, NewValue As String) As Long
```

DLL

```
int DPLSetFormFieldValueByTitle(int InstanceID, wchar_t * Title,
  wchar_t * NewValue);
```

Parameters

Title	The title of the form field to set.
NewValue	The new value of the form field. For multi-line text fields you can use Chr(13) or Chr(13) + Chr(10) to force a line feed between lines.

Return values

0	The form field could not be found
1	The value of the form field was set successfully

SetFormFieldVisible

Form fields

Description

Hides or shows the a form field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetFormFieldVisible(Index,
  Visible: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetFormFieldVisible(
  Index As Long, Visible As Long) As Long
```

DLL

```
int DPLSetFormFieldVisible(int InstanceID, int Index, int Visible);
```

Parameters

Index	The index of the required form field. The first form field has an index of 1.
Visible	0 = Hide the form field 1 = Show the form field

Return values

0	Could not find the specified form field
1	The visiblity of the form field was set successfully

SetGDIPlusFileName

Description

Sets the path and filename of the GDI+ DLL (gdiplus.dll) used by the various rendering functions. This can usually be left at the default, which means the DLL will most probably be stored in the Windows/System folder, but on webservers, etc. it may be necessary to store the file in a different location.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetGDIPlusFileName(
  DLLFileName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetGDIPlusFileName(
  DLLFileName As String) As Long
```

DLL

```
int DPLSetGDIPlusFileName(int InstanceID, wchar_t * DLLFileName);
```

Parameters

DLLFileName	The path and file name of the GDI+ DLL, for example "c:\dlls\gdiplus.dll".

Return values

0	The specified file could not be found
1	The GDI+ DLL file name was set successfully

SetGDIPlusOptions

Description

Sets various options for the renderer when the GDI+ library is used.

Options 10, 11 and 12 will override options 2 and 3 if they are set to anything other than 0.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetGDIPlusOptions(OptionID,
  NewValue: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetGDIPlusOptions(
  OptionID As Long, NewValue As Long) As Long
```

DLL

```
int DPLSetGDIPlusOptions(int InstanceID, int OptionID, int NewValue);
```

Parameters

OptionID	0 = Use of GDI+
	1 = Text/vector graphics smoothing
	2 = Interpolation
	3 = Image smoothing
	4 = Process null paths
	5 = Mono threshold
	6 = DLL piggyback
	7 = DLL startup
	8 = DLL suppress background thread
	9 = Enhance thin lines
	10 = GDIPlus SmoothingMode
	11 = GDIPlus InterpolationMode
	12 = GDIPlus PixelOffsetMode
NewValue	For use of GDI+:
	0 = Do not use GDI+
	1 = Use GDI+ (default)
	For text/vector graphics smoothing:
	0 = No smoothing
	1 = Smooth text and vector graphics (default)
	For interpolation:
	0 = Standard
	1 = Accurate (default)
	For images:
	0 = No smoothing (default)
	1 = Smoothing
	For null paths:
	0 = Ignore
	1 = Process (default)
	For the mono threshold:
	0 = No thresholding (default)
	1..255 = Threshold level
	6 = DLL piggyback
	7 = DLL startup
	8 = DLL suppress background thread
	For DLL piggyback:
	0 = Do not allow
	1 = Allow (reuse existing DLL instance)
	For DLL startup (GdiplusStartup/GdiplusShutdown)
	0 = Never call
	1 = Don't call if piggybacking on existing DLL
	2 = Always call
	For DLL suppress background thread:
	0 = No (do not suppress)
	1 = Yes (suppress background thread)
	For Enhance thin lines:
	0 = Do nothing (default)
	1 - 9 = Thicken lines smaller than 1 device unit to thickness specified
	For GDIPlus SmoothingMode
	0 = smDefault, 1 = smHighSpeed, 2= smHighQuality, 3 = smNone, 4 = smAntiAlias
	For GDIPlus InterpolationMode
	0 = imDefault, 1 = imLowQuality, 2 = imHighQuality,
	3 = imBilinear, 4 = imBicubic, 5 = NearestNeighbor
	6 = imHighQualityBilinear, 7 = imHighQualityBicubic
	For GDIPlus PixelOffsetMode
	0 = poDefault, 1 = poHighSpeed, 2 = poHighQuality
	4 = poNone, 4 = poHalf

SetHTMLBoldFont

Description

Specifies the font to use when a <b> or <strong> tag is encountered when using **DrawHTMLText** or **DrawHTMLTextBox**.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetHTMLBoldFont(FontSet: WideString;
  FontID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetHTMLBoldFont(
  FontSet As String, FontID As Long) As Long
```

DLL

```
int DPLSetHTMLBoldFont(int InstanceID, wchar_t * FontSet, int FontID);
```

Parameters

FontSet	The name of the font set to use. For this version of the library it should always be "Default".
FontID	The ID of the font to use

Return values

0	The specified FontID is not a valid font
1	The font was set successfully

SetHTMLBoldItalicFont

Description

Specifies the font to use when both <b> or <strong> and <em> or <i> tags are encountered when using **DrawHTMLText** or **DrawHTMLTextBox**.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetHTMLBoldItalicFont(FontSet: WideString;
  FontID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetHTMLBoldItalicFont(
  FontSet As String, FontID As Long) As Long
```

DLL

```
int DPLSetHTMLBoldItalicFont(int InstanceID, wchar_t * FontSet,
  int FontID);
```

Parameters

FontSet	The name of the font set to use. For this version of the library it should always be "Default".
FontID	The ID of the font to use

Return values

0	The specified FontID is not a valid font
1	The font was set successfully

SetHTMLItalicFont

Description

Specifies the font to use when an <em> or <i> tag is encountered when using **DrawHTMLText** or **DrawHTMLTextBox**.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetHTMLItalicFont(FontSet: WideString;
  FontID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetHTMLItalicFont(
  FontSet As String, FontID As Long) As Long
```

DLL

```
int DPLSetHTMLItalicFont(int InstanceID, wchar_t * FontSet, int FontID);
```

Parameters

FontSet	The name of the font set to use. For this version of the library it should always be "Default".
FontID	The ID of the font to use

Return values

0	The specified FontID is not a valid font
1	The font was set successfully

SetHTMLNormalFont

Description

Specifies the default font for text drawn using **DrawHTMLText** or **DrawHTMLTextBox**.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetHTMLNormalFont(FontSet: WideString;
  FontID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetHTMLNormalFont(
  FontSet As String, FontID As Long) As Long
```

DLL

```
int DPLSetHTMLNormalFont(int InstanceID, wchar_t * FontSet, int FontID);
```

Parameters

FontSet	The name of the font set to use. For this version of the library it should always be "Default".
FontID	The ID of the font to use

Return values

0	The specified FontID is not a valid font
1	The font was set successfully

SetHeaderCommentsFromString

Document properties

Version history

This function was introduced in Quick PDF Library version 9.16.

Description

Allows a binary string to be added between the file header and first objects. The string should start with a % character to indicate that it is a comment.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetHeaderCommentsFromString(
  const Source: AnsiString): Integer;
```

DLL

```
int DPLSetHeaderCommentsFromString(int InstanceID, char * Source);
```

Parameters

Source	The new comments

Return values

0	The header comments could not be set
1	Success

SetHeaderCommentsFromVariant

Version history

This function was introduced in Quick PDF Library version 9.16.

Description

Allows a binary string to be added between the file header and first objects. The string should start with a % character to indicate that it is a comment.

Syntax

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetHeaderCommentsFromVariant(
  Source As Variant) As Long
```

Parameters

Source	A byte array containing the new comments

Return values

0	The header comments could not be set
1	Success

SetImageAsMask

Image handling, Page layout

Description

This function must be called to prepare the image before it is used as a mask for another image. The mask image must be a grayscale image, and be either 1-bit or 8-bit. Depending on your needs you may want to call **ReverseImage** which will reverse the effects of the mask. A soft-mask is just a normal image, so if you have an image setup as a stencil mask and no longer want it to be a mask just change it to a soft mask image (MaskType = 2).

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetImageAsMask(MaskType: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetImageAsMask(
  MaskType As Long) As Long
```

DLL

```
int DPLSetImageAsMask(int InstanceID, int MaskType);
```

Parameters

MaskType	The type of mask to set this image as: 1 = Stencil mask (only 1-bit images) 2 = Soft mask (1-bit and 8-bit images)

SetImageMask

Image handling, Page layout

Description

Sets the mask for the selected image. This can be used to make parts of an image transparent when it is drawn with the **DrawImage** or **DrawScaledImage** functions.

The color range specified will become transparent. This works best with losslessly compressed images such as BMP or TIFF.

JPEG images use a lossy compression, so the image mask may cause halo effects.

The values of the color parameters range from 0 to 1, with 0 indicating 0% and 1 indicating 100% of the color. For monochrome images only the red component will be used.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetImageMask(FromRed, FromGreen, FromBlue,
  ToRed, ToGreen, ToBlue: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetImageMask(FromRed As Double,
  FromGreen As Double, FromBlue As Double, ToRed As Double,
  ToGreen As Double, ToBlue As Double) As Long
```

DLL

```
int DPLSetImageMask(int InstanceID, double FromRed, double FromGreen,
  double FromBlue, double ToRed, double ToGreen, double ToBlue);
```

Parameters

FromRed	The red component of the starting color for the mask
FromGreen	The green component of the starting color for the mask
FromBlue	The blue component of the starting color for the mask
ToRed	The red component of the ending color for the mask
ToGreen	The green component of the ending color for the mask
ToBlue	The blue component of the ending color for the mask

Return values

0	No image was selected
1	The image mask was set successfully

SetImageMaskCMYK

Description

Sets the mask for the selected image. This can be used to make parts of an image transparent when it is drawn with the **DrawImage** or **DrawScaledImage** functions. The color range specified will become transparent. Use this function when the image you added is a CMYK image. Use the **SetImageMask** function for RGB images. The values of the color parameters range from 0 to 1, with 0 indicating 0% and 1 indicating 100% of the color.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetImageMaskCMYK(FromC, FromM, FromY,
  FromK, ToC, ToM, ToY, ToK: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetImageMaskCMYK(
  FromC As Double, FromM As Double, FromY As Double,
  FromK As Double, ToC As Double, ToM As Double, ToY As Double,
  ToK As Double) As Long
```

DLL

```
int DPLSetImageMaskCMYK(int InstanceID, double FromC, double FromM,
  double FromY, double FromK, double ToC, double ToM,
  double ToY, double ToK);
```

Parameters

FromC	The cyan component of the starting color for the mask
FromM	The magenta component of the starting color for the mask
FromY	The yellow component of the starting color for the mask
FromK	The black component of the starting color for the mask
ToC	The red component of the ending color for the mask
ToM	The magenta component of the ending color for the mask
ToY	The yellow component of the ending color for the mask
ToK	The black component of the ending color for the mask

Return values

0	No image was selected
1	The image mask was set successfully

SetImageMaskFromImage

Image handling, Page layout

Description

Use this function to use another image as a transparency mask for the selected image. The mask image must be a grayscale image. If it is not specifically prepared it will be added as a soft mask which only works with Acrobat 5.0 and later. If it is specially prepared using the **SetImageAsMask** function you can choose whether the image will be a stencil mask (which will work with Acrobat 4.0 and later) or a soft mask (which will only work with Acrobat 5.0 and later). Remember that soft masks and stencil masks treat opaque and transparent in an opposite fashion. You may want to call **ReverseImage** on your mask image to ensure consistent results. For compatibility with Acrobat 6.0 and later it is important to set the transparency group for the page to ensure RGB colors in your image are not converted to CMYK yielding strange results. Use the **SetPageTransparencyGroup** function for this. To avoid problems with Acrobat 4.0 you may want to remove the /Decode array from the mask image. This can be achieved with the **ReverseImage** function setting the Reset parameter to 0.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetImageMaskFromImage(
  ImageID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetImageMaskFromImage(
  ImageID As Long) As Long
```

DLL

```
int DPLSetImageMaskFromImage(int InstanceID, int ImageID);
```

Parameters

ImageID	The ID of the image to use as the mask

SetImageOptional

Description

Links the specified image to an optional content group. This allows the image to be selectively shown in Acrobat 6 or later.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetImageOptional(
  OptionalContentGroupID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetImageOptional(
  OptionalContentGroupID As Long) As Long
```

DLL

```
int DPLSetImageOptional(int InstanceID, int OptionalContentGroupID);
```

Parameters

OptionalContentGroupID	An ID returned by the **NewOptionalContentGroup**, **GetOptionalContentGroupID** or **GetOptionalContentConfigOrderItemID** functions

SetImageResolution

Image handling

Version history

This function was introduced in Quick PDF Library version 7.22.

Description

Sets the horizontal and vertical resolution of the selected image as well as the resolution units. These values are used by the **FitImage** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetImageResolution(Horizontal, Vertical,
  Units: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetImageResolution(
  Horizontal As Long, Vertical As Long, Units As Long) As Long
```

DLL

```
int DPLSetImageResolution(int InstanceID, int Horizontal, int Vertical,
  int Units);
```

Parameters

Horizontal	The new horizontal resolution of the image
Vertical	The new vertical resolution of the image
Units	0 = Unknown 1 = No units, values specify the aspect ratio 2 = Dots per inch (DPI) 3 = Dots per centimetre (DPCM)

Return values

0	No image was selected
1	The resolution of the image was set successfully

SetInformation

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Set the properties of the selected document.

For the CreationDate and ModDate (modification date) properties, the format of the date should be:

D:YYYYMMDDHHmmSSOHH'mm'

where
YYYY shall be the year
MM shall be the month (01-12)
DD shall be the day (01-31)
HH shall be the hour (00-23)
mm shall be the minute (00-59)
SS shall be the second (00-59)
O shall be the relationship of local time to Universal Time (UT) using a +, - or Z character
HH followed by APOSTROPHE (U+0027) (') shall be the absolute value of the offset from UT in hours (00-23)
mm followed by an optional APOSTROPHE (U+0027) (') shall be the absolute value of the offset from UT in minutes (00-59)

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetInformation(Key: Integer;
  NewValue: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetInformation(Key As Long,
  NewValue As String) As Long
```

DLL

```
int DPLSetInformation(int InstanceID, int Key, wchar_t * NewValue);
```

Parameters

Key	The property to set: 0 = PDF Version 1 = Author 2 = Title 3 = Subject 4 = Keywords 5 = Creator 6 = Producer 7 = CreationDate 8 = ModDate 9 = XMP dc:subject
NewValue	The new value of the specified property.

Return values

0	The specified information could not be set. Use the **LastErrorCode** function to determine the reason for failure.
1	The specified information was set successfully

SetJPEGQuality

Rendering and printing

Description

Sets the quality for any JPEG images produced by the library.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetJPEGQuality(Quality: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetJPEGQuality(
  Quality As Long) As Long
```

DLL

```
int DPLSetJPEGQuality(int InstanceID, int Quality);
```

Parameters

Quality	A number between 1 and 100 indicating the quality of the image. The higher the value, the better the image quality, but the larger the file size. The lower the value, the smaller the resulting file size, but at the expense of picture quality.

SetJavaScriptMode

Document properties, JavaScript

Description

This function allows you to control the way JavaScript is stored in the document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetJavaScriptMode(JSMode: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetJavaScriptMode(
  JSMode As Long) As Long
```

DLL

```
int DPLSetJavaScriptMode(int InstanceID, int JSMode);
```

Parameters

JSMode	1 = Store JavaScript in a stream
	2 = Store JavaScript in a compressed stream
	Anything else = Store JavaScript as a string (default)

SetKerning

Description

Sets the amount of kerning for the specified character pair.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetKerning(CharPair: WideString;
  Adjustment: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetKerning(CharPair As String,
  Adjustment As Long) As Long
```

DLL

```
int DPLSetKerning(int InstanceID, wchar_t * CharPair, int Adjustment);
```

Parameters

CharPair	A two-character string containing the characters making the kerning pair, for example "AW"
Adjustment	The amount to reduce the space between the kerning pair by. This is the same value as shown in graphics programs such as Adobe Illustrator. A value of 1000 is the same as the height of the text.

Return values

0	The kerning could not be set. Either the CharPair was not 2 characters in length, or a font has not been selected.
1	The kerning for the specified pair of characters was set successfully

SetLineCap

Description

Sets the line cap style for subsequently drawn lines.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetLineCap(LineCap: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetLineCap(
  LineCap As Long) As Long
```

DLL

```
int DPLSetLineCap(int InstanceID, int LineCap);
```

Parameters

LineCap	The line cap style to use: 0 = Butt 1 = Round 2 = Projecting square cap

Return values

0	The LineCap parameter was not valid
1	The line cap style was set successfully

SetLineColor

Vector graphics, Color

Description

Sets the outline color for any subsequently drawn graphics. The values of the color parameters range from 0 to 1, with 0 indicating 0% and 1 indicating 100% of the color.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetLineColor(Red, Green,
  Blue: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetLineColor(Red As Double,
  Green As Double, Blue As Double) As Long
```

DLL

```
int DPLSetLineColor(int InstanceID, double Red, double Green, double Blue);
```

Parameters

Red	The red component of the color
Green	The green component of the color
Blue	The blue component of the color

SetLineColorCMYK

Description

Sets the outline color of subsequently drawn graphics. Similar to the **SetLineColor** function, but the color components are specified in CMYK mode (Cyan, Magenta, Yellow and Black). The values of the color parameters range from 0 to 1, with 0 indicating 0% and 1 indicating 100% of the color.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetLineColorCMYK(C, M, Y,
  K: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetLineColorCMYK(C As Double,
  M As Double, Y As Double, K As Double) As Long
```

DLL

```
int DPLSetLineColorCMYK(int InstanceID, double C, double M, double Y,
  double K);
```

Parameters

C	The cyan component of the color
M	The magenta component of the color
Y	The yellow component of the color
K	The black component of the color

SetLineColorSep

Description

Sets the outline color of subsequently drawn graphics. Similar to the **SetFillColor** function, but a tint of a separation color added with the **AddSeparationColor** function is used.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetLineColorSep(ColorName: WideString;
  Tint: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetLineColorSep(
  ColorName As String, Tint As Double) As Long
```

DLL

```
int DPLSetLineColorSep(int InstanceID, wchar_t * ColorName, double Tint);
```

Parameters

ColorName	The name of the separation color that was used with the **AddSeparationColor** function
Tint	The amount of color to use. 0 indicates no color (white), 1 indicates maximum color.

Return values

0	The separation color name could not be found
1	The line color was set successfully

SetLineDash

Description

Sets the outline dash pattern for subsequently drawn graphics.

Calling this function with either parameter set to zero will return to a solid line style for subseqently drawn graphics.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetLineDash(DashOn,
  DashOff: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetLineDash(DashOn As Double,
  DashOff As Double) As Long
```

DLL

```
int DPLSetLineDash(int InstanceID, double DashOn, double DashOff);
```

Parameters

DashOn	The width of the dashes
DashOff	The width of the space between the dashes

SetLineDashEx

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Sets the outline dash pattern for subsequently drawn graphics. The dash pattern can be specified with a series of numeric values as per the PDF specification.

Calling this function with an empty string for the DashValues parameter will return to a solid line style for subseqently drawn graphics.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetLineDashEx(
  DashValues: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetLineDashEx(
  DashValues As String) As Long
```

DLL

```
int DPLSetLineDashEx(int InstanceID, wchar_t * DashValues);
```

Parameters

DashValues	The dash pattern to use.

Return values

0	The value of the DashValues parameter was not valid. It should be a list of numeric values separated by spaces. For example "1 1 5 1".
1	The dash pattern was set successfully.

SetLineJoin

Description

Sets the line join style for subsequently drawn graphics.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetLineJoin(LineJoin: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetLineJoin(
  LineJoin As Long) As Long
```

DLL

```
int DPLSetLineJoin(int InstanceID, int LineJoin);
```

Parameters

LineJoin	The line join style to use:
	0 = Miter join
	1 = Round join
	2 = Bevel join

Return values

0	The LineJoin parameter was invalid
1	The line join style was set successfully

SetLineShader

Vector graphics, Path definition and drawing, Color

Version history

This function was introduced in Quick PDF Library version 7.11.

Description

Sets the outline color to the specified shader for subsequently drawn graphics.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetLineShader(
  ShaderName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetLineShader(
  ShaderName As String) As Long
```

DLL

```
int DPLSetLineShader(int InstanceID, wchar_t * ShaderName);
```

Parameters

ShaderName	The shader name that was used when the shader was created.

Return values

0	The shader could not be found
1	The shader outline was setup correctly

SetLineWidth

Description

Sets the outline width for any subsequently drawn shapes.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetLineWidth(LineWidth: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetLineWidth(
   LineWidth As Double) As Long
```

DLL

```
int DPLSetLineWidth(int InstanceID, double LineWidth);
```

Parameters

LineWidth	The width to use

SetMarkupAnnotStyle

Color, Annotations and hotspot links

Version history

This function was introduced in Quick PDF Library version 7.25.

Description

Sets the background color and transparency of a text markup annotation.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetMarkupAnnotStyle(Index: Integer; Red,
  Green, Blue, Transparency: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetMarkupAnnotStyle(
  Index As Long, Red As Double, Green As Double, Blue As Double,
  Transparency As Double) As Long
```

DLL

```
int DPLSetMarkupAnnotStyle(int InstanceID, int Index, double Red,
  double Green, double Blue, double Transparency);
```

Parameters

Index	The index of the annotation. The first annotation on the page has an index of 1.
Red	The red component of the color
Green	The green component of the color
Blue	The blue component of the color
Transparency	The amount of transparency to apply 0 = No transparency 50 = 50% transparency 100 = Invisible

SetMeasureDictBoundsCount

Measurement and coordinate units

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Sets the number of items in the Bounds array of a measurement dictionary.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetMeasureDictBoundsCount(MeasureDictID,
  NewCount: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetMeasureDictBoundsCount(
  MeasureDictID As Long, NewCount As Long) As Long
```

DLL

```
int DPLSetMeasureDictBoundsCount(int InstanceID, int MeasureDictID,
  int NewCount);
```

Parameters

MeasureDictID	A value returned from the **GetImageMeasureDict** function
NewCount	The new number of items in the list. Must be a multiple of 2.

Return values

0	The MeasureDictID parameter was incorrect
1	Success

SetMeasureDictBoundsItem

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Sets the value of an item in the Bounds array of a measurement dictionary.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetMeasureDictBoundsItem(MeasureDictID,
  ItemIndex: Integer; NewValue: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetMeasureDictBoundsItem(
  MeasureDictID As Long, ItemIndex As Long,
  NewValue As Double) As Long
```

DLL

```
int DPLSetMeasureDictBoundsItem(int InstanceID, int MeasureDictID,
  int ItemIndex, double NewValue);
```

Parameters

MeasureDictID	A value returned from the **GetImageMeasureDict** function
ItemIndex	The index of the item to set. The first item has an index of 1.
NewValue	The new value of the item.

Return values

0	The MeasureDictID parameter was incorrect or the ItemIndex parameter was out of range
1	Success

SetMeasureDictCoordinateSystem

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Sets the coordinate system of a measurement dictionary.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetMeasureDictCoordinateSystem(
  MeasureDictID, CoordinateSystemID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetMeasureDictCoordinateSystem(
  MeasureDictID As Long, CoordinateSystemID As Long) As Long
```

DLL

```
int DPLSetMeasureDictCoordinateSystem(int InstanceID, int MeasureDictID,
  int CoordinateSystemID);
```

Parameters

MeasureDictID	A value returned from the **GetImageMeasureDict** function
CoordinateSystemID	1 = Rectilinear coordinate system (RL) 2 = Geospatial coordinate system (GEO)

Return values

0	The MeasureDictID parameter was incorrect or the CoordinateSystemID parameter was out of range
1	Success

SetMeasureDictGPTSCount

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Sets the number of items in the GPTS array of a measurement dictionary.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetMeasureDictGPTSCount(MeasureDictID,
  NewCount: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetMeasureDictGPTSCount(
  MeasureDictID As Long, NewCount As Long) As Long
```

DLL

```
int DPLSetMeasureDictGPTSCount(int InstanceID, int MeasureDictID,
  int NewCount);
```

Parameters

MeasureDictID	A value returned from the **GetImageMeasureDict** function
NewCount	The new number of items in the list. Must be a multiple of 2.

Return values

0	The MeasureDictID parameter was incorrect
1	Success

SetMeasureDictGPTSItem

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Sets the value of an item in the GPTS array of a measurement dictionary.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetMeasureDictGPTSItem(MeasureDictID,
  ItemIndex: Integer; NewValue: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetMeasureDictGPTSItem(
  MeasureDictID As Long, ItemIndex As Long,
  NewValue As Double) As Long
```

DLL

```
int DPLSetMeasureDictGPTSItem(int InstanceID, int MeasureDictID,
  int ItemIndex, double NewValue);
```

Parameters

MeasureDictID	A value returned from the **GetImageMeasureDict** function
ItemIndex	The index of the item to set. The first item has an index of 1.
NewValue	The new value of the item.

Return values

0	The MeasureDictID parameter was incorrect or the ItemIndex parameter was out of range
1	Success

SetMeasureDictLPTSCount

Measurement and coordinate units

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Sets the number of items in the LPTS array of a measurement dictionary.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetMeasureDictLPTSCount(MeasureDictID,
  NewCount: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetMeasureDictLPTSCount(
  MeasureDictID As Long, NewCount As Long) As Long
```

DLL

```
int DPLSetMeasureDictLPTSCount(int InstanceID, int MeasureDictID,
  int NewCount);
```

Parameters

MeasureDictID	A value returned from the **GetImageMeasureDict** function
NewCount	The new number of items in the list. Must be a multiple of 2.

Return values

0	The MeasureDictID parameter was incorrect
1	Success

SetMeasureDictLPTSItem

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Sets the value of an item in the LPTS array of a measurement dictionary.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetMeasureDictLPTSItem(MeasureDictID,
  ItemIndex: Integer; NewValue: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetMeasureDictLPTSItem(
  MeasureDictID As Long, ItemIndex As Long,
  NewValue As Double) As Long
```

DLL

```
int DPLSetMeasureDictLPTSItem(int InstanceID, int MeasureDictID,
  int ItemIndex, double NewValue);
```

Parameters

MeasureDictID	A value returned from the **GetImageMeasureDict** function
ItemIndex	The index of the item to set. The first item has an index of 1.
NewValue	The new value of the item.

Return values

0	The MeasureDictID parameter was incorrect or the ItemIndex parameter was out of range
1	Success

SetMeasureDictPDU

Measurement and coordinate units

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Sets the page display units of a measurement dictionary.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetMeasureDictPDU(MeasureDictID,
  LinearUnit, AreaUnit, AngularUnit: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetMeasureDictPDU(
  MeasureDictID As Long, LinearUnit As Long, AreaUnit As Long,
  AngularUnit As Long) As Long
```

DLL

```
int DPLSetMeasureDictPDU(int InstanceID, int MeasureDictID,
  int LinearUnit, int AreaUnit, int AngularUnit);
```

Parameters

MeasureDictID	A value returned from the **GetImageMeasureDict** function
LinearUnit	1 = M (a meter) 2 = KM (a kilometer) 3 = FT (an international foot) 4 = USFT (a U.S. Survey foot) 5 = MI (an international mile) 6 = NM (an international nautical mile)
AreaUnit	1 = SQM (a square meter) 2 = HA (a hectare = 10,000 square meters) 3 = SQKM (a square kilometer) 4 = SQFT (a square foot) 5 = A (an acre) 6 = SQMI (a square mile)
AngularUnit	1 = DEG (a degree) 2 = GRD (a grad = 0.9 degrees)

Return values

0	The MeasureDictID parameter was incorrect or one of the other parameters was out of range.
1	Success

SetMeasurementUnits

Measurement and coordinate units

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Set the units to use for all measurements given to and returned from the library.

Default user space is exactly 1/72 inches per unit, which is approximately the same as a "point", a unit used in the printing industry. 25.4 millimetres is one inch.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetMeasurementUnits(
  MeasurementUnits: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetMeasurementUnits(
  MeasurementUnits As Long) As Long
```

DLL

```
int DPLSetMeasurementUnits(int InstanceID, int MeasurementUnits);
```

Parameters

MeasurementUnits	The units to use: 0 = Default user space 1 = Millimetres 2 = Inches Anything else = Default user space

SetNeedAppearances

Description

Sets the value of the document's "NeedAppearances" key. Setting this to 1 (True) will instruct the PDF viewer to create the appearances for the form fields when the document is loaded. The document must have at least one form field for this function to have any effect.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetNeedAppearances(
  NewValue: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetNeedAppearances(
  NewValue As Long) As Long
```

DLL

```
int DPLSetNeedAppearances(int InstanceID, int NewValue);
```

Parameters

NewValue	0 = Set NeedAppearances to False 1 = Set NeedAppearances to True

Return values

0	The document does not have any form fields
1	The NeedAppearances flag was set successfully

SetObjectFromString

Version history

This function was renamed in Quick PDF Library version 8.11.
The function name in earlier versions was SetObjectSource.

Description

Sets the raw PDF object data for the specified object number. This is for advanced use only.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetObjectFromString(ObjectNumber: Integer;
  const Source: AnsiString): Integer;
```

DLL

```
int DPLSetObjectFromString(int InstanceID, int ObjectNumber,
  char * Source);
```

Parameters

ObjectNumber	The number of the object to update. The first object is numbered 1 and the last object has an object number equal to the result of the **GetObjectCount** function.
Source	The raw PDF object data to associate with the specified object.

Return values

0	The ObjectNumber parameter was out of bounds.
1	The specified object was updated successfully.

SetObjectFromVariant

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Sets the raw PDF object data for the specified object number from a variant byte array. This is for advanced use only.

Syntax

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetObjectFromVariant(
    ObjectNumber As Long, NewValue As Variant) As Long
```

Parameters

ObjectNumber	The number of the object to update. The first object is numbered 1 and the last object has an object number equal to the result of the **GetObjectCount** function.
NewValue	The raw PDF object data to associate with the specified object.

Return values

0	The ObjectNumber parameter was out of bounds.
1	The specified object was updated successfully.

SetOpenActionDestination

Description

This function allows the opening page and zoom factor to be set for the selected document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetOpenActionDestination(OpenPage,
  Zoom: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetOpenActionDestination(
  OpenPage As Long, Zoom As Long) As Long
```

DLL

```
int DPLSetOpenActionDestination(int InstanceID, int OpenPage, int Zoom);
```

Parameters

OpenPage	The page number to jump to when the document is opened
Zoom	The zoom percentage to use when the document is opened: 0..1600 = percentage zoom -1 = Fit in window -2 = Fit width

Return values

0	The open action could not be set
1	The open action was set successfully

SetOpenActionDestinationFull

Document properties

Version history

This function was introduced in Quick PDF Library version 7.12.

Description

This function allows the opening page and various sizing/positioning values to be set for the selected document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetOpenActionDestinationFull(OpenPage,
  Zoom, DestType: Integer; Left, Top, Right, Bottom: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetOpenActionDestinationFull(
  OpenPage As Long, Zoom As Long, DestType As Long,
  Left As Double, Top As Double, Right As Double,
  Bottom As Double) As Long
```

DLL

```
int DPLSetOpenActionDestinationFull(int InstanceID, int OpenPage,
  int Zoom, int DestType, double Left, double Top, double Right,
  double Bottom);
```

Parameters

OpenPage	The page number to jump to when the document is opened
Zoom	The zoom percentage to use when the document is opened, valid values from 0 to 6400. Only used for DestType = 1, should be set to 0 for other DestTypes.
DestType	1 = "XYZ" - the target page is positioned at the point specified by the Left and Top parameters. The Zoom parameter specifies the zoom percentage. 2 = "Fit" - the entire page is zoomed to fit the window. None of the other parameters are used and should be set to zero. 3 = "FitH" - the page is zoomed so that the entire width of the page is visible. The height of the page may be greater or less than the height of the window. The page is positioned at the vertical position specified by the Top parameter. 4 = "FitV" - the page is zoomed so that the entire height of the page can be seen. The width of the page may be greater or less than the width of the window. The page is positioned at the horizontal position specified by the Left parameter. 5 = "FitR" - the page is zoomed so that a certain rectangle on the page is visible. The Left, Top, Right and Bottom parameters define the rectangular area on the page. 6 = "FitB" - the page is zoomed so that it's bounding box is visible. 7 = "FitBH" - the page is positioned vertically at the position specified by the Top parameter. The page is zoomed so that the entire width of the page's bounding box is visible. 8 = "FitBV" - the page is positioned at the horizontal position specified by the Left parameter. The page is zoomed just enough to fit the entire height of the bounding box into the window.
Left	The horizontal position used by DestType = 1, 4, 5 and 8
Top	The vertical position used by DestType = 1, 3, 5 and 7
Right	The horizontal position of the righthand edge of the rectangle. Used by DestType = 5
Bottom	The horizontal position of the bottom of the rectangle. Used by DestType = 5

Return values

0	The open action destination could not be set. The usually indicates that the Zoom or DestType parameters are out of range.
1	The open action destination was set successfully.

SetOpenActionJavaScript

Document properties, JavaScript

Description

Use this function to run a block of JavaScript as the document is opened.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetOpenActionJavaScript(
  JavaScript: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetOpenActionJavaScript(
  JavaScript As String) As Long
```

DLL

```
int DPLSetOpenActionJavaScript(int InstanceID, wchar_t * JavaScript);
```

Parameters

JavaScript	The JavaScript to use for this action.

Return values

0	The JavaScript could not be added
1	The JavaScript was added successfully

SetOpenActionMenu

Description

Specifies an Acrobat menu item to execute when the document is first loaded.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetOpenActionMenu(
  MenuItem: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetOpenActionMenu(
  MenuItem As String) As Long
```

DLL

```
int DPLSetOpenActionMenu(int InstanceID, wchar_t * MenuItem);
```

Parameters

MenuItem	The menu item which should be executed, for example "print"

Return values

0	The open action could not be set
1	The open action was set successfully

SetOptionalContentConfigLocked

Content Streams and Optional Content Groups

Version history

This function was introduced in Quick PDF Library version 8.15.

Description

This function is used to lock an optional content group as defined by the specified optional content configuration dictionary.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetOptionalContentConfigLocked(
  OptionalContentConfigID, OptionalContentGroupID,
  NewLocked: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetOptionalContentConfigLocked(
  OptionalContentConfigID As Long,
  OptionalContentGroupID As Long, NewLocked As Long) As Long
```

DLL

```
int DPLSetOptionalContentConfigLocked(int InstanceID,
  int OptionalContentConfigID, int OptionalContentGroupID,
  int NewLocked);
```

Parameters

OptionalContentConfigID	The first default optional content configuration dictionary has an ID of 1. Higher numbers are used for other optional content configuration dictionaries.
OptionalContentGroupID	An ID returned by the **NewOptionalContentGroup**, **GetOptionalContentGroupID** or **GetOptionalContentConfigOrderItemID** functions
NewLocked	0 = Unlocked 1 = Locked

Return values

0	The optional content group could not be locked
1	Success

SetOptionalContentConfigState

Content Streams and Optional Content Groups

Version history

This function was introduced in Quick PDF Library version 8.12.

Description

This function is used to set the state of an optional content group as defined by the specified optional content configuration dictionary.

All optional content configuration dictionaries have a base state (either ON, OFF or Unchanged) and two membership arrays called /ON and /OFF.

A reference to the optional content group is added to the appropriate /ON or /OFF array (or removed from either array) depending on the value of the base state.

A particular optional content group can only be set to Unchanged if the base state of the optional content configuration dictionary is Unchanged.

The base state of the default optional content configuration dictionary (accessed by setting OptionalContentConfigID to 1) is always ON, so optional content groups in this configuration dictionary can only be set to ON or OFF.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetOptionalContentConfigState(
  OptionalContentConfigID, OptionalContentGroupID,
  NewState: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetOptionalContentConfigState(
  OptionalContentConfigID As Long,
  OptionalContentGroupID As Long, NewState As Long) As Long
```

DLL

```
int DPLSetOptionalContentConfigState(int InstanceID,
  int OptionalContentConfigID, int OptionalContentGroupID,
  int NewState);
```

Parameters

OptionalContentConfigID	The first default optional content configuration dictionary has an ID of 1. Higher numbers are used for other optional content configuration dictionaries.
OptionalContentGroupID	An ID returned by the **NewOptionalContentGroup**, **GetOptionalContentGroupID** or **GetOptionalContentConfigOrderItemID** functions
NewState	Specifies the state that the optional content group in this configuration dictionary should be set to: 1 = Set to ON 2 = Set to OFF 3 = Set to unchanged (if possible)

Return values

0	The state could not be set
1	The state was set successfully

SetOptionalContentGroupName

Version history

This function was introduced in Quick PDF Library version 9.12.

Description

Sets the name of the specified optional content group.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetOptionalContentGroupName(
  OptionalContentGroupID: Integer; NewGroupName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetOptionalContentGroupName(
  OptionalContentGroupID As Long, NewGroupName As String) As Long
```

DLL

```
int DPLSetOptionalContentGroupName(int InstanceID,
  int OptionalContentGroupID, wchar_t * NewGroupName);
```

Parameters

OptionalContentGroupID	An ID returned by the **NewOptionalContentGroup**, **GetOptionalContentGroupID** or **GetOptionalContentConfigOrderItemID** functions
NewGroupName	The new name for the OCG

Return values

0	The name could not be set
1	The OCG's name was set successfully

SetOptionalContentGroupPrintable

Content Streams and Optional Content Groups

Version history

This function was introduced in Quick PDF Library version 7.12.

Description

This function allows an optional content group to be marked as visible or invisible when the document is printed.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetOptionalContentGroupPrintable(
  OptionalContentGroupID, Printable: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetOptionalContentGroupPrintable(
  OptionalContentGroupID As Long, Printable As Long) As Long
```

DLL

```
int DPLSetOptionalContentGroupPrintable(int InstanceID,
  int OptionalContentGroupID, int Printable);
```

Parameters

OptionalContentGroupID	An ID returned by the **NewOptionalContentGroup**, **GetOptionalContentGroupID** or **GetOptionalContentConfigOrderItemID** functions
Printable	0 = Not printed 1 = Printed

SetOptionalContentGroupVisible

Content Streams and Optional Content Groups

Description

This function allows an optional content group to be marked as visible or invisible when the document is opened.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetOptionalContentGroupVisible(
  OptionalContentGroupID, Visible: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetOptionalContentGroupVisible(
  OptionalContentGroupID As Long, Visible As Long) As Long
```

DLL

```
int DPLSetOptionalContentGroupVisible(int InstanceID,
  int OptionalContentGroupID, int Visible);
```

Parameters

OptionalContentGroupID	An ID returned by the **NewOptionalContentGroup**, **GetOptionalContentGroupID** or **GetOptionalContentConfigOrderItemID** functions
Visible	0 = Not visible 1 = Visible

Return values

Non-zero	An ID that can be used as the OptionalContentGroupID parameter with the other optional content group functions

SetOrigin

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Sets the origin for all subsequent drawing operations.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetOrigin(Origin: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetOrigin(Origin As Long) As Long
```

DLL

```
int DPLSetOrigin(int InstanceID, int Origin);
```

Parameters

Origin	Specifies which page corner to use for the origin: 0 = Bottom left (default) 1 = Top left 2 = Top right 3 = Bottom right Anything else = Bottom left

SetOutlineColor

Description

Sets the color of an outline item (bookmark). The values of the color parameters range from 0 to 1, with 0 indicating 0% and 1 indicating 100% of the color.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetOutlineColor(OutlineID: Integer; Red,
  Green, Blue: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetOutlineColor(
  OutlineID As Long, Red As Double, Green As Double,
  Blue As Double) As Long
```

DLL

```
int DPLSetOutlineColor(int InstanceID, int OutlineID, double Red,
  double Green, double Blue);
```

Parameters

OutlineID	The ID of the outline as returned by the **NewOutline** function. Alternatively, use the **GetOutlineID** function to get a valid outline ID.
Red	The red component of the color
Green	The green component of the color
Blue	The blue component of the color

Return values

0	The Outline ID provided was invalid
1	The color of the outline item was set successfully

SetOutlineDestination

Description

Sets the destination that an outline item (bookmark) points to.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetOutlineDestination(OutlineID,
  DestPage: Integer; DestPosition: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetOutlineDestination(
  OutlineID As Long, DestPage As Long,
  DestPosition As Double) As Long
```

DLL

```
int DPLSetOutlineDestination(int InstanceID, int OutlineID, int DestPage,
  double DestPosition);
```

Parameters

OutlineID	The ID of the outline as returned by the **NewOutline** function. Alternatively, use the **GetOutlineID** function to get a valid outline ID.
DestPage	The page number that this outline item links to
DestPosition	The vertical position of the page that this outline item links to. Jumping to the bottom of the page will result in the following page being shown. If possible link to the top of the page.

Return values

0	The Outline ID provided was invalid
1	The destination of the outline item was set successfully

SetOutlineDestinationFull

Version history

This function was introduced in Quick PDF Library version 7.12.

Description

Sets the destination of an outline item (bookmark) to a specific postion and zoom percentage.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetOutlineDestinationFull(OutlineID,
  DestPage, Zoom, DestType: Integer; Left, Top, Right,
  Bottom: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetOutlineDestinationFull(
  OutlineID As Long, DestPage As Long, Zoom As Long,
  DestType As Long, Left As Double, Top As Double,
  Right As Double, Bottom As Double) As Long
```

DLL

```
int DPLSetOutlineDestinationFull(int InstanceID, int OutlineID,
  int DestPage, int Zoom, int DestType, double Left, double Top,
  double Right, double Bottom);
```

Parameters

OutlineID	The ID of the outline as returned by the **NewOutline** function. Alternatively, use the **GetOutlineID** function to get a valid outline ID.
DestPage	The page number that this outline item links to
Zoom	The zoom percentage to use when the outline destination is opened, valid values from 0 to 6400. Only used for DestType = 1, should be set to 0 for other DestTypes.
DestType	1 = "XYZ" - the target page is positioned at the point specified by the Left and Top parameters. The Zoom parameter specifies the zoom percentage. 2 = "Fit" - the entire page is zoomed to fit the window. None of the other parameters are used and should be set to zero. 3 = "FitH" - the page is zoomed so that the entire width of the page is visible. The height of the page may be greater or less than the height of the window. The page is positioned at the vertical position specified by the Top parameter. 4 = "FitV" - the page is zoomed so that the entire height of the page can be seen. The width of the page may be greater or less than the width of the window. The page is positioned at the horizontal position specified by the Left parameter. 5 = "FitR" - the page is zoomed so that a certain rectangle on the page is visible. The Left, Top, Right and Bottom parameters define the rectangular area on the page. 6 = "FitB" - the page is zoomed so that it's bounding box is visible. 7 = "FitBH" - the page is positioned vertically at the position specified by the Top parameter. The page is zoomed so that the entire width of the page's bounding box is visible. 8 = "FitBV" - the page is positioned at the horizontal position specified by the Left parameter. The page is zoomed just enough to fit the entire height of the bounding box into the window.
Left	The horizontal position used by DestType = 1, 4, 5 and 8
Top	The vertical position used by DestType = 1, 3, 5 and 7
Right	The horizontal position of the righthand edge of the rectangle. Used by DestType = 5
Bottom	The horizontal position of the bottom of the rectangle. Used by DestType = 5

Return values

0	The outline destination could not be set. The usually indicates that the Zoom or DestType parameters are out of range or the OutlineID is invalid.
1	The outline destination was set successfully

SetOutlineDestinationZoom

Outlines

Description

Sets the destination of an outline item (bookmark) to a specific postion on a page, and sets the zoom percentage of the document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetOutlineDestinationZoom(OutlineID,
  DestPage: Integer; DestPosition: Double; Zoom: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetOutlineDestinationZoom(
  OutlineID As Long, DestPage As Long, DestPosition As Double,
  Zoom As Long) As Long
```

DLL

```
int DPLSetOutlineDestinationZoom(int InstanceID, int OutlineID,
  int DestPage, double DestPosition, int Zoom);
```

Parameters

OutlineID	The ID of the outline as returned by the **NewOutline** function. Alternatively, use the **GetOutlineID** function to get a valid outline ID.
DestPage	The page number that this outline should link to
DestPosition	The vertical position on the page that the outline should link to. Specifying a point at the bottom of the page will result in the next page being shown - it is better to link to a point at the top of the page.
Zoom	The zoom factor to show the target page at: 0..1600 = Zoom percentage -1 = Fit in window -2 = Fit width

Return values

0	The OutlineID was invalid
1	Destination set successfull

SetOutlineJavaScript

JavaScript, Outlines

Description

Specifies the JavaScript to run when the user clicks on the outline item (bookmark).

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetOutlineJavaScript(OutlineID: Integer;
  JavaScript: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetOutlineJavaScript(
  OutlineID As Long, JavaScript As String) As Long
```

DLL

```
int DPLSetOutlineJavaScript(int InstanceID, int OutlineID,
  wchar_t * JavaScript);
```

Parameters

OutlineID	The ID of the outline as returned by the **NewOutline** function. Alternatively, use the **GetOutlineID** function to get a valid outline ID.
JavaScript	The JavaScript to execute.

Return values

0	The OutlineID was invalid
1	The JavaScript action was successfully added to the outline ID

SetOutlineNamedDestination

Annotations and hotspot links, Outlines

Version history

This function was introduced in Quick PDF Library version 7.22.

Description

Sets the destination of the specified outline item (bookmark) to a named destination.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetOutlineNamedDestination(
  OutlineID: Integer; DestName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetOutlineNamedDestination(
  OutlineID As Long, DestName As String) As Long
```

DLL

```
int DPLSetOutlineNamedDestination(int InstanceID, int OutlineID,
  wchar_t * DestName);
```

Parameters

OutlineID	The ID of the outline as returned by the **NewOutline** function. Alternatively, use the **GetOutlineID** function to get a valid outline ID.
DestName	The named destination.

Return values

0	The OutlineID was invalid
1	Success

SetOutlineOpenFile

Description

Sets the outline item (bookmark) to open a file when it is clicked.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetOutlineOpenFile(OutlineID: Integer;
  FileName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetOutlineOpenFile(
  OutlineID As Long, FileName As String) As Long
```

DLL

```
int DPLSetOutlineOpenFile(int InstanceID, int OutlineID,
  wchar_t * FileName);
```

Parameters

OutlineID	The ID of the outline as returned by the **NewOutline** function. Alternatively, use the **GetOutlineID** function to get a valid outline ID.
FileName	The file to open when the outline is clicked. This should be in a specific format. Back slashes should be converted to forward slashes and the drive, if any, should be specified as just the drive letter between forward slashes without a colon. For example, the file "c:\my documents\hello.pdf" should be specified as "/c/my documents/hello.pdf". Relative path names are valid, including paths that include the ".." operator to move up a directory.

Return values

0	The OutlineID was invalid
1	The outline destination was set successfully

SetOutlineRemoteDestination

Outlines

DEBENU
QUICKPDF
LIBRARY

Version history

> This function was introduced in Quick PDF Library version 7.22.

Description

> Sets the outline item (bookmark) to open another PDF when it is clicked.
> The opening page number and various sizing/positioning values can be specified.
> Note: because the page size of the target document is not known, all positions are specified in points measured from the bottom left corner of the opening page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetOutlineRemoteDestination(
    OutlineID: Integer; FileName: WideString; OpenPage, Zoom,
    DestType: Integer; PntLeft, PntTop, PntRight, PntBottom: Double;
    NewWindow: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetOutlineRemoteDestination(
    OutlineID As Long, FileName As String, OpenPage As Long,
    Zoom As Long, DestType As Long, PntLeft As Double,
    PntTop As Double, PntRight As Double, PntBottom As Double,
    NewWindow As Long) As Long
```

DLL

```
int DPLSetOutlineRemoteDestination(int InstanceID, int OutlineID,
    wchar_t * FileName, int OpenPage, int Zoom, int DestType,
    double PntLeft, double PntTop, double PntRight,
    double PntBottom, int NewWindow);
```

Parameters

OutlineID	The ID of the outline as returned by the **NewOutline** function. Alternatively, use the **GetOutlineID** function to get a valid outline ID.
FileName	The filename of the PDF document to open when the outline is clicked. This should be in a specific format. Back slashes should be converted to forward slashes and the drive, if any, should be specified as just the drive letter between forward slashes without a colon. For example, the file "c:\my documents\hello.pdf" should be specified as "/c/my documents/hello.pdf". Relative path names are valid, including paths that include the ".." operator to move up a directory.
OpenPage	The page number to jump to when the target document is opened. The first page has an index of zero (0).
Zoom	The zoom percentage to use when the document is opened, valid values from 0 to 6400. Only used for DestType = 1, should be set to 0 for other DestTypes.
DestType	1 = "XYZ" - the target page is positioned at the point specified by the Left and Top parameters. The Zoom parameter specifies the zoom percentage. 2 = "Fit" - the entire page is zoomed to fit the window. None of the other parameters are used and should be set to zero. 3 = "FitH" - the page is zoomed so that the entire width of the page is visible. The height of the page may be greater or less than the height of the window. The page is positioned at the vertical position specified by the Top parameter. 4 = "FitV" - the page is zoomed so that the entire height of the page can be seen. The width of the page may be greater or less than the width of the window. The page is positioned at the horizontal position specified by the Left parameter. 5 = "FitR" - the page is zoomed so that a certain rectangle on the page is visible. The Left, Top, Right and Bottom parameters define the rectangular area on the page. 6 = "FitB" - the page is zoomed so that it's bounding box is visible. 7 = "FitBH" - the page is positioned vertically at the position specified by the Top parameter. The page is zoomed so that the entire width of the page's bounding box is visible. 8 = "FitBV" - the page is positioned at the horizontal position specified by the Left parameter. The page is zoomed just enough to fit the entire height of the bounding box into the window.
PntLeft	The horizontal position used by DestType = 1, 4, 5 and 8. The position is specified in points measured from the bottom left corner of the target document's page.
PntTop	The vertical position used by DestType = 1, 3, 5 and 7. The position is specified in points measured from the bottom left corner of the target document's page.
PntRight	The horizontal position of the righthand edge of the rectangle. Used by DestType = 5. The position is specified in points measured from the bottom left corner of the target document's page.
PntBottom	The horizontal position of the bottom of the rectangle. Used by DestType = 5. The position is specified in points measured from the bottom left corner of the target document's page.
NewWindow	0 = Replace the current document with the target document 1 = Open the target document in a new window unless the user has specified a different preference in the PDF viewer

Return values

0	The OutlineID was invalid
1	The outline destination was set successfully

SetOutlineStyle

Description

Sets the way an outline item (bookmark) is displayed.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetOutlineStyle(OutlineID, SetItalic,
  SetBold: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetOutlineStyle(
  OutlineID As Long, SetItalic As Long, SetBold As Long) As Long
```

DLL

```
int DPLSetOutlineStyle(int InstanceID, int OutlineID, int SetItalic,
  int SetBold);
```

Parameters

OutlineID	The ID of the outline as returned by the **NewOutline** function. Alternatively, use the **GetOutlineID** function to get a valid outline ID.
SetItalic	0 = Normal 1 = Italic
SetBold	0 = Normal 1 = Bold

Return values

0	The Outline ID provided was invalid
1	The style of the outline item was set successfully

SetOutlineTitle

Description

Sets the title of an outline item (bookmark).

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetOutlineTitle(OutlineID: Integer;
  NewTitle: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetOutlineTitle(
  OutlineID As Long, NewTitle As String) As Long
```

DLL

```
int DPLSetOutlineTitle(int InstanceID, int OutlineID, wchar_t * NewTitle);
```

Parameters

OutlineID	The ID of the outline as returned by the **NewOutline** function. Alternatively, use the **GetOutlineID** function to get a valid outline ID.
NewTitle	The new title for the outline item.

Return values

0	The Outline ID provided was invalid
1	The title of the outline item was set successfully

SetOutlineWebLink

Description

Specifies an internet link that should be opened when the user clicks on the outline item (bookmark).

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetOutlineWebLink(OutlineID: Integer;
  Link: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetOutlineWebLink(
  OutlineID As Long, Link As String) As Long
```

DLL

```
int DPLSetOutlineWebLink(int InstanceID, int OutlineID, wchar_t * Link);
```

Parameters

OutlineID	The ID of the outline as returned by the **NewOutline** function. Alternatively, use the **GetOutlineID** function to get a valid outline ID.
Link	The URL to link to. Some examples: "http://www.example.com/" "mailto:info@example.com"

Return values

0	The OutlineID was invalid
1	The web link action was added to the outline item successfully

SetOverprint

Version history

This function was introduced in Quick PDF Library version 7.22.

Description

Sets the overprint parameter of the graphics state for subsequently drawn text and graphics.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetOverprint(StrokingOverprint,
  OtherOverprint, OverprintMode: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetOverprint(
  StrokingOverprint As Long, OtherOverprint As Long,
  OverprintMode As Long) As Long
```

DLL

```
int DPLSetOverprint(int InstanceID, int StrokingOverprint,
  int OtherOverprint, int OverprintMode);
```

Parameters

StrokingOverprint	Controls overprint for stroking operations: 0 = Turn overprint off 1 = Turn overprint on
OtherOverprint	Controls overprint for non-stroking operations: 0 = Turn overprint off 1 = Turn overprint on
OverprintMode	Sets the interpretation of a tint value of 0.0 for a color component in a DeviceCMYK colour space. 0 = Default behaviour 1 = Nonzero overprint mode

Return values

0	An error occurred. One or more of the parameters were out of range.
1	Success

SetPDFAMode

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Sets up the document for PDF/A standards compliance mode.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetPDFAMode(NewMode: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetPDFAMode(
  NewMode As Long) As Long
```

DLL

```
int DPLSetPDFAMode(int InstanceID, int NewMode);
```

Parameters

NewMode	2 = PDF/A-1b

Return values

0	Invalid NewMode parameter
1	The compliance mode was set successfully

SetPNGTransparencyColor

Image handling, Color

Version history

This function was introduced in Quick PDF Library version 7.11.

Description

Sets the RGB color to use as the transparency mask in PNG images that are generated by the rendering functions.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetPNGTransparencyColor(RedByte, GreenByte,
  BlueByte: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetPNGTransparencyColor(
  RedByte As Long, GreenByte As Long, BlueByte As Long) As Long
```

DLL

```
int DPLSetPNGTransparencyColor(int InstanceID, int RedByte, int GreenByte,
  int BlueByte);
```

Parameters

RedByte	The red component
GreenByte	The green component
BlueByte	The blue component

SetPageActionMenu

Description

Specifies a menu item to run when the document is first opened.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetPageActionMenu(
  MenuItem: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetPageActionMenu(
  MenuItem As String) As Long
```

DLL

```
int DPLSetPageActionMenu(int InstanceID, wchar_t * MenuItem);
```

Parameters

MenuItem	The MenuItem to call, for example "Print"

Return values

0	The open action could not be set, there is a problem with the document
1	The page open action was set successfully

SetPageBox

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Version history

This function was introduced in Quick PDF Library version 7.18.

Description

Sets the dimensions of the selected page's boundary rectangles.

The MediaBox represents the physical medium of the page.

The CropBox represents the visible region of the page, the contents will be clipped to this region.

The BleedBox is similar to the CropBox, but is the rectangle used in a production environment.

The TrimBox indicates the intended dimensions of the finished page after trimming, and the ArtBox defines the extent of the page's meaningful content as intended by the page's creator.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetPageBox(BoxType: Integer; Left, Top,
  Width, Height: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetPageBox(BoxType As Long,
  Left As Double, Top As Double, Width As Double,
  Height As Double) As Long
```

DLL

```
int DPLSetPageBox(int InstanceID, int BoxType, double Left, double Top,
  double Width, double Height);
```

Parameters

BoxType	1 = MediaBox 2 = CropBox 3 = BleedBox 4 = TrimBox 5 = ArtBox
Left	The horizontal co-ordinate of the left edge of the rectangle
Top	The vertical co-ordinate of the top edge of the rectangle
Width	The width of the rectangle
Height	The height of the rectangle

SetPageContentFromString

Version history

This function was renamed in Quick PDF Library version 8.11.
The function name in earlier versions was SetPageContent.

Description

This function allows the content of the selected PDF page to be set. This is for advanced use only! If incorrect information is put into the page's content stream then the PDF file may not load correctly with Acrobat or any other PDF viewer.

In previous versions of Quick PDF Library this function would only set the content of the selected content stream part.

From version 8.11 this function sets the content of the entire page resulting in a single content stream part. The **SetContentStreamFromString** function can be used to set the PDF page description commands of the content stream part selected with the **SelectContentStream** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetPageContentFromString(
  const Source: AnsiString): Integer;
```

DLL

```
int DPLSetPageContentFromString(int InstanceID, char * Source);
```

Parameters

Source	The new contents of the page

SetPageContentFromVariant

Page properties, Page layout, Page manipulation

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

This function allows the content of the selected PDF page to be set. This is for advanced use only! If incorrect information is put into the page's content stream then the PDF file may not load correctly with Acrobat or any other PDF viewer.

This function sets the content of the entire page resulting in a single content stream part.

Syntax

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetPageContentFromVariant(
  NewValue As Variant) As Long
```

Parameters

NewValue   The new contents of the page as a byte array variant

SetPageDimensions

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Set the size of the selected page.

This function updates the MediaBox entry which represents the physical medium of the page and will only affect content subsequently added to the page. This function does not resize the already existing content of the page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetPageDimensions(NewPageWidth,
   NewPageHeight: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetPageDimensions(
   NewPageWidth As Double, NewPageHeight As Double) As Long
```

DLL

```
int DPLSetPageDimensions(int InstanceID, double NewPageWidth,
   double NewPageHeight);
```

Parameters

NewPageWidth	The new width of the page
NewPageHeight	The new height of the page

Return values

0	The page size could not be set. This should never occur.
1	The page was resized successfully

SetPageLayout

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Sets the initial page layout of the document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetPageLayout(
  NewPageLayout: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetPageLayout(
  NewPageLayout As Long) As Long
```

DLL

```
int DPLSetPageLayout(int InstanceID, int NewPageLayout);
```

Parameters

NewPageLayout	0 = Single page
	1 = One column
	2 = Two columns, odd-numbered pages on left
	3 = Two columns, odd-numbered pages on right
	4 = Two pages, odd-numbered pages on left
	5 = Two pages, odd-numbered pages on right
	6 = No preference (setting removed from document)

Return values

0	The page layout could not be set
1	The page layout was set successfully

SetPageMode

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Sets the initial page mode of the document.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetPageMode(NewPageMode: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetPageMode(
  NewPageMode As Long) As Long
```

DLL

```
int DPLSetPageMode(int InstanceID, int NewPageMode);
```

Parameters

NewPageMode	0 = Normal view 1 = Show the outlines pane 2 = Show the thumbnails pane 3 = Show the document in full screen mode 4 = Optional content group panel visible 5 = Attachments panel visible

Return values

0	The page mode could not be set
1	The page mode was set successfully

SetPageSize

Page properties, Page layout

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Use this function to set the current page to a named size, for example "A4" or "Letter Landscape".

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetPageSize(PaperName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetPageSize(
  PaperName As String) As Long
```

DLL

```
int DPLSetPageSize(int InstanceID, wchar_t * PaperName);
```

Parameters

PaperName	The name of the paper, one of the following: A0 to A10, B0 to B10, ISOB0 to ISOB10, C0 to C7, DL, Letter, Legal, Statement, Tabloid, Ledger, Executive, Folio. You can make a landscape page by adding the word Landscape after the paper name, for example "A3 Landscape".

Return values

0	The specified paper name was not valid
1	The page was resized successfully

SetPageThumbnail

Description

Sets the selected image as the "thumbnail" for the selected page.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetPageThumbnail: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetPageThumbnail As Long
```

DLL

```
int DPLSetPageThumbnail(int InstanceID);
```

Return values

0	No image was selected
1	The thumbnail was set successfully

SetPageTransparencyGroup

Vector graphics, Text, Page layout

Description

Allows the transparency group for the page to be set. Whenever image are used as masks for other images the page transparency group should be set to ensure consistent results across different versions of PDF viewers.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetPageTransparencyGroup(CS, Isolate,
  Knockout: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetPageTransparencyGroup(
  CS As Long, Isolate As Long, Knockout As Long) As Long
```

DLL

```
int DPLSetPageTransparencyGroup(int InstanceID, int CS, int Isolate,
  int Knockout);
```

Parameters

CS	The color space to use: 1 = RGB 2 = CMYK
Isolate	This parameter has no effect and is reserved for future use. It should always be set to 0.
Knockout	Indicates whether items added to the page are drawn over each other or "knocked out" of the page. In knockout mode a "hole" is made through existing objects on the page in the shape of the new object. The new object is then drawn against the background. 0 = Do not knockout 1 = Knockout

SetPageUserUnit

Version history

This function was introduced in Quick PDF Library version 10.15.

Description

Applies a scaling factor to the PDF to allow pages size of larger than 200x200 inches to be defined. SetPageDimensions allows a maximum of 14040x14400 units to be define and this is still the case in PDF 1.6 and above. PDF 1.6 and above allow chaning the UserUnit for 1/72" to a much larger value. ie SerPageUserUnit(2); would scale the page maximum size to 400x400" where in point would actually scale to 2 points.

If you need to use this function then you should make sure QP.SetInformation(0, "1.6"); to set te PDF verision to at least version 1.6.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetPageUserUnit(UserUnit: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetPageUserUnit(
  UserUnit As Double) As Long
```

DLL

```
int DPLSetPageUserUnit(int InstanceID, double UserUnit);
```

Parameters

UserUnit	The scale factor to apply. Default for all PDF's is 1.0.

SetPrecision

Description

Use this function to set the precision of numerical values stored in the PDF document. Setting the precision to a lower number will reduce the size of the generated file, while a higher precision will result in a larger file, although objects and graphics will be more accurately positioned. The default precision is 4.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetPrecision(
  NewPrecision: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetPrecision(
  NewPrecision As Long) As Long
```

DLL

```
int DPLSetPrecision(int InstanceID, int NewPrecision);
```

Parameters

NewPrecision	The precision to use for subsequent drawing operations. A value from 2 to 8.

Return values

0	The precision specified was out of range
1	The precision was set successfully

SetPrinterDevModeFromString

Version history

This function was introduced in Quick PDF Library version 8.12.

Description

Sets the printer DEVMODE structure for the next printing operation using the value retrieved from a prior call to **GetPrinterDevModeToString**.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetPrinterDevModeFromString(
  const Source: AnsiString): Integer;
```

DLL

```
int DPLSetPrinterDevModeFromString(int InstanceID, char * Source);
```

Parameters

Source	A value returned from the **GetPrinterDevModeToString** function.

SetPrinterDevModeFromVariant

Rendering and printing

Version history

This function was introduced in Quick PDF Library version 8.12.

Description

Sets the printer DEVMODE structure for the next printing operation using the value retrieved from a prior call to **GetPrinterDevModeToVariant**.

Syntax

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetPrinterDevModeFromVariant(
  Source As Variant) As Long
```

Parameters

Source   A value returned from the **GetPrinterDevModeToVariant** function.

SetRenderCropType

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Sets the page boundary to use as the cropping area for rendering.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetRenderCropType(
  NewCropType: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetRenderCropType(
  NewCropType As Long) As Long
```

DLL

```
int DPLSetRenderCropType(int InstanceID, int NewCropType);
```

Parameters

NewCropType	1 = MediaBox 2 = CropBox 3 = BleedBox 4 = TrimBox 5 = ArtBox

Return values

0	The NewCropType parameter was out of range.
1	The rendering crop type was set successfully.

SetRenderDCErasePage

Rendering and printing

Version history

This function was introduced in Quick PDF Library version 7.25.

Description

By default the **RenderPageToDC** and **DARenderPageToDC** functions fill the page area with solid white background before rendering the page contents.

This function can be used to suppress the background allowing the page contents to be drawn over any existing content in the supplied device context.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetRenderDCErasePage(
  NewErasePage: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetRenderDCErasePage(
  NewErasePage As Long) As Long
```

DLL

```
int DPLSetRenderDCErasePage(int InstanceID, int NewErasePage);
```

Parameters

NewErasePage	0 = No page background is drawn 1 = The page area is filled with a solid white background before rendering

SetRenderDCOffset

Rendering and printing

Version history

This function was introduced in Quick PDF Library version 7.22.

Description

Sets the position on the device context that the **RenderPageToDC** and **DARenderPageToDC** functions use for the top-left corner of the rendered output.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetRenderDCOffset(NewOffsetX,
  NewOffsetY: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetRenderDCOffset(
  NewOffsetX As Long, NewOffsetY As Long) As Long
```

DLL

```
int DPLSetRenderDCOffset(int InstanceID, int NewOffsetX, int NewOffsetY);
```

Parameters

NewOffsetX	The horizontal offset measured in pixels
NewOffsetY	The vertical offset measured in pixels

SetRenderOptions

Rendering and printing

Version history

This function was introduced in Quick PDF Library version 9.15.

Description

Sets various options for the renderer.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetRenderOptions(OptionID,
  NewValue: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetRenderOptions(
  OptionID As Long, NewValue As Long) As Long
```

DLL

```
int DPLSetRenderOptions(int InstanceID, int OptionID, int NewValue);
```

Parameters

OptionID	1 = Render Formfields 2 = Render Annotations 3 = Render Formfields only 4 = Gamma Correction 5 = ICCBased colorspaces 6 = Progress HWND 7 = Progress Message 8 = Progress Data 9 = Path combine mode
NewValue	For RenderFormFields: 0 = Don't render fomfields 1 = Render formfields (default) For RenderAnnotations: 0 = Don't render annotations 1 = Render annotations(default) For RenderFormFieldsOnly: 0 = Render the page including formfields (default) 1 = Only render the formfields For UseGammaCorrection: 0 = Turn off CMYK gamma correction 1 = Use CMYK Gamma correction (default) For Ignore ICCBased colorspaces: 0 = Render using ICCBased colorspaces 1 = Ignore ICCBased colorspace corrections For progress options: Reserved for future use For path combine mode: 0 = Normal (no path combining) 1 = Combine paths

SetRenderScale

Version history

This function was introduced in Quick PDF Library version 7.22.

Description

Applies a non-integer scaling to the DPI parameter of subsequent calls to any of the rendering functions.

For example, if the render scale is set to 0.1 and the **RenderPageToFile** function is called with the DPI parameter set to 125, the resulting image will be rendered with an effective DPI of 12.5.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetRenderScale(NewScale: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetRenderScale(
  NewScale As Double) As Long
```

DLL

```
int DPLSetRenderScale(int InstanceID, double NewScale);
```

Parameters

NewScale   The new render scale

SetScale

Description

Scales the co-ordinate system for all subsequent drawing operations. A scale factor of 1 is equivalent to calling **SetMeasurementUnits**(0) which sets the measurement units to be Points. A scale factor of (72 / 25.4) is equivalent to calling **SetMeasurementUnits**(1) which sets the measurement units to be millimetres.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetScale(NewScale: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetScale(
  NewScale As Double) As Long
```

DLL

```
int DPLSetScale(int InstanceID, double NewScale);
```

Parameters

NewScale   The scale factor to use

SetSignProcessCustomDict

Version history

This function was introduced in Quick PDF Library version 11.15.

Description

Adds a value to a custom dictionary in the digital signature object

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetSignProcessCustomDict(
  SignProcessID: Integer; Key, NewValue: WideString;
  StorageType: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetSignProcessCustomDict(
  SignProcessID As Long, Key As String, NewValue As String,
  StorageType As Long) As Long
```

DLL

```
int DPLSetSignProcessCustomDict(int InstanceID, int SignProcessID,
  wchar_t * Key, wchar_t * NewValue, int StorageType);
```

Parameters

SignProcessID	A value returned by the **NewSignProcessFromFile**, **NewSignProcessFromStream** or **NewSignProcessFromString** functions.
Key	The key for the value. Nested dictionaries can be created by separating keys with a forward slash, eg. "Key1/Key2/Key3"
NewValue	The value to assign to the key
StorageType	0 = Unicode string 1 = Hex data, stored as hex 2 = Hex data, stored as literal 3 = Name 4 = Numeric 5 = Boolean ("true" or "false")

SetSignProcessCustomSubFilter

Version history

This function was introduced in Quick PDF Library version 11.11.

Description

Sets the SubFilter entry with custom string for a digital signature process, specifying the encoding of the signature value.
Similar to **SetSignProcessSubFilter** but the Subfilter entry is customizable

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetSignProcessCustomSubFilter(
  SignProcessID: Integer; SubFilterStr: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetSignProcessCustomSubFilter(
  SignProcessID As Long, SubFilterStr As String) As Long
```

DLL

```
int DPLSetSignProcessCustomSubFilter(int InstanceID, int SignProcessID,
  wchar_t * SubFilterStr);
```

Parameters

SignProcessID	A value returned by the **NewSignProcessFromFile**, **NewSignProcessFromStream** or **NewSignProcessFromString** functions.
SubFilterStr	Custom SubFilter string entry

Return values

0	The SignProcessID parameter was invalid or the SubFilter parameter was out of range
1	Success

SetSignProcessField

Version history

This function was introduced in Quick PDF Library version 9.13.

Description

Sets the signature field to use for a digital signature process.

If a field with a specified name is not found a new signature field will be added with the given name. The new field will be invisible (zero width and height) and will be attached to the first page in the document. Call **SetSignProcessFieldBounds** to set location and size of new form field and **SetSignProcessFieldPage** to set the page it is placed on.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetSignProcessField(SignProcessID: Integer;
  SignatureFieldName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetSignProcessField(
  SignProcessID As Long, SignatureFieldName As String) As Long
```

DLL

```
int DPLSetSignProcessField(int InstanceID, int SignProcessID,
  wchar_t * SignatureFieldName);
```

Parameters

SignProcessID	A value returned by the **NewSignProcessFromFile**, **NewSignProcessFromStream** or **NewSignProcessFromString** functions.
SignatureFieldName	The name of the signature form field

SetSignProcessFieldBounds

Version history

This function was introduced in Quick PDF Library version 9.14.

Description

Sets the location and size of the signature field in the specified digital signature process.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetSignProcessFieldBounds(
  SignProcessID: Integer; Left, Top, Width, Height: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetSignProcessFieldBounds(
  SignProcessID As Long, Left As Double, Top As Double,
  Width As Double, Height As Double) As Long
```

DLL

```
int DPLSetSignProcessFieldBounds(int InstanceID, int SignProcessID,
  double Left, double Top, double Width, double Height);
```

Parameters

SignProcessID	A value returned by the **NewSignProcessFromFile**, **NewSignProcessFromStream** or **NewSignProcessFromString** functions.
Left	The horizontal coordinate of the left edge of the area measured in points from the left edge of the media box.
Top	The vertical coordinate of the top edge of the area measured in points from the bottom edge of the media box.
Width	The width of the area measured in points.
Height	The height of the area measured in points.

Return values

0	Invalid SignProcessID parameter
1	Success

SetSignProcessFieldImageFromFile

Security and Signatures

Version history

This function was introduced in Quick PDF Library version 9.14.

Description

Sets the image file to use for a visual signature field in the specified digital signature process.

The **SetSignProcessFieldBounds** function can be used to specify the location and size of the signature field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetSignProcessFieldImageFromFile(
  SignProcessID: Integer; ImageFileName: WideString;
  Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetSignProcessFieldImageFromFile(
  SignProcessID As Long, ImageFileName As String,
  Options As Long) As Long
```

DLL

```
int DPLSetSignProcessFieldImageFromFile(int InstanceID, int SignProcessID,
  wchar_t * ImageFileName, int Options);
```

Parameters

SignProcessID	A value returned by the **NewSignProcessFromFile**, **NewSignProcessFromStream** or **NewSignProcessFromString** functions.
ImageFileName	The path and file name of the image to use for the visual signature.
Options	For multi-page TIFF images this parameter specifies the page number to load. For PNG images: 0 = Load the image as usual 1 = Load the alpha channel as a greyscale image 2 = Load the image and alpha channel (limit alpha to 8-bit) 3 = Load the image (limit image 8-bit/channel) 4 = Load the alpha channel (limit to 8-bit/channel) 5 = Load the image with alpha channel (limit both to 8-bit/channel) 6 = Load the image and alpha channel 7 = Load the image and ICC color profile For other image types this parameter should be set to 0. Setting Options to -1 forces TIFF, EMF and WMF images to be loaded using the GDI+ graphics library. Multipage TIFF images can also be loaded using GDI+ by setting the Options parameter to -PageNumber (for example -3 for page 3).

Return values

0	Image could not be added
1	Success

SetSignProcessFieldImageFromStream

Version history

This function was introduced in Quick PDF Library version 11.15.

Description

Sets the image data to use for a visual signature field in the specified digital signature process.

The **SetSignProcessFieldBounds** function can be used to specify the location and size of the signature field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetSignProcessFieldImageFromStream(
  SignProcessID: Integer; InputStream: TStream; Options: Integer): Integer;
```

Parameters

SignProcessID	A value returned by the **NewSignProcessFromFile**, **NewSignProcessFromStream** or **NewSignProcessFromString** functions.
InputStream	A stream containing the image data
Options	For multi-page TIFF images this parameter specifies the page number to load.
	For PNG images: 0 = Load the image as usual 1 = Load the alpha channel as a greyscale image 2 = Load the image and alpha channel (limit alpha to 8-bit) 3 = Load the image (limit image 8-bit/channel) 4 = Load the alpha channel (limit to 8-bit/channel) 5 = Load the image with alpha channel (limit both to 8-bit/channel) 6 = Load the image and alpha channel 7 = Load the image and ICC color profile
	For other image types this parameter should be set to 0.
	Setting Options to -1 forces TIFF, EMF and WMF images to be loaded using the GDI+ graphics library. Multipage TIFF images can also be loaded using GDI+ by setting the Options parameter to -PageNumber (for example -3 for page 3).

SetSignProcessFieldImageFromString

Version history

This function was introduced in Quick PDF Library version 11.15.

Description

Sets the image data to use for a visual signature field in the specified digital signature process.

The **SetSignProcessFieldBounds** function can be used to specify the location and size of the signature field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetSignProcessFieldImageFromString(
  SignProcessID: Integer; const Source: AnsiString;
  Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetSignProcessFieldImageFromString(
  SignProcessID As Long, Source As String,
  Options As Long) As Long
```

DLL

```
int DPLSetSignProcessFieldImageFromString(int InstanceID,
  int SignProcessID, char * Source, int Options);
```

Parameters

SignProcessID	A value returned by the **NewSignProcessFromFile**, **NewSignProcessFromStream** or **NewSignProcessFromString** functions.
Source	A string of binary characters to use for the image data
Options	For multi-page TIFF images this parameter specifies the page number to load. For PNG images: 0 = Load the image as usual 1 = Load the alpha channel as a greyscale image 2 = Load the image and alpha channel (limit alpha to 8-bit) 3 = Load the image (limit image 8-bit/channel) 4 = Load the alpha channel (limit to 8-bit/channel) 5 = Load the image with alpha channel (limit both to 8-bit/channel) 6 = Load the image and alpha channel 7 = Load the image and ICC color profile For other image types this parameter should be set to 0. Setting Options to -1 forces TIFF, EMF and WMF images to be loaded using the GDI+ graphics library. Multipage TIFF images can also be loaded using GDI+ by setting the Options parameter to -PageNumber (for example -3 for page 3).

SetSignProcessFieldPage

Version history

This function was introduced in Quick PDF Library version 9.15.

Description

Specifies the page number where the new signature field will be placed. By default the signature field will be attached to the first page in the document.

If the field name specified by **SetSignProcessField** already exists then a call to this function will be ignored and the field will remain on the page it is currently attached to.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetSignProcessFieldPage(SignProcessID,
  SignaturePage: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetSignProcessFieldPage(
  SignProcessID As Long, SignaturePage As Long) As Long
```

DLL

```
int DPLSetSignProcessFieldPage(int InstanceID, int SignProcessID,
  int SignaturePage);
```

Parameters

SignProcessID	A value returned by the **NewSignProcessFromFile**, **NewSignProcessFromStream** or **NewSignProcessFromString** functions.
SignaturePage	The number of the page that the signature should appear on.

Return values

0	The SignProcessID parameter is invalid
1	Success

SetSignProcessInfo

Version history

This function was introduced in Quick PDF Library version 9.13.

Description

Sets the signing infomation for a digital signature process.

This information includes the reason for signing, the location and contact info. The supplied details will be displayed by the PDF viewer when the signature has been validated.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetSignProcessInfo(SignProcessID: Integer;
  Reason, Location, ContactInfo: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetSignProcessInfo(
  SignProcessID As Long, Reason As String, Location As String,
  ContactInfo As String) As Long
```

DLL

```
int DPLSetSignProcessInfo(int InstanceID, int SignProcessID,
  wchar_t * Reason, wchar_t * Location, wchar_t * ContactInfo);
```

Parameters

SignProcessID	A value returned by the **NewSignProcessFromFile**, **NewSignProcessFromStream** or **NewSignProcessFromString** functions.
Reason	The reason for signing
Location	The location that the signing was done
ContactInfo	The contact information of the signer

SetSignProcessKeyset

Version history

This function was introduced in Quick PDF Library version 9.16.

Description

Sets the MS Crypto API keyset value.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetSignProcessKeyset(SignProcessID,
  KeysetID: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetSignProcessKeyset(
  SignProcessID As Long, KeysetID As Long) As Long
```

DLL

```
int DPLSetSignProcessKeyset(int InstanceID, int SignProcessID,
  int KeysetID);
```

Parameters

SignProcessID	A value returned by the **NewSignProcessFromFile**, **NewSignProcessFromStream** or **NewSignProcessFromString** functions.
KeysetID	1 = CRYPT_USER_KEYSET (Default) 2 = CRYPT_MACHINE_KEYSET

Return values

0	Invalid SignProcessID parameter or KeysetID out of range
1	Signature process keyset was set successfully

SetSignProcessPFXFromFile

Version history

This function was introduced in Quick PDF Library version 9.13.

Description

Sets a file to use as the digital identity for a digital signature process.

The file should be in PKCS #12 format, also known as a PFX file, and contain a private key as well as an X.509 certificate. PFX files are usually protected with a password.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetSignProcessPFXFromFile(
  SignProcessID: Integer; PFXFileName, PFXPassword: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetSignProcessPFXFromFile(
  SignProcessID As Long, PFXFileName As String,
  PFXPassword As String) As Long
```

DLL

```
int DPLSetSignProcessPFXFromFile(int InstanceID, int SignProcessID,
  wchar_t * PFXFileName, wchar_t * PFXPassword);
```

Parameters

SignProcessID	A value returned by the **NewSignProcessFromFile**, **NewSignProcessFromStream** or **NewSignProcessFromString** functions.
PFXFileName	The path and name of the PFX signature file (PKCS #12 format).
PFXPassword	The password to open the PFX file.

SetSignProcessPFXFromStream

Version history

This function was introduced in Quick PDF Library version 11.15.

Description

Sets the data to use as the digital identity for a digital signature process.

The data should be in PKCS #12 format, also known as a PFX file, and contain a private key as well as an X.509 certificate. PFX files are usually protected with a password.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetSignProcessPFXFromStream(
  SignProcessID: Integer; InputStream: TStream;
  PFXPassword: WideString): Integer;
```

Parameters

SignProcessID	A value returned by the **NewSignProcessFromFile**, **NewSignProcessFromStream** or **NewSignProcessFromString** functions.
InputStream	A stream containing the PFX data (PKCS #12 format).
PFXPassword	The password to open the PFX file.

SetSignProcessPFXFromString

Version history

This function was introduced in Quick PDF Library version 11.15.

Description

Sets the data to use as the digital identity for a digital signature process.

The data should be in PKCS #12 format, also known as a PFX file, and contain a private key as well as an X.509 certificate. PFX files are usually protected with a password.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetSignProcessPFXFromString(
  SignProcessID: Integer; const Source: AnsiString;
  PFXPassword: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetSignProcessPFXFromString(
  SignProcessID As Long, Source As String,
  PFXPassword As String) As Long
```

DLL

```
int DPLSetSignProcessPFXFromString(int InstanceID, int SignProcessID,
  char * Source, wchar_t * PFXPassword);
```

Parameters

SignProcessID	A value returned by the **NewSignProcessFromFile**, **NewSignProcessFromStream** or **NewSignProcessFromString** functions.
Source	A string of binary characters to use as the PFX data (PKCS #12 format).
PFXPassword	The password to open the PFX file.

SetSignProcessPassthrough

Version history

This function was introduced in Quick PDF Library version 9.15.

Description

Sets the signature process to passthrough mode.

In this mode, the PDF is prepared using a placeholder for the signature data. The user can then replace this placeholder with the signature data of their choice using the **GetSignProcessByteRange** function to determine the byte range that the signature hashing should be calculated over.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetSignProcessPassthrough(SignProcessID,
  SignatureLength: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetSignProcessPassthrough(
  SignProcessID As Long, SignatureLength As Long) As Long
```

DLL

```
int DPLSetSignProcessPassthrough(int InstanceID, int SignProcessID,
  int SignatureLength);
```

Parameters

SignProcessID	A value returned by the **NewSignProcessFromFile**, **NewSignProcessFromStream** or **NewSignProcessFromString** functions.
SignatureLength	The length in bytes of the raw binary signature data. This value will be doubled when allocating the space in the PDF as the signature data should be written using hex encoding (two characters per raw byte).

Return values

0	The signature could not be set into passthrough mode.
1	Success

SetSignProcessSubFilter

Version history

This function was introduced in Quick PDF Library version 9.14.

Description

Sets the SubFilter entry for a digital signature process, specifying the encoding of the signature value.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetSignProcessSubFilter(SignProcessID,
  SubFilter: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetSignProcessSubFilter(
  SignProcessID As Long, SubFilter As Long) As Long
```

DLL

```
int DPLSetSignProcessSubFilter(int InstanceID, int SignProcessID,
  int SubFilter);
```

Parameters

SignProcessID	A value returned by the **NewSignProcessFromFile**, **NewSignProcessFromStream** or **NewSignProcessFromString** functions.
SubFilter	1 = adbe.pkcs7.sha1 2 = adbe.pkcs7.detached

Return values

0	The SignProcessID parameter was invalid or the SubFilter parameter was out of range
1	Success

SetTabOrderMode

Version history

This function was introduced in Quick PDF Library version 9.16.

Description

This function sets the default tabbing order mode for the currently selected page for all of the annotations including the formfields when tabbing in a PDF viewer.

If you use **SetFormFieldTabOrder** to define a custom tabbing order then you should set the tabbing order to 'S'tructure mode.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTabOrderMode(Mode: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTabOrderMode(
  Mode As String) As Long
```

DLL

```
int DPLSetTabOrderMode(int InstanceID, wchar_t * Mode);
```

Parameters

Mode	The mode string
	'S' - Structure mode - use the order of the Annots and/or Formfields as they are defined 'R' - Row Mode - Left to right, top to bottom order 'C' - Column Mode - Top to botton, left to right order

Return values

0	The tabbing mode was not set correctly
1	Success

SetTableBorderColor

Version history

This function was introduced in Quick PDF Library version 7.14.

Description

Sets the color of the specified table border using the RGB color space.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTableBorderColor(TableID,
  BorderIndex: Integer; Red, Green, Blue: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTableBorderColor(
  TableID As Long, BorderIndex As Long, Red As Double,
  Green As Double, Blue As Double) As Long
```

DLL

```
int DPLSetTableBorderColor(int InstanceID, int TableID, int BorderIndex,
  double Red, double Green, double Blue);
```

Parameters

TableID	A TableID returned by the **CreateTable** function
BorderIndex	0 = All borders 1 = Left 2 = Top 3 = Right 4 = Bottom
Red	The red component of the color, a value from 0 to 1
Green	The green component of the color, a value from 0 to 1
Blue	The blue component of the color, a value from 0 to 1

SetTableBorderColorCMYK

Color, Page layout

Version history

This function was introduced in Quick PDF Library version 7.14.

Description

Sets the color of the specified table border using the CMYK color space.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTableBorderColorCMYK(TableID,
  BorderIndex: Integer; C, M, Y, K: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTableBorderColorCMYK(
  TableID As Long, BorderIndex As Long, C As Double,
  M As Double, Y As Double, K As Double) As Long
```

DLL

```
int DPLSetTableBorderColorCMYK(int InstanceID, int TableID,
  int BorderIndex, double C, double M, double Y, double K);
```

Parameters

TableID	A TableID returned by the **CreateTable** function
BorderIndex	0 = All borders 1 = Left 2 = Top 3 = Right 4 = Bottom
C	The cyan component of the color, a value from 0 to 1
M	The magenta component of the color, a value from 0 to 1
Y	The yellow component of the color, a value from 0 to 1
K	The black component of the color, a value from 0 to 1

SetTableBorderWidth

Version history

This function was introduced in Quick PDF Library version 7.14.

Description

Sets the width of the specified table border.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTableBorderWidth(TableID,
  BorderIndex: Integer; NewWidth: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTableBorderWidth(
  TableID As Long, BorderIndex As Long,
  NewWidth As Double) As Long
```

DLL

```
int DPLSetTableBorderWidth(int InstanceID, int TableID, int BorderIndex,
  double NewWidth);
```

Parameters

TableID	A TableID returned by the **CreateTable** function
BorderIndex	0 = All borders 1 = Left 2 = Top 3 = Right 4 = Bottom
NewWidth	The new width of the specified table border

SetTableCellAlignment

Version history

This function was introduced in Quick PDF Library version 7.14.

Description

Sets the vertical and horizontal alignment of one or more cells.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTableCellAlignment(TableID, FirstRow,
  FirstColumn, LastRow, LastColumn, NewCellAlignment: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTableCellAlignment(
  TableID As Long, FirstRow As Long, FirstColumn As Long,
  LastRow As Long, LastColumn As Long,
  NewCellAlignment As Long) As Long
```

DLL

```
int DPLSetTableCellAlignment(int InstanceID, int TableID, int FirstRow,
  int FirstColumn, int LastRow, int LastColumn,
  int NewCellAlignment);
```

Parameters

TableID	A TableID returned by the **CreateTable** function
FirstRow	The the number of the first row to set. Top row is row number 1.
FirstColumn	The the number of the first column to set. Left most column is column number 1.
LastRow	The number of the final row to set
LastColumn	The number of the final column to set
NewCellAlignment	0 = top left 1 = top center 2 = top right 3 = middle left 4 = middle center 5 = middle right 6 = bottom left 7 = bottom center 8 = bottom right

SetTableCellBackgroundColor

Version history

This function was introduced in Quick PDF Library version 7.14.

Description

Sets the background color of one or more cells using the RGB color space.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTableCellBackgroundColor(TableID,
  FirstRow, FirstColumn, LastRow, LastColumn: Integer; Red, Green,
  Blue: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTableCellBackgroundColor(
  TableID As Long, FirstRow As Long, FirstColumn As Long,
  LastRow As Long, LastColumn As Long, Red As Double,
  Green As Double, Blue As Double) As Long
```

DLL

```
int DPLSetTableCellBackgroundColor(int InstanceID, int TableID,
  int FirstRow, int FirstColumn, int LastRow, int LastColumn,
  double Red, double Green, double Blue);
```

Parameters

TableID	A TableID returned by the **CreateTable** function
FirstRow	The the number of the first row to set. Top row is row number 1.
FirstColumn	The the number of the first column to set. Left most column is column number 1.
LastRow	The number of the final row to set
LastColumn	The number of the final column to set
Red	The red component of the color, a value from 0 to 1
Green	The green component of the color, a value from 0 to 1
Blue	The blue component of the color, a value from 0 to 1

SetTableCellBackgroundColorCMYK

Color, Page layout

Version history

This function was introduced in Quick PDF Library version 7.14.

Description

Sets the background color of one or more cells using the CMYK color space.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTableCellBackgroundColorCMYK(TableID,
  FirstRow, FirstColumn, LastRow, LastColumn: Integer; C, M, Y,
  K: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTableCellBackgroundColorCMYK(
  TableID As Long, FirstRow As Long, FirstColumn As Long,
  LastRow As Long, LastColumn As Long, C As Double, M As Double,
  Y As Double, K As Double) As Long
```

DLL

```
int DPLSetTableCellBackgroundColorCMYK(int InstanceID, int TableID,
  int FirstRow, int FirstColumn, int LastRow, int LastColumn,
  double C, double M, double Y, double K);
```

Parameters

TableID	A TableID returned by the **CreateTable** function
FirstRow	The the number of the first row to set. Top row is row number 1.
FirstColumn	The the number of the first column to set. Left most column is column number 1.
LastRow	The number of the final row to set
LastColumn	The number of the final column to set
C	The cyan component of the color, a value from 0 to 1
M	The magenta component of the color, a value from 0 to 1
Y	The yellow component of the color, a value from 0 to 1
K	The black component of the color, a value from 0 to 1

SetTableCellBorderColor

Color, Page layout

Version history

This function was introduced in Quick PDF Library version 7.14.

Description

Sets the color of one or more cell borders using the RGB color space.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTableCellBorderColor(TableID, FirstRow,
  FirstColumn, LastRow, LastColumn, BorderIndex: Integer; Red, Green,
  Blue: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTableCellBorderColor(
  TableID As Long, FirstRow As Long, FirstColumn As Long,
  LastRow As Long, LastColumn As Long, BorderIndex As Long,
  Red As Double, Green As Double, Blue As Double) As Long
```

DLL

```
int DPLSetTableCellBorderColor(int InstanceID, int TableID, int FirstRow,
  int FirstColumn, int LastRow, int LastColumn, int BorderIndex,
  double Red, double Green, double Blue);
```

Parameters

TableID	A TableID returned by the **CreateTable** function
FirstRow	The the number of the first row to set. Top row is row number 1.
FirstColumn	The the number of the first column to set. Left most column is column number 1.
LastRow	The number of the final row to set
LastColumn	The number of the final column to set
BorderIndex	0 = All borders 1 = Left 2 = Top 3 = Right 4 = Bottom
Red	The red component of the color, a value from 0 to 1
Green	The green component of the color, a value from 0 to 1
Blue	The blue component of the color, a value from 0 to 1

SetTableCellBorderColorCMYK

Version history

This function was introduced in Quick PDF Library version 7.14.

Description

Sets the color of one or more cell borders using the CMYK color space.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTableCellBorderColorCMYK(TableID,
  FirstRow, FirstColumn, LastRow, LastColumn, BorderIndex: Integer; C, M,
  Y, K: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTableCellBorderColorCMYK(
  TableID As Long, FirstRow As Long, FirstColumn As Long,
  LastRow As Long, LastColumn As Long, BorderIndex As Long,
  C As Double, M As Double, Y As Double, K As Double) As Long
```

DLL

```
int DPLSetTableCellBorderColorCMYK(int InstanceID, int TableID,
  int FirstRow, int FirstColumn, int LastRow, int LastColumn,
  int BorderIndex, double C, double M, double Y, double K);
```

Parameters

TableID	A TableID returned by the **CreateTable** function
FirstRow	The the number of the first row to set. Top row is row number 1.
FirstColumn	The the number of the first column to set. Left most column is column number 1.
LastRow	The number of the final row to set
LastColumn	The number of the final column to set
BorderIndex	0 = All borders 1 = Left 2 = Top 3 = Right 4 = Bottom
C	The cyan component of the color, a value from 0 to 1
M	The magenta component of the color, a value from 0 to 1
Y	The yellow component of the color, a value from 0 to 1
K	The black component of the color, a value from 0 to 1

SetTableCellBorderWidth

Version history

This function was introduced in Quick PDF Library version 7.14.

Description

Sets the width of one or more cell borders.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTableCellBorderWidth(TableID, FirstRow,
  FirstColumn, LastRow, LastColumn, BorderIndex: Integer;
  NewWidth: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTableCellBorderWidth(
  TableID As Long, FirstRow As Long, FirstColumn As Long,
  LastRow As Long, LastColumn As Long, BorderIndex As Long,
  NewWidth As Double) As Long
```

DLL

```
int DPLSetTableCellBorderWidth(int InstanceID, int TableID, int FirstRow,
  int FirstColumn, int LastRow, int LastColumn, int BorderIndex,
  double NewWidth);
```

Parameters

TableID	A TableID returned by the **CreateTable** function
FirstRow	The the number of the first row to set. Top row is row number 1.
FirstColumn	The the number of the first column to set. Left most column is column number 1.
LastRow	The number of the final row to set
LastColumn	The number of the final column to set
BorderIndex	0 = All borders 1 = Left 2 = Top 3 = Right 4 = Bottom
NewWidth	The new width of the specified border

SetTableCellContent

Version history

This function was introduced in Quick PDF Library version 7.14.

Description

Sets the content of the specified cell. The content will be drawn with the equivalent of the **DrawHTMLText** function, prefixed with the necessary paragraph alignment, font size and font color tags.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTableCellContent(TableID, RowNumber,
  ColumnNumber: Integer; HTMLText: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTableCellContent(
  TableID As Long, RowNumber As Long, ColumnNumber As Long,
  HTMLText As String) As Long
```

DLL

```
int DPLSetTableCellContent(int InstanceID, int TableID, int RowNumber,
  int ColumnNumber, wchar_t * HTMLText);
```

Parameters

TableID	A TableID returned by the **CreateTable** function
RowNumber	The the row number of the cell. Top row is row number 1.
ColumnNumber	The the column number of the cell. Left most column is column number 1.
HTMLText	The HTML text to place into the specified cell

SetTableCellPadding

Version history

This function was introduced in Quick PDF Library version 7.14.

Description

Sets the padding of one or more cells. The padding is the distance from the cell boundary to the text contents. The padding is set on the side of the specified border.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTableCellPadding(TableID, FirstRow,
  FirstColumn, LastRow, LastColumn, BorderIndex: Integer;
  NewPadding: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTableCellPadding(
  TableID As Long, FirstRow As Long, FirstColumn As Long,
  LastRow As Long, LastColumn As Long, BorderIndex As Long,
  NewPadding As Double) As Long
```

DLL

```
int DPLSetTableCellPadding(int InstanceID, int TableID, int FirstRow,
  int FirstColumn, int LastRow, int LastColumn, int BorderIndex,
  double NewPadding);
```

Parameters

TableID	A TableID returned by the **CreateTable** function
FirstRow	The the number of the first row to set. Top row is row number 1.
FirstColumn	The the number of the first column to set. Left most column is column number 1.
LastRow	The number of the final row to set
LastColumn	The number of the final column to set
BorderIndex	0 = All borders 1 = Left 2 = Top 3 = Right 4 = Bottom
NewPadding	The new padding on the side of the specified border

SetTableCellTextColor

Version history

This function was introduced in Quick PDF Library version 7.14.

Description

Sets the default text color of one or more cells using the RGB color space.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTableCellTextColor(TableID, FirstRow,
  FirstColumn, LastRow, LastColumn: Integer; Red, Green,
  Blue: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTableCellTextColor(
  TableID As Long, FirstRow As Long, FirstColumn As Long,
  LastRow As Long, LastColumn As Long, Red As Double,
  Green As Double, Blue As Double) As Long
```

DLL

```
int DPLSetTableCellTextColor(int InstanceID, int TableID, int FirstRow,
  int FirstColumn, int LastRow, int LastColumn, double Red,
  double Green, double Blue);
```

Parameters

TableID	A TableID returned by the **CreateTable** function
FirstRow	The the number of the first row to set. Top row is row number 1.
FirstColumn	The the number of the first column to set. Left most column is column number 1.
LastRow	The number of the final row to set
LastColumn	The number of the final column to set
Red	The red component of the color, a value from 0 to 1
Green	The green component of the color, a value from 0 to 1
Blue	The blue component of the color, a value from 0 to 1

SetTableCellTextColorCMYK

Version history

This function was introduced in Quick PDF Library version 7.14.

Description

Sets the default text color of one or more cells using the CMYK color space.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTableCellTextColorCMYK(TableID,
  FirstRow, FirstColumn, LastRow, LastColumn: Integer; C, M, Y,
  K: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTableCellTextColorCMYK(
  TableID As Long, FirstRow As Long, FirstColumn As Long,
  LastRow As Long, LastColumn As Long, C As Double, M As Double,
  Y As Double, K As Double) As Long
```

DLL

```
int DPLSetTableCellTextColorCMYK(int InstanceID, int TableID,
  int FirstRow, int FirstColumn, int LastRow, int LastColumn,
  double C, double M, double Y, double K);
```

Parameters

TableID	A TableID returned by the **CreateTable** function
FirstRow	The the number of the first row to set. Top row is row number 1.
FirstColumn	The the number of the first column to set. Left most column is column number 1.
LastRow	The number of the final row to set
LastColumn	The number of the final column to set
C	The cyan component of the color, a value from 0 to 1
M	The magenta component of the color, a value from 0 to 1
Y	The yellow component of the color, a value from 0 to 1
K	The black component of the color, a value from 0 to 1

SetTableCellTextSize

Version history

This function was introduced in Quick PDF Library version 7.14.

Description

Sets the default text size of one or more cells.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTableCellTextSize(TableID, FirstRow,
  FirstColumn, LastRow, LastColumn: Integer; NewTextSize: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTableCellTextSize(
  TableID As Long, FirstRow As Long, FirstColumn As Long,
  LastRow As Long, LastColumn As Long,
  NewTextSize As Double) As Long
```

DLL

```
int DPLSetTableCellTextSize(int InstanceID, int TableID, int FirstRow,
  int FirstColumn, int LastRow, int LastColumn,
  double NewTextSize);
```

Parameters

TableID	A TableID returned by the **CreateTable** function
FirstRow	The the number of the first row to set. Top row is row number 1.
FirstColumn	The the number of the first column to set. Left most column is column number 1.
LastRow	The number of the final row to set
LastColumn	The number of the final column to set
NewTextSize	The new text size for the specified cell range

SetTableColumnWidth

Version history

This function was introduced in Quick PDF Library version 7.14.

Description

Sets the width of one or more table columns.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTableColumnWidth(TableID, FirstColumn,
  LastColumn: Integer; NewWidth: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTableColumnWidth(
  TableID As Long, FirstColumn As Long, LastColumn As Long,
  NewWidth As Double) As Long
```

DLL

```
int DPLSetTableColumnWidth(int InstanceID, int TableID, int FirstColumn,
  int LastColumn, double NewWidth);
```

Parameters

TableID	A TableID returned by the **CreateTable** function
FirstColumn	The the number of the first column to set. Left most column is column number 1.
LastColumn	The number of the final column to set
NewWidth	The new width of the specified columns

SetTableRowHeight

Version history

This function was introduced in Quick PDF Library version 7.14.

Description

Sets the height of one or more table rows. If the row height is set to zero (default) the row will autosize to the maximum height of the contents of all the cells in the row.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTableRowHeight(TableID, FirstRow,
  LastRow: Integer; NewHeight: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTableRowHeight(
  TableID As Long, FirstRow As Long, LastRow As Long,
  NewHeight As Double) As Long
```

DLL

```
int DPLSetTableRowHeight(int InstanceID, int TableID, int FirstRow,
  int LastRow, double NewHeight);
```

Parameters

TableID	A TableID returned by the **CreateTable** function
FirstRow	The the number of the first row to set. Top row is row number 1.
LastRow	The number of the final row to set
NewHeight	0 = auto size Non-zero = the new maximum height of the row

SetTableThinBorders

Version history

This function was introduced in Quick PDF Library version 8.14.

Description

Sets a table to use thin border lines instead of bevelled edges. These lines appear as a single pixel width for all zoom levels.

The lines are drawn using the color specified by the Red, Green and Blue parameters.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTableThinBorders(TableID,
  ThinBorders: Integer; Red, Green, Blue: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTableThinBorders(
  TableID As Long, ThinBorders As Long, Red As Double,
  Green As Double, Blue As Double) As Long
```

DLL

```
int DPLSetTableThinBorders(int InstanceID, int TableID, int ThinBorders,
  double Red, double Green, double Blue);
```

Parameters

TableID	A TableID returned by the **CreateTable** function
ThinBorders	0 = Use bevelled edges (the default) 1 = Use thin lines
Red	The red component of the color, a value from 0 to 1
Green	The green component of the color, a value from 0 to 1
Blue	The blue component of the color, a value from 0 to 1

Return values

0	The table line style could not be set
1	The table line style was set successfully

SetTableThinBordersCMYK

Version history

This function was introduced in Quick PDF Library version 8.14.

Description

Sets a table to use thin border lines instead of bevelled edges. These lines appear as a single pixel width for all zoom levels.

The lines are drawn using the color specified by the C, M, Y and K parameters.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTableThinBordersCMYK(TableID,
  ThinBorders: Integer; C, M, Y, K: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTableThinBordersCMYK(
  TableID As Long, ThinBorders As Long, C As Double,
  M As Double, Y As Double, K As Double) As Long
```

DLL

```
int DPLSetTableThinBordersCMYK(int InstanceID, int TableID,
  int ThinBorders, double C, double M, double Y, double K);
```

Parameters

TableID	A TableID returned by the **CreateTable** function
ThinBorders	0 = Use bevelled edges (the default) 1 = Use thin lines
C	The cyan component of the color, a value from 0 to 1
M	The magenta component of the color, a value from 0 to 1
Y	The yellow component of the color, a value from 0 to 1
K	The black component of the color, a value from 0 to 1

Return values

0	The table line style could not be set
1	The table line style was set successfully

SetTempFile

Description

Specifies a temporary file which can be used during operations such as encryption. This allows large documents to be processed without running out of memory.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTempFile(FileName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTempFile(
  FileName As String) As Long
```

DLL

```
int DPLSetTempFile(int InstanceID, wchar_t * FileName);
```

Parameters

FileName	The full path and file to use as a temporary file. This path must have write access by the running process. For example, "c:\temp\pdftemp.dat".

Return values

0	The path specified was not valid. A temporary file could not be created.
1	The temporary file could be created successfully

SetTempPath

Description

Sets the folder to use for storage of temporary files which are generated by functions such as **MergeFileList**.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTempPath(NewPath: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTempPath(
  NewPath As String) As Long
```

DLL

```
int DPLSetTempPath(int InstanceID, wchar_t * NewPath);
```

Parameters

NewPath	The new folder to use. This folder must exist already, it will not be created.

Return values

0	The specified folder does not exists or does not have read/write access
1	The temporary path was set successfully

SetTextAlign

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Set the alignment of subsequent text drawn with the **DrawText**, **DrawWrappedText** or **DrawMultiLineText** functions.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTextAlign(TextAlign: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTextAlign(
  TextAlign As Long) As Long
```

DLL

```
int DPLSetTextAlign(int InstanceID, int TextAlign);
```

Parameters

TextAlign	The alignment of the text: 0 = Left aligned (default) 1 = Center aligned 2 = Right aligned 3 = Justified 4 = Force justified 5 = Last line justified Anything else = Left aligned
	"Justified" mode will not justify a line if it's the last line in a paragraph or if the line ends with a hard-break. "Force justified" will justify every line even if it's the last line or if it ends with a hard-break. "Last line justified" will not justify the last line of text, this is useful when different blocks of text are drawn one after the other.

SetTextCharSpacing

Description

Sets the amount of space to add between characters for subsequently drawn text.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTextCharSpacing(
  CharSpacing: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTextCharSpacing(
  CharSpacing As Double) As Long
```

DLL

```
int DPLSetTextCharSpacing(int InstanceID, double CharSpacing);
```

Parameters

CharSpacing	The amount of extra space to add between characters

SetTextColor

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Sets the color for any subsequently drawn text. The values of the color parameters range from 0 to 1, with 0 indicating 0% and 1 indicating 100% of the color.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTextColor(Red, Green,
  Blue: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTextColor(Red As Double,
  Green As Double, Blue As Double) As Long
```

DLL

```
int DPLSetTextColor(int InstanceID, double Red, double Green, double Blue);
```

Parameters

Red	The red component of the color
Green	The green component of the color
Blue	The blue component of the color

SetTextColorCMYK

Description

Sets the color for any subsequently drawn text. Similar to the **SetTextColor** function, but the color components are specified in the CMYK color space (Cyan, Magenta, Yellow, Black). The values of the color parameters range from 0 to 1, with 0 indicating 0% and 1 indicating 100% of the color.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTextColorCMYK(C, M, Y,
  K: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTextColorCMYK(C As Double,
  M As Double, Y As Double, K As Double) As Long
```

DLL

```
int DPLSetTextColorCMYK(int InstanceID, double C, double M, double Y,
  double K);
```

Parameters

C	The cyan component of the color
M	The magenta component of the color
Y	The yellow component of the color
K	The black component of the color

SetTextColorSep

Description

Sets the color for any subsequently drawn text. Similar to the **SetTextColor** function, but a tint of a separation color added with the **AddSeparationColor** function is used.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTextColorSep(ColorName: WideString;
  Tint: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTextColorSep(
  ColorName As String, Tint As Double) As Long
```

DLL

```
int DPLSetTextColorSep(int InstanceID, wchar_t * ColorName, double Tint);
```

Parameters

ColorName	The name of the separation color that was used with the **AddSeparationColor** function
Tint	The amount of color to use. 0 indicates no color (white), 1 indicates maximum color.

Return values

0	The separation color name could not be found
1	The text color was set successfully

SetTextExtractionArea

Text, Extraction

Version history

This function was introduced in Quick PDF Library version 8.12.

Description

Sets the area for certain modes of text extraction. Any text that appears outside this area will be excluded from the results. This function has no effect on text extraction using modes 0 to 2.

From 8.13, this function sets the text extraction area for the selected document only. It also only affects the results of the **GetPageText** function.

To adjust the text extraction for the **ExtractFilePageText** and **DAExtractPageText** functions, use the new **DASetTextExtractionArea** function.

The coordinate values passed into this function are specified using the units set with the **SetMeasurementUnits** function and the origin set with the **SetOrigin** function.

The area limitation can be removed by calling this function with a value of zero for both the Width and Height parameters.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTextExtractionArea(Left, Top, Width,
  Height: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTextExtractionArea(
  Left As Double, Top As Double, Width As Double,
  Height As Double) As Long
```

DLL

```
int DPLSetTextExtractionArea(int InstanceID, double Left, double Top,
  double Width, double Height);
```

Parameters

Left	The horizontal coordinate of the left edge of the area
Top	The vertical coordinate of the top edge of the area
Width	The width of the area
Height	The height of the area

Return values

1	The text extraction area was set successfully
2	The text extraction area was cleared

SetTextExtractionOptions

Text, Extraction

Version history

This function was introduced in Quick PDF Library version 8.11.

Description

Sets various options that affect the text extraction functionality.

From 8.13, this function sets the text extraction options for the selected document only. It also only affects the results of the **GetPageText** function.

To adjust the text extraction for the **ExtractFilePageText** and **DAExtractPageText** functions, use the new **DASetTextExtractionOptions** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTextExtractionOptions(OptionID,
  NewValue: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTextExtractionOptions(
  OptionID As Long, NewValue As Long) As Long
```

DLL

```
int DPLSetTextExtractionOptions(int InstanceID, int OptionID,
  int NewValue);
```

Parameters

OptionID	1 = Ignore Font changes to allow grouping different blocks together 2 = Ignore Color changes to allow grouping different blocks together 3 = Ignore Text Block changes to allow grouping different blocks together 4 = Output CMYK color values 5 = Sort text blocks based on top left position 6 = Descenders from font metrics 7 = Ignore overlaps 8 = Ignore duplicates 9 = Split on double space 10 = Trim characters outside area 11 = Alternative block matching 12 = Ignore rotated text blocks 13 = Trim leading and trailing whitespace from text blocks 14 = Output non ASCII characters below Space character (0x32) 15 = Remove certain character strings such as underscore lines (see below)
NewValue	For OptionID = 1, 2, 3 and 6: 0 = Use, 1 = Ignore For OptionID = 4: 0 = Show as RGB (default), 1 = Show as CMYK For OptionID = 5: 0 = Do not sort blocks (default), 1 = Sort blocks For OptionID = 7, 8 and 12: 0 = Do not ignore, 1 = Ignore OptionID = 9: 0 = Do not split on double space (default) 1 = Split on double space OptionID = 10: 0 = Do not trim characters outside area (default) 1 = Trim characters outside area OptionID = 11: 0 = Regular block matching 1 = Alternative block matching OptionID = 13: 0 = Do not trim leading or trailing whitespace 1 = Trim leading and trailing whitespace OptionID = 14 0 = Remove non ASCII chracters below space character from output (default) 1 = Output raw unfiltered ASCII characters OptionID = 15 0 = Output text lines made with Underscore characters (default) 1 = Remove text lines made with Underscore characters

Return values

0	The OptionID or NewValue parameter was not valid
1	The text extraction option was set successfully

SetTextExtractionScaling

Version history

This function was introduced in Quick PDF Library version 8.16.

Description

Sets the scaling to use for the **GetPageText** function in Mode 7. This controls the number of rows and columns in the monospaced text output.

The setting is applied to the selected document only.

To adjust the text extraction for the **ExtractFilePageText** and **DAExtractPageText** functions, use the **DASetTextExtractionScaling** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTextExtractionScaling(Options: Integer;
  Horizontal, Vertical: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTextExtractionScaling(
  Options As Long, Horizontal As Double,
  Vertical As Double) As Long
```

DLL

```
int DPLSetTextExtractionScaling(int InstanceID, int Options,
  double Horizontal, double Vertical);
```

Parameters

Options	Should always be set to 0. This indicates a scaling factor will be set for the Horizontal and Vertical parameters, with a default value of 5 for horizontal and 8 for vertical. Smaller values stretch the text out into more rows/columns.
Horizontal	The scaling to use for the horizontal axis in units defined by the Options parameter.
Vertical	The scaling to use for the vertical axis in units defined by the Options parameter.

Return values

0	The Options parameter was not valid or a value less than 1 was used for the Horizontal or Vertical parameters.
1	Text extraction scaling was set successfully.

SetTextExtractionWordGap

Text, Extraction

Version history

This function was introduced in Quick PDF Library version 7.21.

Description

Sets the word gap ratio for the text extraction functionality.

From 8.13, this function sets the text extraction options for the selected document only. It affects the results of any of the text extraction function that use options 3,4,5,6,7 or 8.

To adjust the text extraction for the **ExtractFilePageText** and **DAExtractPageText** functions, use the new **DASetTextExtractionWordGap** function.

The word gap ratio is the maximum distance between two text blocks specified as the ratio of the horizontal distance between the blocks to the height of the text.

The default initial value is 0.7 and smaller values will allow closer distances between words.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTextExtractionWordGap(
  NewWordGap: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTextExtractionWordGap(
  NewWordGap As Double) As Long
```

DLL

```
int DPLSetTextExtractionWordGap(int InstanceID, double NewWordGap);
```

Parameters

NewWordGap	The new WordGap ratio

Return values

1	The word gap ratio was set successfully.

SetTextHighlight

Description

Sets the text highlighting mode for subsequently drawn text.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTextHighlight(
  Highlight: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTextHighlight(
  Highlight As Long) As Long
```

DLL

```
int DPLSetTextHighlight(int InstanceID, int Highlight);
```

Parameters

Highlight	The text highlighting mode to use: 0 = None 1 = Square 2 = Rounded

SetTextHighlightColor

Text, Color

Description

Sets the color used to highlight text.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTextHighlightColor(Red, Green,
  Blue: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTextHighlightColor(
  Red As Double, Green As Double, Blue As Double) As Long
```

DLL

```
int DPLSetTextHighlightColor(int InstanceID, double Red, double Green,
  double Blue);
```

Parameters

Red	A value between 0 and 1 indicating the amount of red to add to the highlight color. 0 indicates no red, 1 indicates maximum red.
Green	A value between 0 and 1 indicating the amount of green to add to the highlight color. 0 indicates no green, 1 indicates maximum green.
Blue	A value between 0 and 1 indicating the amount of blue to add to the highlight color. 0 indicates no blue, 1 indicates maximum blue.

SetTextHighlightColorCMYK

Description

Sets the color used to highlight text, but allows the color to be specified in the CMYK color space.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTextHighlightColorCMYK(C, M, Y,
  K: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTextHighlightColorCMYK(
  C As Double, M As Double, Y As Double, K As Double) As Long
```

DLL

```
int DPLSetTextHighlightColorCMYK(int InstanceID, double C, double M,
  double Y, double K);
```

Parameters

C	A value between 0 and 1 indicating the amount of cyan to add to the highlight color. 0 indicates no cyan, 1 indicates maximum cyan.
M	A value between 0 and 1 indicating the amount of magenta to add to the highlight color. 0 indicates no magenta, 1 indicates maximum magenta.
Y	A value between 0 and 1 indicating the amount of yellow to add to the highlight color. 0 indicates no yellow, 1 indicates maximum yellow.
K	A value between 0 and 1 indicating the amount of black to add to the highlight color. 0 indicates no black, 1 indicates maximum black.

SetTextHighlightColorSep

Description

Sets the color used to highlight text. Similar to the **SetTextHighlightColor** function, but a tint of a separation color added with the **AddSeparationColor** function is used.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTextHighlightColorSep(
  ColorName: WideString; Tint: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTextHighlightColorSep(
  ColorName As String, Tint As Double) As Long
```

DLL

```
int DPLSetTextHighlightColorSep(int InstanceID, wchar_t * ColorName,
  double Tint);
```

Parameters

ColorName	The name of the separation color that was used with the **AddSeparationColor** function
Tint	The amount of color to use. 0 indicates no color (white), 1 indicates maximum color.

Return values

0	The separation color name could not be found
1	The text highlight color was set successfully

SetTextMode

Description

Specifies the mode to draw subsequent text in. Modes 4 to 7 are used to add text to the clipping path. If one of these modes is selected and text is drawn onto the page, then subsequent items drawn onto the page will be clipped to the outline of the text.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTextMode(TextMode: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTextMode(
  TextMode As Long) As Long
```

DLL

```
int DPLSetTextMode(int InstanceID, int TextMode);
```

Parameters

TextMode	The text mode: 0 = Filled text (default) 1 = Outline text 2 = Fill then stroke text 3 = Invisible text 4 = Fill text and add to clipping path 5 = Stroke text and add to clipping path 6 = Fill then stroke text and add to clipping path 7 = Add text to clipping path Anything else = Filled text (default)

SetTextRise

Description

Allows text to be positioned above or below the baseline. This is useful for superscript and subscript text.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTextRise(Rise: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTextRise(
  Rise As Double) As Long
```

DLL

```
int DPLSetTextRise(int InstanceID, double Rise);
```

Parameters

Rise	The amount to raise or lower subsequent text from the baseline. Positive values result in text that is higher than normal (superscript), negative values result in text that is lower than normal (subscript).

SetTextScaling

Description

Sets the amount to scale text in the direction the text is written. This stretches all the characters in the font as well as the spacing between the characters.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTextScaling(
   ScalePercentage: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTextScaling(
   ScalePercentage As Double) As Long
```

DLL

```
int DPLSetTextScaling(int InstanceID, double ScalePercentage);
```

Parameters

ScalePercentage	The percentage to scale the text by. Values less than 100 will result in narrower text. Values greater than 100 will result in wider text.

SetTextShader

Version history

This function was introduced in Quick PDF Library version 7.11.

Description

Sets the text color to the specified shader for subsequently drawn text.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTextShader(
  ShaderName: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTextShader(
  ShaderName As String) As Long
```

DLL

```
int DPLSetTextShader(int InstanceID, wchar_t * ShaderName);
```

Parameters

ShaderName	The shader name that was used when the shader was created.

Return values

0	The shader could not be found
1	The text shader was setup correctly

SetTextSize

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Set the size of the text to use for any subsequently draw text. The text size is always measured in points, even if the measurement units have been changed with **SetMeasurementUnits**.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTextSize(TextSize: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTextSize(
  TextSize As Double) As Long
```

DLL

```
int DPLSetTextSize(int InstanceID, double TextSize);
```

Parameters

TextSize	The text size in points

Return values

0	A font has not been selected
1	The text size was set successfully

SetTextSpacing

Description

Set the amount of space to add between each line for the **DrawWrappedText**, **GetWrappedTextHeight** and **DrawMultiLineText** functions.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTextSpacing(Spacing: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTextSpacing(
  Spacing As Double) As Long
```

DLL

```
int DPLSetTextSpacing(int InstanceID, double Spacing);
```

Parameters

Spacing	The amount of space to add between each line

SetTextUnderline

This function is available in the Lite Edition of Debenu Quick PDF Library, see **Appendix C**.

Description

Sets the underline mode for subsequently drawn text.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTextUnderline(
  Underline: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTextUnderline(
  Underline As Long) As Long
```

DLL

```
int DPLSetTextUnderline(int InstanceID, int Underline);
```

Parameters

Underline	The underline mode to use:
	0 = None
	1 = Single
	2 = Double
	3 = Strikeout
	4 = Over

SetTextUnderlineColor

Description

Sets the color used to draw the lines for subsequently drawn text that has an underline style.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTextUnderlineColor(Red, Green,
  Blue: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTextUnderlineColor(
  Red As Double, Green As Double, Blue As Double) As Long
```

DLL

```
int DPLSetTextUnderlineColor(int InstanceID, double Red, double Green,
  double Blue);
```

Parameters

Red	A value between 0 and 1 indicating the amount of red to add to the underline color. 0 indicates no red, 1 indicates maximum red.
Green	A value between 0 and 1 indicating the amount of green to add to the underline color. 0 indicates no green, 1 indicates maximum green.
Blue	A value between 0 and 1 indicating the amount of blue to add to the underline color. 0 indicates no blue, 1 indicates maximum blue.

SetTextUnderlineColorCMYK

Text, Color

Description

Sets the color used to draw the lines for subsequently drawn text that has an underline style, but allows the color to be set using the CMYK color space.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTextUnderlineColorCMYK(C, M, Y,
  K: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTextUnderlineColorCMYK(
  C As Double, M As Double, Y As Double, K As Double) As Long
```

DLL

```
int DPLSetTextUnderlineColorCMYK(int InstanceID, double C, double M,
  double Y, double K);
```

Parameters

C	A value between 0 and 1 indicating the amount of cyan to add to the underline color. 0 indicates no cyan, 1 indicates maximum cyan.
M	A value between 0 and 1 indicating the amount of magenta to add to the underline color. 0 indicates no magenta, 1 indicates maximum magenta.
Y	A value between 0 and 1 indicating the amount of yellow to add to the underline color. 0 indicates no yellow, 1 indicates maximum yellow.
K	A value between 0 and 1 indicating the amount of black to add to the underline color. 0 indicates no black, 1 indicates maximum black.

SetTextUnderlineColorSep

Text, Color

Description

Sets the color used to draw the lines for subsequently drawn text that has an underline style. Similar to the **SetTextUnderlineColor** function, but a tint of a separation color added with the **AddSeparationColor** function is used.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTextUnderlineColorSep(
  ColorName: WideString; Tint: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTextUnderlineColorSep(
  ColorName As String, Tint As Double) As Long
```

DLL

```
int DPLSetTextUnderlineColorSep(int InstanceID, wchar_t * ColorName,
  double Tint);
```

Parameters

ColorName	The name of the separation color that was used with the **AddSeparationColor** function
Tint	The amount of color to use. 0 indicates no color (white), 1 indicates maximum color.

Return values

0	The separation color name could not be found
1	The text underline color was set successfully

SetTextUnderlineCustomDash

Version history

This function was introduced in Quick PDF Library version 8.14.

Description

Use this function to apply a dashed effect to the underlines added to subsequently drawn text.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTextUnderlineCustomDash(
  DashPattern: WideString; DashPhase: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTextUnderlineCustomDash(
  DashPattern As String, DashPhase As Double) As Long
```

DLL

```
int DPLSetTextUnderlineCustomDash(int InstanceID, wchar_t * DashPattern,
  double DashPhase);
```

Parameters

DashPattern	The dash pattern to use, for example "10 5 0 5".
DashPhase	The dash phase. Usually set to zero.

SetTextUnderlineDash

Description

Use this function to apply a dashed effect to the underlines added to subsequently drawn text.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTextUnderlineDash(DashOn,
  DashOff: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTextUnderlineDash(
  DashOn As Double, DashOff As Double) As Long
```

DLL

```
int DPLSetTextUnderlineDash(int InstanceID, double DashOn, double DashOff);
```

Parameters

DashOn	A factor to use for the solid parts of the dashed line. If a factor of 1 is used then the solid parts of the line will be the same width as the line. A factor of 3 would result in the solid parts of the dashed line being three times longer than the width of the line.
DashOff	A factor to use for the invisible parts of the dashed line. For example, if a factor of 2 is used then the invisible parts of the line will be twice as wide as the width of the line.

SetTextUnderlineDistance

Version history

This function was introduced in Quick PDF Library version 8.14.

Description

Sets the distance of the underlines from the text for subsequently drawn text that has an underline style.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTextUnderlineDistance(
  UnderlineDistance: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTextUnderlineDistance(
  UnderlineDistance As Double) As Long
```

DLL

```
int DPLSetTextUnderlineDistance(int InstanceID, double UnderlineDistance);
```

Parameters

UnderlineDistance	The distance from the text to the underline

SetTextUnderlineWidth

Version history

This function was introduced in Quick PDF Library version 8.14.

Description

Sets the width of the underlines for subsequently drawn text that has an underline style.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTextUnderlineWidth(
  UnderlineWidth: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTextUnderlineWidth(
  UnderlineWidth As Double) As Long
```

DLL

```
int DPLSetTextUnderlineWidth(int InstanceID, double UnderlineWidth);
```

Parameters

UnderlineWidth	The width of the underline to use

SetTextWordSpacing

Description

Sets the amount of space to add between words for subsequently drawn text.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTextWordSpacing(
  WordSpacing: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTextWordSpacing(
  WordSpacing As Double) As Long
```

DLL

```
int DPLSetTextWordSpacing(int InstanceID, double WordSpacing);
```

Parameters

WordSpacing	The amount of extra space to add between words

SetTransparency

Description

Sets the transparency for all subsequently drawn text and graphics.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetTransparency(
  Transparency: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetTransparency(
  Transparency As Long) As Long
```

DLL

```
int DPLSetTransparency(int InstanceID, int Transparency);
```

Parameters

Transparency	The amount of transparency to apply 0 = No transparency 50 = 50% transparency 100 = Invisible

Return values

0	The transparency specified was out of range
1	The transparency was set successfully

SetViewerPreferences

Description

Sets the viewer preferences for the document.

For Option=7 to take effect, the initial page mode should be set to Full Screen using the **SetPageMode** function with the NewPageMode parameter set to 3.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetViewerPreferences(Option,
  NewValue: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetViewerPreferences(
  Option As Long, NewValue As Long) As Long
```

DLL

```
int DPLSetViewerPreferences(int InstanceID, int Option, int NewValue);
```

Parameters

Option	1 = Hide toolbar 2 = Hide menubar 3 = Hide window user interface 4 = Resize window to first page size 5 = Center window 6 = Display document title 7 = Page mode after full screen 8 = Predominant text reading order 9 = Display boundary for viewing 10 = Clipping boundary for viewing 11 = Display voundary for printing 12 = Clipping boundary for printing 13 = Default print dialog: scaling 14 = Default print dialog: duplex 15 = Default print dialog: auto paper tray 16 = Default print dialog: number of copies
NewValue	For Option 1 to 6: 0=No, 1=Yes For Option 7: 0=Normal view, 1=Show the outlines pane, 2=Show the thumbnails pane, 3=Show the layers pane For Option 8: 0=Left to right, 1=Right to left For Option 9 to 12: 0=MediaBox, 1=CropBox, 2=BleedBox, 3=TrimBox, 4=ArtBox For Option 13: 0=None, 1=Application default For Option 14: 0=Simplex, 1=Duplex flip short edge, 2=Duplex flip long edge For Option 15: 0=No, 1=Yes For Option 16: Any positive number

Return values

0	The viewer preferences could not be set
1	The viewer preferences were set successfully

SetXFAFormFieldAccess

Description

Sets the access flags of the specified XFA form field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetXFAFormFieldAccess(
  XFAFieldName: WideString; NewAccess: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetXFAFormFieldAccess(
  XFAFieldName As String, NewAccess As Long) As Long
```

DLL

```
int DPLSetXFAFormFieldAccess(int InstanceID, wchar_t * XFAFieldName,
  int NewAccess);
```

Parameters

XFAFieldName	The name of the XFA field to work with
NewAccess	1 = Non interactive 2 = Open 3 = Protected 4 = Read only

SetXFAFormFieldBorderColor

Form fields, Color

Description

Sets the border color of the specified XFA form field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetXFAFormFieldBorderColor(
  XFAFieldName: WideString; Red, Green, Blue: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetXFAFormFieldBorderColor(
  XFAFieldName As String, Red As Double, Green As Double,
  Blue As Double) As Long
```

DLL

```
int DPLSetXFAFormFieldBorderColor(int InstanceID, wchar_t * XFAFieldName,
  double Red, double Green, double Blue);
```

Parameters

XFAFieldName	The name of the XFA field to work with
Red	The red component of the color, which should be a value between 0 and 1
Green	The green component of the color, which should be a value between 0 and 1
Blue	The blue component of the color, which should be a value between 0 and 1

SetXFAFormFieldBorderPresence

Form fields

Description

Sets the border style of the specified XFA form field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetXFAFormFieldBorderPresence(
  XFAFieldName: WideString; NewPresence: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetXFAFormFieldBorderPresence(
  XFAFieldName As String, NewPresence As Long) As Long
```

DLL

```
int DPLSetXFAFormFieldBorderPresence(int InstanceID,
  wchar_t * XFAFieldName, int NewPresence);
```

Parameters

XFAFieldName	The name of the XFA field to work with
NewPresence	1 = Visible 2 = Invisible 3 = Hidden

SetXFAFormFieldBorderWidth

Form fields

Description

Sets the border width of the specified XFA form field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetXFAFormFieldBorderWidth(
  XFAFieldName: WideString; BorderWidth: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetXFAFormFieldBorderWidth(
  XFAFieldName As String, BorderWidth As Double) As Long
```

DLL

```
int DPLSetXFAFormFieldBorderWidth(int InstanceID, wchar_t * XFAFieldName,
  double BorderWidth);
```

Parameters

XFAFieldName	The name of the XFA field to work with
BorderWidth	The desired width of the border

SetXFAFormFieldValue

Description

Sets the value of the specified XFA form field.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetXFAFormFieldValue(XFAFieldName,
  NewValue: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetXFAFormFieldValue(
  XFAFieldName As String, NewValue As String) As Long
```

DLL

```
int DPLSetXFAFormFieldValue(int InstanceID, wchar_t * XFAFieldName,
  wchar_t * NewValue);
```

Parameters

XFAFieldName	The name of the XFA field to work with
NewValue	The new value for the XFA field

SetXFAFromString

Version history

This function was introduced in Quick PDF Library version 8.16.

Description

Sets the document's XFA form data to the specified XML string.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetXFAFromString(const Source: AnsiString;
  Options: Integer): Integer;
```

DLL

```
int DPLSetXFAFromString(int InstanceID, char * Source, int Options);
```

Parameters

Source	The new XML string to store as the XFA form data.
Options	Reserved for future use. Should be set to 0.

Return values

0	The XFA form data could not be set, this usually means the document does not have an AcroForm dictionary.
1	The XFA form data was set successfully.

SetupCustomPrinter

Description

Changes the properties of a custom printer created with the **NewCustomPrinter** function.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SetupCustomPrinter(
  CustomPrinterName: WideString; Setting, NewValue: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SetupCustomPrinter(
  CustomPrinterName As String, Setting As Long,
  NewValue As Long) As Long
```

DLL

```
int DPLSetupCustomPrinter(int InstanceID, wchar_t * CustomPrinterName,
  int Setting, int NewValue);
```

Parameters

CustomPrinterName	A custom printer name, as returned by the **NewCustomPrinter** function
Setting	0 = Use Paper length and height and not Paper size 1 = Paper size 2 = Paper length 3 = Paper width 4 = Copies 5 = Print quality 6 = Color 7 = Duplex 8 = Collate 9 = Default source (paper trays / bins) 10 = Media type 11 = Orientation
NewValue	For custom paper size 0 Uses length and height to specify custom size in tenths or millimetres For paper size: 1 to 68, DMPAPER_XXX (Win32 API DEVMODE data structure) For paper height and width: Size of paper in tenths of millimetres For copies: Number of copies For print quality: 1 = high, 2 = medium, 3 = low, 4 = draft or an exact DPI, for example 600 For color: 1 = monochrome, 2 = color For duplex: 1 = simplex, 2 = vertical duplex, 3 = horizontal duplex For collate: 0 = no, 1 = yes For default source: 1 to 15, DMBIN_XXX (Win32 API DEVMODE data structure) 256 and higher for custom bins / paper trays, see the **GetPrinterBins** function For media type: 1 = standard, 2 = transparency, 3 = glossy 256 and higher for device-specific media For orientation: 1 = portrait, 2 = landscape

Return values

0	The custom printer could not be found, or the Settings or NewValue parameters were invalid
1	The custom printer settings were changed successfully

SignFile

Version history

This function was introduced in Quick PDF Library version 7.12.

Description

Applies a digital signature to a PDF document on disk.
The signing identity must be in PKCS#12 format containing a certificate and private key.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SignFile(InputFileName, OpenPassword,
  SignatureFieldName, OutputFileName, PFXFileName, PFXPassword, Reason,
  Location, ContactInfo: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SignFile(
  InputFileName As String, OpenPassword As String,
  SignatureFieldName As String, OutputFileName As String,
  PFXFileName As String, PFXPassword As String,
  Reason As String, Location As String,
  ContactInfo As String) As Long
```

DLL

```
int DPLSignFile(int InstanceID, wchar_t * InputFileName,
  wchar_t * OpenPassword, wchar_t * SignatureFieldName,
  wchar_t * OutputFileName, wchar_t * PFXFileName,
  wchar_t * PFXPassword, wchar_t * Reason, wchar_t * Location,
  wchar_t * ContactInfo);
```

Parameters

InputFileName	The path and file name of the input PDF to sign.
OpenPassword	The optional password to open the input PDF if it is encrypted
SignatureFieldName	The name of the signature field to sign. If a field with this name does not exist it will be created. This field cannot be blank.
OutputFileName	The path and file name of the signed PDF that should be created. This should be a different file to InputFileName. If in place signing is required (overwriting the original file) then this parameter should be left blank.
PFXFileName	The path and name of the PKCS#12 certificate/private key file (.pfx file).
PFXPassword	The password to open the PFX file.
Reason	An optional string indicating the reason for signing.
Location	An optional string indicating the location that the signing was done.
ContactInfo	An optional string indicating the contact information of the signer.

Return values

1	The file was signed successfully
2	Input PDF not found
3	Input PDF cannot be read
4	Input PDF password incorrect
5	Certificate file not found
6	Certificate file is invalid
7	Incorrect certificate password
8	Unknown certificate format
9	No private key found in certificate file
10	Could not write output file
11	Could not apply signature
12	The signature field name was blank

SplitPageText

Description

Splits the text and graphics on the current page into two layers. The graphics are placed into the bottom layer with the text in the top layer.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.SplitPageText(Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::SplitPageText(
  Options As Long) As Long
```

DLL

```
int DPLSplitPageText(int InstanceID, int Options);
```

Parameters

Options	This parameter is reserved for future use and should be set to zero

StartPath

Vector graphics, Path definition and drawing

Description

Starts a multi-segment path.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.StartPath(StartX, StartY: Double): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::StartPath(StartX As Double,
  StartY As Double) As Long
```

DLL

```
int DPLStartPath(int InstanceID, double StartX, double StartY);
```

Parameters

StartX	Horizontal co-ordinate of the point where the curve should start
StartY	Vertical co-ordinate of the point where the curve should start

StoreCustomDataFromFile

Description

Saves custom data from a file into the PDF under a key name. This data can later be retrieved using **RetrieveCustomDataToString** or **RetrieveCustomDataToFile**. The storage type (string, stream or compressed stream) and location (Document Information Dictionary or Document Catalog) can be set. If the location is the Document Catalog any storage type can be used, but the key must have a special prefix assigned to you by Adobe. If the location is the Document Information Dictionary any key apart from the standard keys can be used, but only strings can be used.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.StoreCustomDataFromFile(Key,
  FileName: WideString; Location, Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::StoreCustomDataFromFile(
  Key As String, FileName As String, Location As Long,
  Options As Long) As Long
```

DLL

```
int DPLStoreCustomDataFromFile(int InstanceID, wchar_t * Key,
  wchar_t * FileName, int Location, int Options);
```

Parameters

Key	The key to store the data under. If the location is the Document Information Dictionary then the key cannot be "Author", "Title", "Subject", "Keywords", "Creator" or "Producer". Any other key can be used but keys should be chosen with care so they make sense to the user. If the location is the Document Catalog then the key must have a special prefix assigned to you by Adobe to avoid conflicts with other software.
FileName	The path and name of the file containing the data to store in the PDF under the specified key.
Location	1 = Store the data in the Document Information Dictionary 2 = Store the data in the Document Catalog
Options	0 = Store the data as a string (the only option available if the location is the Document Information Dictionary) 1 = Store the data in a stream 2 = Store the data in a compressed stream

Return values

0	The file containing the data could not be opened, or the Key parameter was invalid
1	The data was stored successfully

StoreCustomDataFromString

Version history

This function was renamed in Quick PDF Library version 7.11.
The function name in earlier versions was StoreCustomData.

Description

Saves custom data into the PDF under a key name. This data can later be retrieved using the **RetrieveCustomDataToString**, **RetrieveCustomDataToVariant** or **RetrieveCustomDataToFile** functions. The storage type (string, stream or compressed stream) and location (Document Information Dictionary or Document Catalog) can be set. If the location is the Document Catalog any storage type can be used, but the key must have a special prefix assigned to you by Adobe. If the location is the Document Information Dictionary any key apart from the standard keys can be used, but only strings can be used.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.StoreCustomDataFromString(const Key,
  NewValue: AnsiString; Location, Options: Integer): Integer;
```

DLL

```
int DPLStoreCustomDataFromString(int InstanceID, char * Key,
  char * NewValue, int Location, int Options);
```

Parameters

Key	The key to store the data under. If the location is the Document Information Dictionary then the key cannot be "Author", "Title", "Subject", "Keywords", "Creator" or "Producer". Any other key can be used but keys should be chosen with care so they make sense to the user. If the location is the Document Catalog then the key must have a special prefix assigned to you by Adobe to avoid conflicts with other software.
NewValue	The new value for the data
Location	1 = Store the data in the Document Information Dictionary 2 = Store the data in the Document Catalog
Options	0 = Store the data as a string (the only option available if the location is the Document Information Dictionary) 1 = Store the data in a stream 2 = Store the data in a compressed stream

Return values

0	The data could not be stored because the key name was a reserved name
1	The data was stored successfully

StoreCustomDataFromVariant

Description

This function saves custom data, provided as a variant byte array, into the PDF under a key name. This data can later be retrieved using **RetrieveCustomDataToVariant**. The storage type (string, stream or compressed stream) and location (Document Information Dictionary or Document Catalog) can be set. If the location is the Document Catalog any storage type can be used, but the key must have a special prefix assigned to you by Adobe. If the location is the Document Information Dictionary any key apart from the standard keys can be used, but only strings can be used.

Syntax

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::StoreCustomDataFromVariant(
  Key As String, NewValue As Variant, Location As Long,
  Options As Long) As Long
```

Parameters

Key	The key to store the data under. If the location is the Document Information Dictionary then the key cannot be "Author", "Title", "Subject", "Keywords", "Creator" or "Producer". Any other key can be used but keys should be chosen with care so they make sense to the user. If the location is the Document Catalog then the key must have a special prefix assigned to you by Adobe to avoid conflicts with other software.
NewValue	A variant byte array containing the data to store in the PDF
Location	1 = Store the data in the Document Information Dictionary 2 = Store the data in the Document Catalog
Options	0 = Store the data as a string (the only option available if the location is the Document Information Dictionary) 1 = Store the data in a stream 2 = Store the data in a compressed stream

Return values

0	The Location parameter was invalid
1	The custom data was stored successfully

StringResultLength

Description

Returns the character length of the most recent string returned from the library by all functions that return Unicode (16-bit) strings.

The value returned is the number of 16-bit characters. So the total byte length will be twice that value.

A few functions return 8-bit strings, the **AnsiStringResultLength** function must be used to obtain the data length for those functions.

Syntax

DLL

```
int DPLStringResultLength(int InstanceID);
```

TestTempPath

Description

Tests that folder used for storage of temporary files has read/write access by the process running the library.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.TestTempPath: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::TestTempPath As Long
```

DLL

```
int DPLTestTempPath(int InstanceID);
```

Return values

0	The temporary path does not have read/write access
1	The temporary path is valid

TransformFile

Document manipulation, Miscellaneous functions

Version history

This function was introduced in Quick PDF Library version 9.11.

Description

Applies a transformation to a file allowing objects to be renumbered and reordered.

In certain cases this can result in a more compact cross reference table reducing the size of the PDF.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.TransformFile(InputFileName, Password,
  OutputFileName: WideString; TransformType, Options: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::TransformFile(
  InputFileName As String, Password As String,
  OutputFileName As String, TransformType As Long,
  Options As Long) As Long
```

DLL

```
int DPLTransformFile(int InstanceID, wchar_t * InputFileName,
  wchar_t * Password, wchar_t * OutputFileName,
  int TransformType, int Options);
```

Parameters

InputFileName	The path and file name of the input PDF to transform.
Password	The optional password to open the input PDF if it is encrypted
OutputFileName	The path and file name of the signed PDF that should be created. This should be different to InputFileName.
TransformType	1 = Renumber all objects writing them out in order 2 = Same as 1 but uses an xref stream
Options	Reserved for future use, should be set to zero.

Return values

1	Success
2	Input PDF not found
3	Input PDF cannot be read
4	Input PDF password incorrect
5	Could not write output file

UnlockKey

Description

Unlocks the library. The library must be unlocked using a registration key before it can be used.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.UnlockKey(LicenseKey: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::UnlockKey(
  LicenseKey As String) As Long
```

DLL

```
int DPLUnlockKey(int InstanceID, wchar_t * LicenseKey);
```

Parameters

LicenseKey	The registration key

Return values

0	The library could not be unlocked
1	The library was unlocked successfully

Unlocked

Miscellaneous functions

Description

Determine if the library has been unlocked. If the library has not been unlocked it cannot be used.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.Unlocked: Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::Unlocked As Long
```

DLL

```
int DPLUnlocked(int InstanceID);
```

Return values

0	The library has not been unlocked
1	The library has been unlocked

UpdateAndFlattenFormField

Form fields, Page layout

Version history

This function was introduced in Quick PDF Library version 9.11.

Description

Use this function to draw the visual appearance onto the page it is associated with. The form field will then be removed from the document and only it's appearance will remain - it will no longer be an interactive field.

If the field is flattened successfully the field index of subsequent form fields will be decreased by 1.

The appearance stream of the form field will be generated before the form field is flattened. This behaviour is the same as the **FlattenFormField** function before version 9.11.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.UpdateAndFlattenFormField(
   Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::UpdateAndFlattenFormField(
   Index As Long) As Long
```

DLL

```
int DPLUpdateAndFlattenFormField(int InstanceID, int Index);
```

Parameters

Index	The index of the form field to work with. The first form field has an index of 1.

Return values

0	The form field could not be found or it was not possible to flatten the form field
1	The form field was flattened successfully

UpdateAppearanceStream

Form fields

Description

Generates an appearance stream for the form field. Appearance streams can be generated for text, pushbutton and choice form fields.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.UpdateAppearanceStream(
  Index: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::UpdateAppearanceStream(
  Index As Long) As Long
```

DLL

```
int DPLUpdateAppearanceStream(int InstanceID, int Index);
```

Parameters

Index	The index of the form field to work with. The first form field has an index of 1.

Return values

0	The form field could not be found or an appearance stream could not be created for the specified field
1	The appearance stream for the specified form field was created successfully

UpdateTrueTypeSubsettedFont

Version history

This function was introduced in Quick PDF Library version 10.12.

Description

Updates the selected font with a new subset.

This can only be done if the font was originally created using **AddTrueTypeSubsettedFont** using Options 2, 3, 4 or 5.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.UpdateTrueTypeSubsettedFont(
  SubsetChars: WideString): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::UpdateTrueTypeSubsettedFont(
  SubsetChars As String) As Long
```

DLL

```
int DPLUpdateTrueTypeSubsettedFont(int InstanceID, wchar_t * SubsetChars);
```

Parameters

SubsetChars	The new list of characters to include in the font subset in addition to the existing characters.

Return values

0	Could not update the font subset
1	Success

UseKerning

Description

Specifies whether to use kerning for text subsequently drawn using the **DrawText** and **DrawRotatedText** functions.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.UseKerning(Kern: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::UseKerning(Kern As Long) As Long
```

DLL

```
int DPLUseKerning(int InstanceID, int Kern);
```

Parameters

Kern	0 = Do not use kerning 1 = Use kerning 2 = Do not attempt to load kerning from TrueType fonts subsequently added to the document

UseUnsafeContentStreams

Version history

This function was renamed in Quick PDF Library version 8.11.
The function name in earlier versions was UseUnsafeLayers.

Description

A page in a PDF document has one or more content stream parts that together contain all the PDF page description commands for the page.

This function specifies whether content stream parts that were not created by Quick PDF Library should be automatically re-used or not.

Syntax

Delphi

```
function TDebenuPDFLibrary1115.UseUnsafeContentStreams(
  SafetyLevel: Integer): Integer;
```

ActiveX

```
Function DebenuPDFLibrary1115.PDFLibrary::UseUnsafeContentStreams(
  SafetyLevel As Long) As Long
```

DLL

```
int DPLUseUnsafeContentStreams(int InstanceID, int SafetyLevel);
```

Parameters

SafetyLevel	0 = Only re-use existing Quick PDF Library content stream parts (Default) 1 = Re-use any content stream part

Return values

0	The SafetyLevel parameter was out of range
1	The safety level was set successfully

Appendix A: Supported HTML tags

A limited HTML subset is supported:

\<br\> to break onto a new line
\<b\> or \<strong\> for bold
\<i\> or \<em\> for italics
\<sup\> and \<sub\> for superscript/subscript.
\<u\> for underline
\<u style="double"\> for double underline
\<u style="strikeout"\> for strikeout (a line drawn through the text)
\<u style="over"\> for a line drawn above the text
\<p align="left"\> for left aligned paragraphs
\<p align="center"\> for centered paragraphs
\<p align="justified"\> for justified paragraphs
\<ul\>, \<ol\> and \<li\> for ordered/unordered lists
\<a href="http://..."\> for web links
\<a href="https://..."\> for web links
\<a href="file://..."\> for local file links

\<font
size="___"
color="___"
background="___"
roundback="yes/no"
mode="___"
outlinecolor="___"
outlinewidth="___pt"
\>
\<span background="___" roundback="yes/no"\>

The font size can be specified as a standard HTML size, or a point size such as "11.5pt", the outline width must be specified in points, for example "1.5pt".

Text and background colors can be specified in RGB using the standard HTML hexadecimal notation, for example "#3A498C". CMYK colors can be specified using eight hexadecimal values and omitting the #, for example "5C238F02".

If the roundback attribute is "yes", the background rectangles will be drawn with rounded edges.

Appendix B: Function groups

Annotations and hotspot links

AddArcToPath
AddFreeTextAnnotation
AddFreeTextAnnotationEx
AddLinkToDestination
AddLinkToEmbeddedFile
AddLinkToFile
AddLinkToFileDest
AddLinkToFileEx
AddLinkToJavaScript
AddLinkToLocalFile
AddLinkToPage
AddLinkToWeb
AddNoteAnnotation
AddRelativeLinkToFile
AddRelativeLinkToFileDest
AddRelativeLinkToFileEx
AddRelativeLinkToLocalFile
AddSVGAnnotationFromFile
AddSWFAnnotationFromFile
AddStampAnnotation
AddStampAnnotationFromImage
AddStampAnnotationFromImageID
AddTextMarkupAnnotation
AddU3DAnnotationFromFile
AnnotationCount
AttachAnnotToForm
CheckPageAnnots
CloneOutlineAction
DeleteAnnotation
DrawPostScriptXObject
FlattenAnnot
GetActionDest
GetActionType
GetActionURL
GetAnnotActionID
GetAnnotDblProperty
GetAnnotDest
GetAnnotEmbeddedFileName
GetAnnotEmbeddedFileToFile
GetAnnotEmbeddedFileToString
GetAnnotIntProperty
GetAnnotQuadCount
GetAnnotQuadPoints
GetAnnotSoundToFile
GetAnnotSoundToString
GetAnnotStrProperty
GetBaseURL
GetDestName
GetDestPage
GetDestType
GetDestValue
GetFormFieldActionID
GetNamedDestination
GetOutlineActionID
GetTabOrderMode
IsAnnotFormField

Annotations and hotspot links continued…

NewDestination
NewNamedDestination
SetActionURL
SetAnnotBorderColor
SetAnnotBorderStyle
SetAnnotContents
SetAnnotDblProperty
SetAnnotIntProperty
SetAnnotOptional
SetAnnotQuadPoints
SetAnnotRect
SetAnnotStrProperty
SetBaseURL
SetDestProperties
SetDestValue
SetMarkupAnnotStyle
SetOutlineNamedDestination
SetTabOrderMode

Barcodes

DrawBarcode
DrawDataMatrixSymbol
DrawIntelligentMailBarcode
DrawPDF417Symbol
DrawPDF417SymbolEx
DrawQRCode

Color

AddSeparationColor
DAGetTextBlockColor
DAGetTextBlockColorType
GetFormFieldBackgroundColor
GetFormFieldBackgroundColorType
GetFormFieldBorderColor
GetFormFieldBorderColorType
GetFormFieldColor
GetOutlineColor
GetPageColorSpaces
GetPageJavaScript
GetTextBlockColor
GetTextBlockColorType
ImageFillColor
NewRGBAxialShader
NewTilingPatternFromCapturedPage
SetAnnotBorderColor
SetFillColor
SetFillColorCMYK
SetFillColorSep
SetFillShader
SetFillTilingPattern
SetFormFieldBackgroundColor
SetFormFieldBackgroundColorCMYK
SetFormFieldBackgroundColorGray
SetFormFieldBackgroundColorSep

Color continued…

SetFormFieldBorderColor
SetFormFieldBorderColorCMYK
SetFormFieldBorderColorGray
SetFormFieldBorderColorSep
SetFormFieldColor
SetFormFieldColorCMYK
SetFormFieldColorSep
SetImageMaskCMYK
SetLineColor
SetLineColorCMYK
SetLineColorSep
SetLineShader
SetMarkupAnnotStyle
SetOutlineColor
SetPNGTransparencyColor
SetTableBorderColor
SetTableBorderColorCMYK
SetTableCellBackgroundColor
SetTableCellBackgroundColorCMYK
SetTableCellBorderColor
SetTableCellBorderColorCMYK
SetTableCellTextColor
SetTableCellTextColorCMYK
SetTextColor
SetTextColorCMYK
SetTextColorSep
SetTextHighlightColor
SetTextHighlightColorCMYK
SetTextHighlightColorSep
SetTextShader
SetTextUnderlineColor
SetTextUnderlineColorCMYK
SetTextUnderlineColorSep
SetXFAFormFieldBorderColor

Content Streams and Optional Content Groups

BalanceContentStream
CombineContentStreams
ContentStreamCount
ContentStreamSafe
DeleteContentStream
DeleteOptionalContentGroup
EditableContentStream
EncapsulateContentStream
GetContentStreamToString
GetContentStreamToVariant
GetOptionalContentConfigCount
GetOptionalContentConfigLocked
GetOptionalContentConfigOrderCount
GetOptionalContentConfigOrderItemID
GetOptionalContentConfigOrderItemLabe
GetOptionalContentConfigOrderItemLevel
GetOptionalContentConfigOrderItemType
GetOptionalContentConfigState

Content Streams and Optional Content Groups continued…

GetOptionalContentGroupID
GetOptionalContentGroupName
GetOptionalContentGroupPrintable
GetOptionalContentGroupVisible
MoveContentStream
NewContentStream
NewOptionalContentGroup
OptionalContentGroupCount
RemoveSharedContentStreams
SelectContentStream
SetAnnotOptional
SetCapturedPageOptional
SetCapturedPageTransparencyGroup
SetContentStreamFromString
SetContentStreamFromVariant
SetContentStreamOptional
SetFormFieldOptional
SetImageOptional
SetOptionalContentConfigLocked
SetOptionalContentConfigState
SetOptionalContentGroupName
SetOptionalContentGroupPrintable
SetOptionalContentGroupVisible
UseUnsafeContentStreams

Direct access functionality

DAAppendFile
DACapturePage
DACapturePageEx
DACloseFile
DADrawCapturedPage
DADrawRotatedCapturedPage
DAEmbedFileStreams
DAExtractPageText
DAExtractPageTextBlocks
DAFindPage
DAGetAnnotationCount
DAGetFormFieldCount
DAGetFormFieldTitle
DAGetFormFieldValue
DAGetImageDataToString
DAGetImageDataToVariant
DAGetImageDblProperty
DAGetImageIntProperty
DAGetImageListCount
DAGetInformation
DAGetObjectCount
DAGetObjectToString
DAGetObjectToVariant
DAGetPageBox
DAGetPageContentToString
DAGetPageContentToVariant
DAGetPageCount
DAGetPageHeight

Direct access functionality continued…

DAGetPageImageList
DAGetPageWidth
DAGetTextBlockAsString
DAGetTextBlockBound
DAGetTextBlockCharWidth
DAGetTextBlockColor
DAGetTextBlockColorType
DAGetTextBlockCount
DAGetTextBlockFontName
DAGetTextBlockFontSize
DAGetTextBlockText
DAHasPageBox
DAHidePage
DAMovePage
DANewPage
DANewPages
DANormalizePage
DAOpenFile
DAOpenFileReadOnly
DAOpenFromStream
DAPageRotation
DAReleaseImageList
DAReleaseTextBlocks
DARemoveUsageRights
DARenderPageToDC
DARenderPageToFile
DARenderPageToStream
DARenderPageToString
DARenderPageToVariant
DARotatePage
DASaveAsFile
DASaveCopyToStream
DASaveImageDataToFile
DASaveToStream
DASetInformation
DASetPageBox
DASetPageLayout
DASetPageMode
DASetPageSize
DASetTextExtractionArea
DASetTextExtractionOptions
DASetTextExtractionScaling
DASetTextExtractionWordGap
DAShiftedHeader

Document management

AppendToFile
AppendToString
AppendToVariant
BalancePageTree
DAAppendFile
DAOpenFile
DAOpenFileReadOnly
DAOpenFromStream
DASaveAsFile

Document management continued…

DASaveCopyToStream
DASaveToStream
DAShiftedHeader
DecryptFile
DocumentCount
GetCanvasDC
GetCanvasDCEx
GetDocumentFileName
GetDocumentID
GetDocumentRepaired
InsertPages
LoadFromCanvasDC
LoadFromFile
LoadFromStream
LoadFromString
LoadFromVariant
MovePage
NewDestination
NewDocument
RemoveDocument
SaveToFile
SaveToStream
SaveToString
SaveToVariant
SelectDocument
SelectedDocument
SetAppendInputFromString
SetAppendInputFromVariant
SetFindImagesMode

Document manipulation

CheckFileCompliance
DAEmbedFileStreams
DANormalizePage
DARemoveUsageRights
ExtractFilePages
ExtractFilePagesEx
ExtractPageRanges
LinearizeFile
MergeDocument
MergeFileList
MergeFileListFast
MergeFiles
MergeStreams
RemoveUsageRights
ReplaceFonts
TransformFile

Document properties

AddEmbeddedFile
AddFileAttachment
AddGlobalJavaScript
AddLinkToEmbeddedFile
AnalyseFile

Document properties continued…

CompressContent
CompressFonts
CompressImages
DAGetInformation
DAGetPageCount
DASetInformation
DASetPageLayout
DASetPageMode
Decrypt
DeleteAnalysis
DocJavaScriptAction
EmbedFile
EmbeddedFileCount
EncryptionAlgorithm
EncryptionStatus
EncryptionStrength
FindFonts
FindImages
GetAnalysisInfo
GetBaseURL
GetCatalogInformation
GetCustomInformation
GetCustomKeys
GetDocJavaScript
GetDocumentFileSize
GetDocumentIdentifier
GetDocumentMetadata
GetDocumentRepaired
GetDocumentResourceList
GetEmbeddedFileContentToFile
GetEmbeddedFileContentToStream
GetEmbeddedFileContentToString
GetEmbeddedFileContentToVariant
GetEmbeddedFileID
GetEmbeddedFileIntProperty
GetEmbeddedFileStrProperty
GetEncryptionFingerprint
GetFileMetadata
GetGlobalJavaScript
GetInformation
GetMaxObjectNumber
GetNamedDestination
GetOpenActionDestination
GetOpenActionJavaScript
GetPageLayout
GetPageMode
GetViewerPreferences
GlobalJavaScriptCount
GlobalJavaScriptPackageName
HasFontResources
ImageCount
IsLinearized
NewPostScriptXObject
PageCount
RemoveCustomInformation
RemoveEmbeddedFile

Document properties continued…

RemoveGlobalJavaScript
RemoveOpenAction
RemoveUsageRights
RemoveXFAEntries
RetrieveCustomDataToFile
RetrieveCustomDataToString
RetrieveCustomDataToVariant
SecurityInfo
SetBaseURL
SetCatalogInformation
SetCustomInformation
SetDecodeMode
SetDocumentMetadata
SetEmbeddedFileStrProperty
SetHeaderCommentsFromString
SetHeaderCommentsFromVariant
SetInformation
SetJavaScriptMode
SetOpenActionDestination
SetOpenActionDestinationFull
SetOpenActionJavaScript
SetOpenActionMenu
SetPDFAMode
SetPageLayout
SetPageMode
SetViewerPreferences
StoreCustomDataFromFile
StoreCustomDataFromString
StoreCustomDataFromVariant

Extraction

CopyPageRanges
CopyPageRangesEx
DAExtractPageText
DAExtractPageTextBlocks
DAGetTextBlockAsString
DAGetTextBlockBound
DAGetTextBlockCharWidth
DAGetTextBlockColor
DAGetTextBlockColorType
DAGetTextBlockCount
DAGetTextBlockFontName
DAGetTextBlockFontSize
DAGetTextBlockText
DASetTextExtractionArea
DASetTextExtractionOptions
DASetTextExtractionScaling
DASetTextExtractionWordGap
ExtractFilePageContentToString
ExtractFilePageContentToVariant
ExtractFilePageText
ExtractFilePageTextBlocks
ExtractFilePages
ExtractFilePagesEx
ExtractPageRanges

Extraction continued…

ExtractPageTextBlocks
ExtractPages
GetPageText
GetTextBlockAsString
GetTextBlockBound
GetTextBlockCharWidth
GetTextBlockColor
GetTextBlockColorType
GetTextBlockCount
GetTextBlockFontName
GetTextBlockFontSize
GetTextBlockText
ReleaseTextBlocks
SetTextExtractionArea
SetTextExtractionOptions
SetTextExtractionScaling
SetTextExtractionWordGap

Fonts

AddCJKFont
AddFormFont
AddOpenTypeFontFromFile
AddOpenTypeFontFromString
AddStandardFont
AddSubsettedFont
AddTrueTypeFont
AddTrueTypeFontFromFile
AddTrueTypeFontFromString
AddTrueTypeSubsettedFont
AddType1Font
CharWidth
CompressFonts
DAGetTextBlockCharWidth
DAGetTextBlockFontName
DAGetTextBlockFontSize
FindFonts
FontCount
FontFamily
FontHasKerning
FontName
FontReference
FontSize
FontType
GetFontEncoding
GetFontFlags
GetFontID
GetFontIsEmbedded
GetFontIsSubsetted
GetFontMetrics
GetFontObjectNumber
GetFormFontCount
GetFormFontName
GetInstalledFontsByCharset
GetInstalledFontsByCodePage
GetKerning

Fonts continued…

GetTextAscent
GetTextBlockBound
GetTextBlockCharWidth
GetTextBlockFontName
GetTextBlockFontSize
GetTextBound
GetTextDescent
GetTextHeight
GetTextSize
GetTextWidth
GetUnicodeCharactersFromEncoding
HasFontResources
NoEmbedFontListAdd
NoEmbedFontListCount
NoEmbedFontListGet
NoEmbedFontListRemoveAll
NoEmbedFontListRemoveIndex
NoEmbedFontListRemoveName
ReplaceFonts
SaveFontToFile
SelectFont
SelectedFont
SetFontEncoding
SetFontFlags
SetFormFieldStandardFont
SetKerning
UpdateTrueTypeSubsettedFont
UseKerning

Form fields

AddArcToPath
AddFormFieldChoiceSub
AddFormFieldSub
AddFormFont
AttachAnnotToForm
DAGetFormFieldCount
DAGetFormFieldTitle
DAGetFormFieldValue
DeleteFormField
FindFormFieldByTitle
FlattenFormField
FormFieldCount
FormFieldHasParent
FormFieldJavaScriptAction
FormFieldWebLinkAction
GetFormFieldActionID
GetFormFieldAlignment
GetFormFieldAnnotFlags
GetFormFieldBackgroundColor
GetFormFieldBackgroundColorType
GetFormFieldBorderColor
GetFormFieldBorderColorType
GetFormFieldBorderProperty
GetFormFieldBorderStyle
GetFormFieldBound

Form fields continued…

GetFormFieldCaption
GetFormFieldCaptionEx
GetFormFieldCheckStyle
GetFormFieldChildTitle
GetFormFieldChoiceType
GetFormFieldColor
GetFormFieldComb
GetFormFieldDefaultValue
GetFormFieldDescription
GetFormFieldFlags
GetFormFieldFontName
GetFormFieldJavaScript
GetFormFieldKidCount
GetFormFieldKidTempIndex
GetFormFieldMaxLen
GetFormFieldNoExport
GetFormFieldPage
GetFormFieldPrintable
GetFormFieldReadOnly
GetFormFieldRequired
GetFormFieldRichTextString
GetFormFieldRotation
GetFormFieldSubCount
GetFormFieldSubDisplayName
GetFormFieldSubName
GetFormFieldSubmitActionString
GetFormFieldTabOrder
GetFormFieldTabOrderEx
GetFormFieldTextFlags
GetFormFieldTextSize
GetFormFieldTitle
GetFormFieldType
GetFormFieldValue
GetFormFieldValueByTitle
GetFormFieldVisible
GetFormFieldWebLink
GetFormFontCount
GetFormFontName
GetTabOrderMode
GetXFAFormFieldCount
GetXFAFormFieldName
GetXFAFormFieldNames
GetXFAFormFieldValue
GetXFAToString
IsAnnotFormField
NewChildFormField
NewFormField
RemoveAppearanceStream
RemoveFormFieldBackgroundColor
RemoveFormFieldBorderColor
RemoveFormFieldChoiceSub
RemoveXFAEntries
SetCharWidth
SetFormFieldAlignment
SetFormFieldAnnotFlags
SetFormFieldBackgroundColor

Form fields continued…

SetFormFieldBackgroundColorCMYK
SetFormFieldBackgroundColorGray
SetFormFieldBackgroundColorSep
SetFormFieldBorderColor
SetFormFieldBorderColorCMYK
SetFormFieldBorderColorGray
SetFormFieldBorderColorSep
SetFormFieldBorderStyle
SetFormFieldBounds
SetFormFieldCalcOrder
SetFormFieldCaption
SetFormFieldCheckStyle
SetFormFieldChildTitle
SetFormFieldChoiceSub
SetFormFieldChoiceType
SetFormFieldColor
SetFormFieldColorCMYK
SetFormFieldColorSep
SetFormFieldComb
SetFormFieldDefaultValue
SetFormFieldDescription
SetFormFieldFlags
SetFormFieldFont
SetFormFieldHighlightMode
SetFormFieldIcon
SetFormFieldIconStyle
SetFormFieldMaxLen
SetFormFieldNoExport
SetFormFieldOptional
SetFormFieldPage
SetFormFieldPrintable
SetFormFieldReadOnly
SetFormFieldRequired
SetFormFieldResetAction
SetFormFieldRichTextString
SetFormFieldRotation
SetFormFieldSignatureImage
SetFormFieldStandardFont
SetFormFieldSubmitAction
SetFormFieldSubmitActionEx
SetFormFieldTabOrder
SetFormFieldTextFlags
SetFormFieldTextSize
SetFormFieldTitle
SetFormFieldValue
SetFormFieldValueByTitle
SetFormFieldVisible
SetNeedAppearances
SetTabOrderMode
SetXFAFormFieldAccess
SetXFAFormFieldBorderColor
SetXFAFormFieldBorderPresence
SetXFAFormFieldBorderWidth
SetXFAFormFieldValue
SetXFAFromString
UpdateAndFlattenFormField

Form fields continued…

UpdateAppearanceStream

HTML text

DrawHTMLText
DrawHTMLTextBox
DrawHTMLTextBoxMatrix
DrawHTMLTextMatrix
GetHTMLTextHeight
GetHTMLTextLineCount
GetHTMLTextWidth
SetHTMLBoldFont
SetHTMLBoldItalicFont
SetHTMLItalicFont
SetHTMLNormalFont

Image handling

AddImageFromFile
AddImageFromFileOffset
AddImageFromStream
AddImageFromString
AddImageFromVariant
AddSVGAnnotationFromFile
AddSWFAnnotationFromFile
AddU3DAnnotationFromFile
ClearImage
CompressImages
DAGetImageDataToString
DAGetImageDataToVariant
DAGetImageDblProperty
DAGetImageIntProperty
DAGetImageListCount
DAGetPageImageList
DAReleaseImageList
DASaveImageDataToFile
DrawImage
DrawImageMatrix
DrawRotatedImage
DrawScaledImage
FindImages
FitImage
GetImageID
GetImageListCount
GetImageListItemDataToString
GetImageListItemDataToVariant
GetImageListItemDblProperty
GetImageListItemFormatDesc
GetImageListItemIntProperty
GetImagePageCount
GetImagePageCountFromString
GetPageImageList
ImageCount
ImageFillColor
ImageHeight
ImageHorizontalResolution

Image handling continued…

ImageResolutionUnits
ImageType
ImageVerticalResolution
ImageWidth
ImportEMFFromFile
ImportEMFFromStream
ReleaseImage
ReleaseImageList
RenderAsMultipageTIFFToFile
ReplaceImage
ReverseImage
SaveImageListItemDataToFile
SaveImageToFile
SaveImageToStream
SaveImageToString
SaveImageToVariant
SelectImage
SelectedImage
SetBlendMode
SetFindImagesMode
SetFormFieldSignatureImage
SetImageAsMask
SetImageMask
SetImageMaskCMYK
SetImageMaskFromImage
SetImageOptional
SetImageResolution
SetPNGTransparencyColor

JavaScript

AddGlobalJavaScript
AddLinkToJavaScript
DocJavaScriptAction
FormFieldJavaScriptAction
GetDocJavaScript
GetGlobalJavaScript
GetOpenActionJavaScript
GetOutlineJavaScript
GetPageJavaScript
GlobalJavaScriptCount
GlobalJavaScriptPackageName
PageJavaScriptAction
RemoveGlobalJavaScript
SetJavaScriptMode
SetOpenActionJavaScript
SetOutlineJavaScript

Measurement and coordinate units

AddLGIDictToPage
DeletePageLGIDict
GetCSDictEPSG
GetCSDictType
GetCSDictWKT
GetImageMeasureDict

Measurement and coordinate units
continued…

GetImagePtDataDict
GetMeasureDictBoundsCount
GetMeasureDictBoundsItem
GetMeasureDictCoordinateSystem
GetMeasureDictDCSDict
GetMeasureDictGCSDict
GetMeasureDictGPTSCount
GetMeasureDictGPTSItem
GetMeasureDictLPTSCount
GetMeasureDictLPTSItem
GetMeasureDictPDU
GetOrigin
GetPageLGIDictContent
GetPageLGIDictCount
GetPageViewPortCount
GetPageViewPortID
GetViewPortBBox
GetViewPortMeasureDict
GetViewPortName
GetViewPortPtDataDict
MultiplyScale
SetCSDictEPSG
SetCSDictType
SetCSDictWKT
SetMeasureDictBoundsCount
SetMeasureDictBoundsItem
SetMeasureDictCoordinateSystem
SetMeasureDictGPTSCount
SetMeasureDictGPTSItem
SetMeasureDictLPTSCount
SetMeasureDictLPTSItem
SetMeasureDictPDU
SetMeasurementUnits
SetOrigin
SetPrecision
SetScale

Miscellaneous functions

AddToBuffer
AddToFileList
AnsiStringResultLength
CheckObjects
CheckPageAnnots
ClearFileList
CreateBuffer
CreateLibrary
CreateNewObject
DAGetObjectCount
DAGetObjectToString
DAGetObjectToVariant
EncodeStringFromVariant
FileListCount
FileListItem
GetImagePageCount

Miscellaneous functions continued…

GetImagePageCountFromString
GetMaxObjectNumber
GetObjectCount
GetObjectDecodeError
GetObjectToString
GetObjectToVariant
GetStringListCount
GetStringListItem
GetTempPath
GetUnicodeCharactersFromEncoding
LastErrorCode
LastRenderError
LibraryVersion
LibraryVersionEx
LicenseInfo
LinearizeFile
NoEmbedFontListAdd
NoEmbedFontListCount
NoEmbedFontListGet
NoEmbedFontListRemoveAll
NoEmbedFontListRemoveIndex
NoEmbedFontListRemoveName
ReleaseBuffer
ReleaseLibrary
ReleaseStringList
SetAnsiMode
SetCairoFileName
SetCompatibility
SetDPLRFileName
SetObjectFromString
SetObjectFromVariant
SetTempFile
SetTempPath
StringResultLength
TestTempPath
TransformFile
UnlockKey
Unlocked

Outlines

CloneOutlineAction
CloseOutline
CompareOutlines
GetFirstChildOutline
GetFirstOutline
GetNextOutline
GetOutlineActionID
GetOutlineColor
GetOutlineDest
GetOutlineID
GetOutlineJavaScript
GetOutlineObjectNumber
GetOutlineOpenFile
GetOutlinePage
GetOutlineStyle

Outlines continued…

GetOutlineWebLink
GetParentOutline
GetPrevOutline
MoveOutlineAfter
MoveOutlineBefore
NewOutline
NewStaticOutline
OpenOutline
OutlineCount
OutlineTitle
RemoveOutline
SetOutlineColor
SetOutlineDestination
SetOutlineDestinationFull
SetOutlineDestinationZoom
SetOutlineJavaScript
SetOutlineNamedDestination
SetOutlineOpenFile
SetOutlineRemoteDestination
SetOutlineStyle
SetOutlineTitle
SetOutlineWebLink

Page layout

AddSVGAnnotationFromFile
AddSWFAnnotationFromFile
AddU3DAnnotationFromFile
AppendSpace
AppendTableColumns
AppendTableRows
AppendText
ApplyStyle
BeginPageUpdate
CreateTable
DADrawCapturedPage
DADrawRotatedCapturedPage
DrawCapturedPage
DrawCapturedPageMatrix
DrawHTMLText
DrawHTMLTextBox
DrawHTMLTextBoxMatrix
DrawHTMLTextMatrix
DrawImage
DrawImageMatrix
DrawMultiLineText
DrawPostScriptXObject
DrawRotatedCapturedPage
DrawRotatedImage
DrawRotatedMultiLineText
DrawRotatedText
DrawRotatedTextBox
DrawRotatedTextBoxEx
DrawRoundedBox
DrawRoundedRotatedBox
DrawScaledImage

Page layout continued…

DrawSpacedText
DrawTableRows
DrawText
DrawTextArc
DrawTextBox
DrawTextBoxMatrix
DrawWrappedText
EndPageUpdate
FitImage
FitRotatedTextBox
FitTextBox
FlattenAnnot
FlattenFormField
GetBarcodeWidth
GetTableCellDblProperty
GetTableCellIntProperty
GetTableCellStrProperty
GetTableColumnCount
GetTableLastDrawnRow
GetTableRowCount
GetTextAscent
GetTextBound
GetTextDescent
GetTextHeight
GetTextSize
GetTextWidth
GetWrappedText
GetWrappedTextHeight
GetWrappedTextLineCount
ImageFillColor
InsertTableColumns
InsertTableRows
LoadState
MergeTableCells
ReplaceImage
SaveState
SaveStyle
SelectImage
SelectPage
SelectedImage
SelectedPage
SetCapturedPageOptional
SetCapturedPageTransparencyGroup
SetImageAsMask
SetImageMask
SetImageMaskCMYK
SetImageMaskFromImage
SetOverprint
SetPageContentFromString
SetPageContentFromVariant
SetPageDimensions
SetPageSize
SetPageTransparencyGroup
SetTableBorderColor
SetTableBorderColorCMYK
SetTableBorderWidth

Page layout continued…

SetTableCellAlignment
SetTableCellBackgroundColor
SetTableCellBackgroundColorCMYK
SetTableCellBorderColor
SetTableCellBorderColorCMYK
SetTableCellBorderWidth
SetTableCellContent
SetTableCellPadding
SetTableCellTextColor
SetTableCellTextColorCMYK
SetTableCellTextSize
SetTableColumnWidth
SetTableRowHeight
SetTableThinBorders
SetTableThinBordersCMYK
SetTransparency
UpdateAndFlattenFormField

Page manipulation

AddPageMatrix
BalanceContentStream
CapturePage
CapturePageEx
ClonePages
CopyPageRanges
CopyPageRangesEx
DACapturePage
DACapturePageEx
DAExtractPageText
DAHidePage
DAMovePage
DANewPage
DANewPages
DANormalizePage
DeletePages
DrawBox
DrawRotatedBox
DrawRotatedCapturedPage
ExtractFilePageContentToString
ExtractFilePageContentToVariant
ExtractFilePages
ExtractFilePagesEx
ExtractPageRanges
ExtractPages
GetContentStreamToString
GetContentStreamToVariant
GetPageContentToString
GetPageContentToVariant
GetPageText
HidePage
InsertPages
MovePage
NewPage
NewPageFromCanvasDC
NewPages

Page manipulation continued…

NormalizePage
ReplaceTag
RotatePage
SelectPage
SelectedPage
SetContentStreamFromString
SetContentStreamFromVariant
SetPageContentFromString
SetPageContentFromVariant
SetPageThumbnail
SplitPageText

Page properties

AddLGIDictToPage
AddLinkToDestination
AddLinkToPage
AddPageLabels
BalancePageTree
ClearPageLabels
CompressPage
DAGetPageBox
DAGetPageContentToString
DAGetPageContentToVariant
DAGetPageHeight
DAGetPageImageList
DAGetPageWidth
DAHasPageBox
DAPageRotation
DAReleaseImageList
DARotatePage
DASetPageBox
DASetPageSize
DeletePageLGIDict
ExtractFilePageText
ExtractFilePageTextBlocks
GetContentStreamToString
GetContentStreamToVariant
GetPageBox
GetPageColorSpaces
GetPageContentToString
GetPageContentToVariant
GetPageImageList
GetPageJavaScript
GetPageLGIDictContent
GetPageLGIDictCount
GetPageLabel
GetPageMetricsToString
GetPageUserUnit
GetPageViewPortCount
GetPageViewPortID
GetViewPortBBox
GetViewPortMeasureDict
GetViewPortName
GetViewPortPtDataDict
HasPageBox

Page properties continued…

HidePage
PageHasFontResources
PageHeight
PageJavaScriptAction
PageRotation
PageWidth
ReleaseImageList
RemovePageBox
RotatePage
SetContentStreamFromString
SetContentStreamFromVariant
SetCropBox
SetFindImagesMode
SetPageActionMenu
SetPageBox
SetPageContentFromString
SetPageContentFromVariant
SetPageDimensions
SetPageSize
SetPageUserUnit

Path definition and drawing

AddArcToPath
AddBoxToPath
AddCurveToPath
AddLineToPath
ClosePath
DrawPath
DrawPathEvenOdd
MovePath
SetClippingPath
SetClippingPathEvenOdd
SetFillShader
SetLineShader
SetTextShader
StartPath

Rendering and printing

DARenderPageToDC
DARenderPageToFile
DARenderPageToStream
DARenderPageToString
DARenderPageToVariant
GetDefaultPrinterName
GetLatestPrinterNames
GetPrintPreviewBitmapToString
GetPrintPreviewBitmapToVariant
GetPrinterBins
GetPrinterDevModeToString
GetPrinterDevModeToVariant
GetPrinterMediaTypes
GetPrinterNames
GetRenderScale
LastRenderError

Rendering and printing continued…

NewCustomPrinter
NewInternalPrinterObject
PrintDocument
PrintDocumentToFile
PrintDocumentToPrinterObject
PrintMode
PrintOptions
PrintPages
PrintPagesToFile
PrintPagesToPrinterObject
RenderAsMultipageTIFFToFile
RenderDocumentToFile
RenderPageToDC
RenderPageToDCClip
RenderPageToFile
RenderPageToStream
RenderPageToString
RenderPageToVariant
RequestPrinterStatus
SelectRenderer
SetGDIPlusFileName
SetGDIPlusOptions
SetJPEGQuality
SetPrinterDevModeFromString
SetPrinterDevModeFromVariant
SetRenderCropType
SetRenderDCErasePage
SetRenderDCOffset
SetRenderOptions
SetRenderScale
SetupCustomPrinter

Security and Signatures

CheckPassword
Decrypt
DecryptFile
EncodePermissions
Encrypt
EncryptFile
EncryptWithFingerprint
EncryptionAlgorithm
EncryptionStatus
EncryptionStrength
EndSignProcessToFile
EndSignProcessToStream
EndSignProcessToString
GetEncryptionFingerprint
GetSignProcessByteRange
GetSignProcessResult
NewSignProcessFromFile
NewSignProcessFromStream
NewSignProcessFromString
ReleaseSignProcess
SecurityInfo
SetFormFieldSignatureImage

Security and Signatures continued...

SetSignProcessCustomDict
SetSignProcessCustomSubFilter
SetSignProcessField
SetSignProcessFieldBounds
SetSignProcessFieldImageFromFile
SetSignProcessFieldImageFromStream
SetSignProcessFieldImageFromString
SetSignProcessFieldPage
SetSignProcessInfo
SetSignProcessKeyset
SetSignProcessPFXFromFile
SetSignProcessPFXFromStream
SetSignProcessPFXFromString
SetSignProcessPassthrough
SetSignProcessSubFilter
SignFile

Text

AddCJKFont
AddFreeTextAnnotation
AddFreeTextAnnotationEx
AddOpenTypeFontFromFile
AddOpenTypeFontFromString
AddStandardFont
AddSubsettedFont
AddTrueTypeFont
AddTrueTypeFontFromFile
AddTrueTypeFontFromString
AddTrueTypeSubsettedFont
AddType1Font
AppendSpace
AppendText
ApplyStyle
CharWidth
ClearTextFormatting
DAExtractPageTextBlocks
DAGetTextBlockAsString
DAGetTextBlockBound
DAGetTextBlockCharWidth
DAGetTextBlockColor
DAGetTextBlockColorType
DAGetTextBlockCount
DAGetTextBlockFontName
DAGetTextBlockFontSize
DAGetTextBlockText
DANormalizePage
DASetTextExtractionArea
DASetTextExtractionOptions
DASetTextExtractionScaling
DASetTextExtractionWordGap
DrawHTMLText
DrawHTMLTextBox
DrawHTMLTextBoxMatrix
DrawMultiLineText
DrawRotatedMultiLineText

Text continued...

DrawRotatedText
DrawRotatedTextBox
DrawRotatedTextBoxEx
DrawSpacedText
DrawText
DrawTextArc
DrawTextBox
DrawTextBoxMatrix
DrawWrappedText
EncodeStringFromVariant
ExtractFilePageTextBlocks
ExtractPageTextBlocks
FitRotatedTextBox
FitTextBox
FontHasKerning
FontSize
GetFontID
GetHTMLTextHeight
GetHTMLTextLineCount
GetHTMLTextWidth
GetKerning
GetTextAscent
GetTextBlockAsString
GetTextBlockBound
GetTextBlockCharWidth
GetTextBlockColor
GetTextBlockColorType
GetTextBlockCount
GetTextBlockFontName
GetTextBlockFontSize
GetTextBlockText
GetTextBound
GetTextDescent
GetTextHeight
GetTextSize
GetTextWidth
GetUnicodeCharactersFromEncoding
GetWrappedText
GetWrappedTextBreakString
GetWrappedTextHeight
GetWrappedTextLineCount
NormalizePage
ReleaseTextBlocks
RemoveStyle
SaveStyle
SelectFont
SelectedFont
SetBlendMode
SetBreakString
SetCharWidth
SetFormFieldTextSize
SetHTMLBoldFont
SetHTMLBoldItalicFont
SetHTMLItalicFont
SetHTMLNormalFont
SetKerning

Text continued…

SetPageTransparencyGroup
SetTextAlign
SetTextCharSpacing
SetTextColor
SetTextColorCMYK
SetTextColorSep
SetTextExtractionArea
SetTextExtractionOptions
SetTextExtractionScaling
SetTextExtractionWordGap
SetTextHighlight
SetTextHighlightColor
SetTextHighlightColorCMYK
SetTextHighlightColorSep
SetTextMode
SetTextRise
SetTextScaling
SetTextSize
SetTextSpacing
SetTextUnderline
SetTextUnderlineColor
SetTextUnderlineColorCMYK
SetTextUnderlineColorSep
SetTextUnderlineCustomDash
SetTextUnderlineDash
SetTextUnderlineDistance
SetTextUnderlineWidth
SetTextWordSpacing
SetTransparency
UpdateTrueTypeSubsettedFont
UseKerning

Vector graphics

AddArcToPath
AddBoxToPath
AddCurveToPath
AddLineToPath
AddSVGAnnotationFromFile
AddSWFAnnotationFromFile
AddSeparationColor
AddU3DAnnotationFromFile
ClosePath
DrawArc
DrawBarcode
DrawBox
DrawCircle
DrawDataMatrixSymbol
DrawEllipse
DrawEllipticArc
DrawIntelligentMailBarcode
DrawLine
DrawPDF417Symbol
DrawPDF417SymbolEx
DrawPath
DrawPathEvenOdd

Vector graphics continued…

DrawQRCode
DrawRotatedBox
DrawRoundedBox
DrawRoundedRotatedBox
GetBarcodeWidth
GetCanvasDC
GetCanvasDCEx
ImportEMFFromFile
ImportEMFFromStream
LoadFromCanvasDC
LoadState
MovePath
NewPageFromCanvasDC
NewRGBAxialShader
NewTilingPatternFromCapturedPage
NoEmbedFontListAdd
NoEmbedFontListCount
NoEmbedFontListGet
NoEmbedFontListRemoveAll
NoEmbedFontListRemoveIndex
NoEmbedFontListRemoveName
SaveState
SetBlendMode
SetClippingPath
SetClippingPathEvenOdd
SetCustomLineDash
SetFillColor
SetFillColorCMYK
SetFillColorSep
SetFillShader
SetFillTilingPattern
SetLineCap
SetLineColor
SetLineColorCMYK
SetLineColorSep
SetLineDash
SetLineDashEx
SetLineJoin
SetLineShader
SetLineWidth
SetOverprint
SetPageTransparencyGroup
SetTextShader
SetTransparency
StartPath

AddImageFromFile

AddLinkToWeb

AddStandardFont

DeletePages

DocumentCount

DrawHTMLText

DrawHTMLTextBox

DrawImage

DrawQRCode

DrawText

DrawTextBox

ExtractPages

FindImages

GetImageListCount

GetImageListItemFormatDesc

GetInformation

GetPageBox

GetPageImageList

HasFontResources

ImageCount

ImageHeight

ImageWidth

IsLinearized

LastErrorCode

LoadFromFile

MergeDocument

NewDocument

NewPage

NormalizePage

PageCount

PageHasFontResources

PageHeight

PageRotation

PageWidth

RemoveDocument

RotatePage

SaveToFile

SecurityInfo

SelectDocument

SelectedDocument

SelectFont

SelectImage

SelectPage

SetBaseURL

SetInformation

SetMeasurementUnits

SetOrigin

SetPageBox