MAKO™

API Reference

Version 7.3.0

1 Version X.X API Documentation

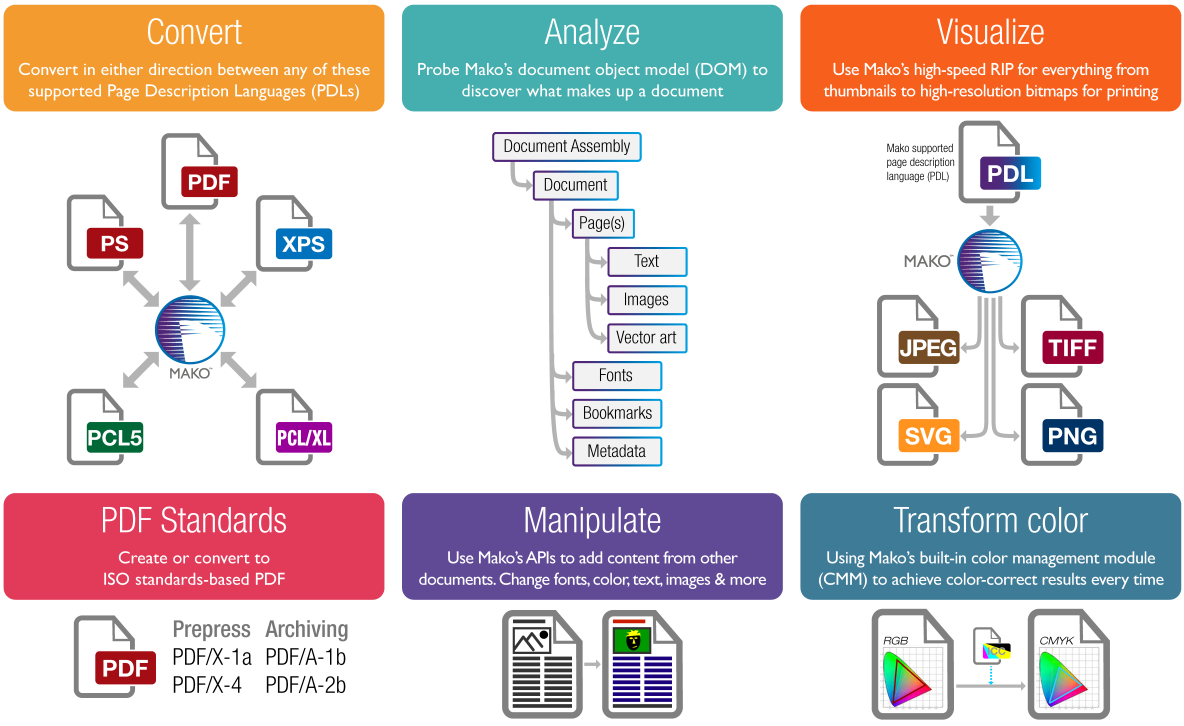


Figure 1 Mako functions

1.1 Introduction

This document describes the core API of the Global Graphics Mako™ SDK. Mako provides the following key capabilities:

A new, simplified Mako API that provides:

- Easy access to the contents of PDF, XPS, PostScript, PCL5 and PCL/XL files, and the on-demand creation of DOM from these formats

- The ability to read XPS in a streaming fashion
- DOM editing for individual pages
- Simple methods for creating, rearranging and merging pages
- Simple methods for creating, rearranging and merging documents
- A growing set of transformations that can be applied to DOM nodes, brushes, images etc. to perform common DOM tasks
- A configurable XPS Output module that can write modified document assemblies to an XPS file or stream
- A PDF output module that can write modified document assemblies to a PDF file or stream
- A PS output module that can write modified document assemblies to a PS file or stream
- A PCL5 output module that can write modified document assemblies to a PCL 5e or 5c file or stream
- A PCL/XL output module that can write modified document assemblies to a PCLXL file or stream
- An on-demand SVG generator that can generate SVG for individual DOM nodes, subtrees, or entire pages
- An on-demand XAML generator that can generate XAML for individual DOM nodes, subtrees, or entire pages
- An XPS-compatible renderer for iOS™ and macOS™ using Quartz 2D
- An XPS-compatible renderer for Android™ using the Skia™ library
- A renderer based on Jaws™ for rendering individual DOM nodes, subtrees, or entire pages to an image or an RGB frame buffer

This release supports bi-directional conversion from any of the supported page description languages (PDLs), ie PDF, XPS, PostScript, PCL5 and PCL/XL.

As well as the core libraries and interfaces, this release includes sample applications demonstrating the use of Mako to create a simple iOS™ XPS viewer application, console applications, DLL wrappers and Microsoft Windows™ printer drivers.

1.2 Contents

The release consists of:

1. A suite of libraries implementing the Mako APIs (`./libs`). For Windows™ these are also included as NuGet packages.
2. The C++ header files describing the published Mako API (`./interface/jawsmako` and `./interface/edl`).
3. A collection of sample applications using Mako including:
    - Mako Converter (`./makoapps/makoconverter`). A simple example showing how to convert from PDF or XPS to XPS using the Mako APIs (Windows and MacOS).
    - Simple Examples (`./makoapps/simpleexamples`). A sample application demonstrating numerous uses of the Mako APIs (Windows and macOS).
    - iOS XPS Viewer (`./makoapps/iOSXPSViewer`) for iOS that provides a simple XPS viewer using the Mako APIs.
    - Android XPS Viewer (`./makoapps/AndroidXPSViewer`) for Android that provides a simple XPS viewer using the Mako APIs.
    - A sample XPSDrv implementation (`./makoapps/xpsdrv`) supporting five streaming output formats (PS, PDF, PCLXL, PCL5e and PCL5c) selectable via individual .inf installation files. This driver encapsulates Mako as an XPSDrv filter with streaming input and streaming output.
4. HTML documentation (this document) describing the public interfaces (`./html`). A PDF version is also included.

2 Topic Index

3 Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

4 Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

5 File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

6 Topic Documentation

6.1 JawsMako API

The JawsMako API.

Topics

- JawsMako Library

  JawsMako encompasses an API that is intended to provide a simpler and more flexible interface than that provided by its predecessor, EDL. It allows for simplified access to supported input types, providing easy APIs for accessing document assemblies, documents and pages, and for performing common tasks. It also provides methods of managing the amount of memory used by JawsMako.
- Error handling

  *EDL errors. Many of the EDL APIs return an (unsigned) integer result code where **EDL_OK** (0) indicates "success" and a non-zero value indicates an error of some kind.*
  *A look-up function is provided to convert these integer result codes into (localizable) strings (see **EDL::getEDLErrorString()**)*
- DOM Objects

  Class descriptions of DOM (Document Object Model) objects.

- Input/Output

  Classes concerned with reading input and writing output.
- Interactive features

  Interactive objects such as annotations and forms.
- Utility

  Utility classes.
- Rendering

  Classes and methods for rendering content.
- Layout

  Classes to perform layout of text and produce fixed content DOM.
- Colors

  Classes and methods for dealing with color spaces and values.
- Validate

  Validator classes. These provide interfaces allowing the validation of documents to published standards.
- Text

  JawsMako Text Conveniences.
- Types

  Common types and required headers for the JawsMako interface.

6.1.1 Detailed Description

The JawsMako API.

JawsMako encompasses an API that is intended to provide a simpler and more flexible interface than that provided by its predecessor, EDL. It allows for simplified access to supported input types, providing easy APIs for accessing document assemblies, documents and pages, and for performing common tasks. It also provides methods of managing the amount of memory used by JawsMako.

Errors are handled through exceptions of type JawsMako::IError which is currently a synonym for EDL::IEDLError.

This API will be extended in future JawsMako releases.

6.1.2 JawsMako Library

JawsMako encompasses an API that is intended to provide a simpler and more flexible interface than that provided by its predecessor, EDL. It allows for simplified access to supported input types, providing easy APIs for accessing document assemblies, documents and pages, and for performing common tasks. It also provides methods of managing the amount of memory used by JawsMako.

Classes

- class IEDLClassFactory

  EDL Factory Interface allows one part of the EDL infrastructure to register class creation methods identified by either GUIDs and /or names (strings) and then another part of the EDL infrastructure to request the creation of instances of one or more of these classes by quoting the same GUID or name.
- class CClassParams

  EDL Object Interface.
- class IEDLObject

  IEDLObject is an abstract base class that is used by all classes that are intended to be created via an EDL class factory.

- class IEDLTempStoreObject

  A mechanism for storing and accessing temporary data for use with Jaws Mako.
- class IEDLTempStore

  A self-cleaning area for storage of temporary data in the form of streams. One per session, obtainable from an ISession. Exceptions of type IEDLError are thrown on failures.
- class ISession

  EDL session class.
- class IRunnable

  Interface to filter's runnable classes.
- class CClassID

  An object to represent a 128-bit globally unique ID.
- class JawsMako::IJawsMako

  An instance of the IJawsMako library. Only one instance of this object is currently allowed. This class may also be used as both an EDL factory and an EDL session, and passed to any EDL API that requires these objects.
- class JawsMako::CTemporaryStoreParameters

  Allows the temporary storage parameters to be optionally overridden.

Functions

- static JAWSMAKO_API IJawsMakoPtr JawsMako::IJawsMako::create (const U8String &tempDir=U8String(""), const U8String &cacheDir=U8String(""), const CTemporaryStoreParameters &tempStoreParams=CTemporaryStoreParameters const std::map< std::string, std::string > &initialParams=std::map< std::string, std::string >())

  Create an IJawsMako instance.
  Only one may be created at any one time.

6.1.2.1 Detailed Description

JawsMako encompasses an API that is intended to provide a simpler and more flexible interface than that provided by its predecessor, EDL. It allows for simplified access to supported input types, providing easy APIs for accessing document assemblies, documents and pages, and for performing common tasks. It also provides methods of managing the amount of memory used by JawsMako.

Errors are handled through exceptions of type JawsMako::IError which is currently a synonym for EDL::IEDLError.

6.1.2.2 Function Documentation

create()

```
static JAWSMAKO_API IJawsMakoPtr JawsMako::IJawsMako::create (
            const U8String & tempDir = U8String(""),
            const U8String & cacheDir = U8String(""),
            const CTemporaryStoreParameters & tempStoreParams = CTemporaryStoreParameters(),
            const std::map< std::string, std::string > & initialParams = std::map< std:↩
:string, std::string >() )  [static]
```

Create an IJawsMako instance.
Only one may be created at any one time.

Parameters

tempDir	An absolute path to a directory JawsMako can use to store temporary files. If a non-empty string is passed, the directory must exist. If an empty string is passed, JawsMako will choose an appropriate temporary location based on the host operating system.
cacheDir	An absolute path to a directory where JawsMako may persistently store cached information. If an empty string is passed, JawsMako will choose an appropriate location based on the host operating system. If a directory is passed it must exist.
tempStoreParams	The desired temporary store parameters. See CTemporaryStoreParams for details.
initialParams	The set of initial parameters for the JawsMako instance. Currently this is used for licensing.

See also

IJawsMakoLicense

Returns

IJawsMakoPtr A smart pointer to the new instance.

6.1.3 Error handling

EDL errors. Many of the EDL APIs return an (unsigned) integer result code where **EDL_OK** (0) indicates "success" and a non-zero value indicates an error of some kind.
A look-up function is provided to convert these integer result codes into (localizable) strings (see **EDL::get↩ EDLErrorString()**)

Enumerations

- enum EDLErrorCode { }

  *These result codes are returned by a Mako API as an unsigned integer result code. **EDL_OK** (0) indicates "success" whereas a non-zero value indicates an error of some kind.*

6.1.3.1 Detailed Description

EDL errors. Many of the EDL APIs return an (unsigned) integer result code where **EDL_OK** (0) indicates "success" and a non-zero value indicates an error of some kind.
A look-up function is provided to convert these integer result codes into (localizable) strings (see **EDL::get↩ EDLErrorString()**)

6.1.3.2 Enumeration Type Documentation

EDLErrorCode

```
enum EDLErrorCode
```

These result codes are returned by a Mako API as an unsigned integer result code. **EDL_OK** (0) indicates "success" whereas a non-zero value indicates an error of some kind.

Enumerator

EDL_ERR_PANIC	Unstable state - abort the application.
EDL_ERR_UNDEFINED	Undefined error.
EDL_ERR_LICENSE	Invalid serial number or licence provided.
EDL_ERR_FILENOTFOUND	File not found.
EDL_ERR_CORRUPT_PNG_IMAGE	Corrupt PNG image encountered.
EDL_ERR_OUTOFMEMORY	Out of memory.
EDL_ERR_READ	Read error.
EDL_ERR_LARGE_FILE	File too large.
EDL_ERR_OPENFORWRITE	Open file for write failed.
EDL_ERR_WRITE	Write failed.
EDL_ERR_IOERROR	General IO error.
EDL_ERR_COMPRESS	Compression error.
EDL_ERR_CONFIG	Configuration error.
EDL_ERR_RESTRICTED_FONT	A restricted font has been encountered.
EDL_ERR_ABORTED	Job aborted by user.
EDL_ERR_BAD_ARGUMENTS	General error for bad arguments passed to an API function.
EDL_ERR_VECTOR_ERROR	An out-of range value was accessed in a CEDLVector.
EDL_ERR_BAD_FONT_MAP	An error was found in a FontMap file.
EDL_ERR_UNICODE	An invalid Unicode string was encountered.
EDL_ERR_PAGE_TOO_LARGE	A page was too large to be used.
EDL_ERR_OPENFORREAD	Open for read failed.
JM_ERR_GENERAL	A general error occurred. Additional information will be provided with the exception.
JM_ERR_UNSUPPORTED	An attempt was made to use an unsupported, unimplemented feature.
JM_ERR_REVERT_FAILED	An attempt was made to revert a page that could not be performed.
JM_ERR_TARGET_NOT_FOUND	A target with the given DOMid was not found.
JM_ERR_RANGE_ERROR	An attempt was made to request or add an item from the API with an out-of-bounds index.
JM_ERR_PAGE_NOT_FOUND	The given page was not found in a document.
JM_ERR_DOCUMENT_NOT_FOUND	The given document was not found in an assembly.
JM_DUPLICATE_FIELD_PARTIAL_NAME	A field with a duplicate partial name was found.
JM_ERR_FORM_FIELD_NOT_FOUND	The given form field was not found.
JM_ERR_DUPLICATE_WIDGET	A widget with a duplicate reference was not found.
JM_ERR_WIDGET_NOT_FOUND	The given widget annotation or annotation reference could not be found.
JM_ERR_BAD_CONFIGURATION	General configuration error.
JM_ERR_APPEARANCE_NOT_FOUND	The required annotation appearance was not found.
JM_ERR_ANNOTATION_NOT_FOUND	The required annotation was not found.
JM_ERR_ASSEMBLY_WRITE_FORBIDDEN	The given assembly cannot be written due to security permissions.
JM_ERR_FONT_NOT_FOUND	The requested font was not found.
JM_ERR_TOO_MANY_PDFOUT_WRITERS	There are too many PDF Out writers currently in progress.
JM_ERR_INFORMATION_NOT_AVAILABLE	The requested information is not yet available.
JM_ERR_DIRECTORY_DOESNT_EXIST	A required directory does not exist on disk.

Enumerator

JM_ERR_RESOURCE_NOT_FOUND	The requested resource could not be found.
JM_ERR_BAD_UNICODE_CMAP	A bad ToUnicode CMap was encountered.
JM_ERR_OPTIONAL_CONTENT_GROUP_NOT_↩ FOUND	The requested optional content group was not found.
JM_ERR_INVALID_OPTIONAL_CONTENT	Invalid optional content information encountered.
JM_ERR_PCLXL	PCL/XL Processing error.
JM_ERR_PCL	PCL5 Processing error.
JM_ERR_PJL	PJL Processing error.
JM_ERR_SYMBOLSET_NOT_FOUND	The requested PCL or PCL/XL symbol set was not found.
JM_ERR_PDF_OBJECT_NOT_FOUND	An expected PDF object was not found.
JM_ERR_INVALID_PDF_OBJECT	An invalid PDF object was encountered.
JM_ERR_INCORRECT_PDF_OBJECT_TYPE	A PDF object of an invalid/unexpected type was encountered.
JM_ERR_IJPDS	IJPDS Processing error.
JM_ERR_ATTEMPTED_WRITE_ON_OPEN_INPUT	An attempt was made to open a writable stream on a file a file referenced by an open input.
JM_ERR_TTF_INSTRUCTIONS	Invalid TrueType instructions were encountered.
EDL_ERR_BAD_DIMENSIONS	Bad dimensions specified for an object.
EDL_ERR_NULL_NODE	An unexpected null DOM node pointer was encountered.
EDL_ERR_BAD_BRUSH	A bad brush was encountered.
EDL_ERR_DOM_CREATION_FAILED	Failure while attempting to create a DOM object.
EDL_ERR_INVALID_DOM_ID	An invalid DOMid was encountered.
EDL_ERR_INVALID_TYPE3_GLYPH	An invalid Type 3 Glyph was encountered.
EDL_ERR_TYPE3_GLYPH_NOT_FOUND	A Type 3 glyph was not found.
EDL_ERR_INVALID_GLYPHS	An invalid Glyphs node was encountered.
EDL_ERR_BOUNDS_CALCULATION_FAILED	An error was encountered while trying to find the bounds of a node.
EDL_ERR_NODE_ERROR	An error was encountered while manipulating a node.
EDL_ERR_BAD_FONT	A bad font was encountered.
EDL_ERR_BAD_GEOMETRY	Bad path geometry was encountered.
EDL_ERR_IMMUTABLE_GEOMETRY	An attempt was made to edit an immutable geometry element.
EDL_ERR_NO_COLOR_NAME	No color name could be determined for the given colorant.
EDL_ERR_INVALID_ZIP	XPS input: Invalid ZIP file encountered.
EDL_ERR_NO_CONTENT_TYPE	XPS input: Unable to retrieve the content type for the part.
EDL_ERR_PROCESS_PART	XPS input: Error processing the part.
EDL_ERR_XPS_PROVIDER	XPS input: XPS provider error.
EDL_ERR_MISSING_RESOURCE	XPS input: Missing resource.
EDL_ERR_DTD_CONTENT	XPS input: DTD content detected.
EDL_ERR_MISSING_REQ_ATTRIBUTE	XPS input: An element is missing a required attribute.
EDL_ERR_INVALID_ATTRIBUTE	XPS input: An invalid attribute has been encountered in the element.
EDL_ERR_PROPERTY_ALREADY_SET	XPS input: Attempt to set the same property twice.
EDL_ERR_REMOTERESOURCE_REF	XPS input: Remote resource dictionary references another remote resource dictionary.

Enumerator

EDL_ERR_KEY_SET_NON_RESOURCE	XPS input: Key attribute is set for an element that is not in the resource dictionary.
EDL_ERR_UNSUPPORTED_GLYPHS	XPS input: Unsupported glyphs encountered.
EDL_ERR_INVALID_URI	XPS input: Invalid URI encountered.
EDL_ERR_DEFLATE	XPS input: Deflate error.
EDL_ERR_NO_KEY_RESDICT_OBJECT	XPS input: Resource dictionary is missing the key attribute.
EDL_ERR_ADD_OBJECT_RESDICT	XPS input: Error adding the element to the resource dictionary.
EDL_ERR_INVALID_ABBR_PATH	XPS input: Error processing abbreviated path.
EDL_ERR_NAMESPACE_NOT_UNDERSTOOD	XPS input: Namespace is not understood.
EDL_ERR_XML_PARSER	XPS input: Parser error encountered.
EDL_ERR_NO_REQ_ELEMENT	XPS input: Element is missing a required subelement.
EDL_ERR_MORE_THAN_ONE_ELEMENT	XPS input: Element has more than one instance of a subelement.
EDL_ERR_INVALID_ELEMENT	XPS input: Element contains an invalid subelement.
EDL_ERR_FALLBACK_BEFORE_CHOICE	XPS input: Fallback element has been found before choice.
EDL_ERR_PREFIX_NOT_DEFINED	XPS input: Undefined prefix encountered.
EDL_ERR_UNPREFIXED_ATTRIBUTE	XPS input: Unprefixed attribute encountered.
EDL_ERR_BROKEN_ELEMENT_SEQUENCE	XPS input: Broken element sequence encountered.
EDL_ERR_PREFIXED_ATTRIBUTE	XPS input: Prefixed attribute encountered.
EDL_ERR_INVALID_IMAGE	XPS input: Invalid image encountered.
EDL_ERR_NUMBER_REQ_ELEMENT	XPS input: Insufficient elements of a particular type encountered.
EDL_ERR_MISSING_XPS_PART	XPS input: Missing XPS part.
EDL_ERR_PRINTTICKET_EXISTS	XPS input: PrintTicket already exists.
EDL_ERR_MULTIPLE_STARTPARTS_RELS	XPS input: Multiple StartPart relationships encountered.
EDL_ERR_DICT_ITEM_PRESENT	XPS input: Multiple items with the same key present.
EDL_ERR_MULTIPLE_COREPROPERTIES_RELS	XPS input: Multiple CoreProperty relationships encountered.
EDL_ERR_MULTIPLE_DOCSTRUCTURE_RELS	XPS input: Multiple DocumentStructure relationships encountered.
EDL_ERR_MULTIPLE_THUMBNAIL_RELS	XPS input: Multiple thumbnail relationships encountered.
EDL_ERR_INVALID_THUMBNAIL_TYPE	XPS input: Invalid thumbnail type encountered.
EDL_ERR_PRESERVED_ITEMS_NOT_↩ DECLARED_IGNORABLE	XPS input: Unable to preserve items! The namespace is not declared as ignorable.
EDL_ERR_MORE_THAN_ONE_GROUP_ELEMENT	XPS input: Element contains more than one subelement.
EDL_ERR_NO_REQ_GROUP_ELEMENT	XPS input: Element requires one subelement.
EDL_ERR_INVALID_COLOR_SPECIFICATION	XPS input: Invalid color specification encountered.
EDL_ERR_PROCESS_CONTENT_NOT_↩ DECLARED_IGNORABLE	XPS input: Namespace in ProcessContent attribute is not declared as ignorable.
EDL_ERR_DUPLICATE_URI	XPS input: Duplicate URI encountered.
EDL_ERR_UNEXPECTED_RESOURCE	XPS output: Unexpected resource encountered.
EDL_ERR_DEVICE_OUTOFMEMORY	XPS output: Insufficient device memory to process page.

Enumerator

EDL_ERR_XPS_FROM_ENCRYPTED_SOURCE	XPS output: Attempt to create XPS from an encrypted document.
EDL_ERR_DELETE_OUTLINE_WITH_CLOSED_↩ PARENT	Outline: Attempt to delete outline entry with closed parent.
EDL_ERR_APPEND_OUTLINE_TO_CLOSED_↩ NODE	Outline: Attempt to append outline entry to a closed node.
EDL_ERR_FAILS_TO_ADD_OUTLINE_TO_↩ METADATA_STORAGE	Outline: Failed to append an outline entry to the metadata storage node.
EDL_ERR_INTERNAL_RIP	Internal RIP error.
EDL_ERR_OPEN_PDF	PDF input: Error opening PDF.
EDL_ERR_INVALID_PDF_PASSWORD	PDF input: Error opening encrypted PDF; Invalid password and/or incorrect private key.
EDL_ERR_PSOUT_GENERAL_FAILURE	PS output: General error.
EDL_ERR_PDFOUT_GENERAL_FAILURE	PDF output: General error.
EDL_ERR_INVALID_FONT	PDF output: Error processing a font.
EDL_ERR_INCOMPATIBLE_PDFA	PDF/A output: Incompatible content found.
EDL_ERR_INCOMPATIBLE_PDFX	PDF/X output: Incompatible content found.
EDL_ERR_BAD_COLOR_SPACE	Color: A bad color space was encountered.
EDL_ERR_COLOR_CONVERSION_FAILURE	Color: Color conversion failed.
EDL_ERR_INCOMPATIBLE_IMAGE_PARAMETER	Image: An image encoder has been given an image or parameter that it cannot handle.
EDL_ERR_IMAGE_PARAMETER_OUT_OF_RANGE	Image: An image parameter for encoding or decoding is outside the expected range.
EDL_ERR_IMAGE_DECODE_FAILURE	Image: An image could not be decoded.
EDL_ERR_IMAGE_ENCODE_FAILURE	Image: An image could not be encoded.
EDL_ERR_ZIP_WRITE_ONLY	Zip: An attempt was made to retrieve a stream from a write-only Zip file.
EDL_ERR_ZIP_PATH_NOT_FOUND	Zip: An expected entry in a Zip package was not found.
EDL_ERR_LICENSE_MISSING	License: License checking indicated that required license information is missing.
EDL_ERR_LICENSE_BAD_CONFIG	License:
EDL_ERR_LICENSE_INVALID	License: License checking indicated that this license is invalid.

6.1.4 DOM Objects

Class descriptions of DOM (Document Object Model) objects.

Topics

- Brushes

  Interfaces to DOM brush objects.
- Fonts

  Font handling.
- Form XObjects

The children of this node type comprise the contents of a PDF/PS style form. This includes the /Matrix and /BBox (bounds) entries that are normally present in form dictionaries. Here, bounds (if non-empty) is used in preference to calculating the bounds of any children. This node should not be present in the DOM tree as a general node. It must only be used as the contents of an IDOMFormInstance.

- Nodes

  IDOMNode Interface. An interface to basic DOM node functionality. IDOMNode is the base class for many of the other DOM node types, and defines many of the basic functions of DOM nodes.
- Grouping

  Classes to create and manage groups of nodes for different purposes.
- Glyphs

  Classes modelling individual characters from a font, or runs of same.
- Geometry

  DOM interfaces for path geometry, figures and segments.
- Images

  Classes to manage images.
- Pages

  Classes to create and manage Page and Fixed pages.
- PDF-specific

  PDF-specific object classes, such Optional Content Groups.
- Outlines

  Outlines represent indexes to specific locations in the document or an external location.
- Metadata

  Document metadata, job and print tickets.
- Object Hashing

  Hashes for object identification and comparison.
- Other DOM Objects

  Other DOM objects.

6.1.4.1 Detailed Description

Class descriptions of DOM (Document Object Model) objects.

6.1.4.2 Brushes

Interfaces to DOM brush objects.

Classes

- class IDOMBrush

  Interface to the brush element.
- class IDOMTransformableBrush

  Abstract interface for a brush to which a render transform may be applied.
- class IDOMGradientStop

  IDOMGradientStop defines the ramp of colors to use on a gradient.
- class IDOMSolidColorBrush

  A solid color brush is used to fill defined geometric regions with a solid color. If there is an alpha component of the color, it is combined in a multiplicative way with the corresponding opacity attribute.
- class IDOMGradientBrush

A common interface for both IDOMLinearGradient and IDOMRadialGradient. Provides straightforward access to common attributes.

- class IDOMLinearGradientBrush

  IDOMLinearGradientBrush interface. A linear gradient brush is used to specify a gradient along a vector.

- class IDOMRadialGradientBrush

  IDOMRadialGradientBrush interface. A radial gradient brush defines an ellipse to be filled with the gradient. The ellipse is defined by its center, x radius, and y radius. Independently, a gradient origin is specified for the brush. The gradient origin defines the center of the gradient; a gradient stop with an offset at 0.0 defines the color at the gradient origin. The outer bound of the ellipse defines the end "point" of the gradient; that is, a gradient stop with an offset at 1.0 defines the color at the circumference of the ellipse, and all other gradient stops define their offsets relative to the radial distance between the gradient origin and the circumference.

- class IDOMImageBrush

  Provides an interface to a DOM image brush object.

- class IDOMMaskedBrush

  IDOMMaskedBrush interface, this describes a generalization of a masked image. The sub-brush (set by getBrush()/setBrush()) is painted through a mask specified by the image. Importantly, the sub-brush is not subject to the IDOMImageBrush render transform. Tiling is not supported for this brush type.

- class IDOMVisualBrush

  A visual brush is used to fill a region with a vector drawing.

- class IDOMVisualRoot

  IDOMVisualRoot interface.

- class IDOMSoftMaskBrush

  IDOMSoftMaskBrush provides a way of representing a PDF style soft mask in it's entirity. The soft mask brush contains a suitable IDOMTransparency group, as well as the necessary soft mask details. See section 7.5.4 of the PDF 1.7 specification. These are only allowed for OpacityMask entries.

- class IDOMTilingPatternBrush

  IDOMTilingPatternBrush provides a way of representing a PS style tiling pattern.

- class IDOMShadingPatternBrush

  IDOMShadingBrush provides a way of representing a PS style shading pattern.

- class IDOMShadingPatternType1Brush

  IDOMShadingBrush provides a way of representing a PS style type 1 shading pattern.

- class IDOMShadingPatternType2Brush

  IDOMShadingBrush provides a way of representing a PS style type 2 shading pattern.

- class IDOMShadingPatternType3Brush

  IDOMShadingPatternType3Brush provides a way of representing a PS style type 2 shading pattern.

- class IDOMShadingPatternType4567Brush

  IDOMShadingPatternType4567Brush provides a way of representing a PS style type 4 shading pattern.

- class IDOMNullBrush

  IDOMNullBrush provides a way of representing the default marking brush in a Type3 postscript glyph definition or a tiling pattern with paintType 2. This is more of a placeholder that gets replaced when the Type3 glyph or paintType 2 tiling pattern is actually invoked.

Enumerations

- enum IDOMBrush::eBrushType { IDOMBrush::eSolidColor , IDOMBrush::eLinearGradient , IDOMBrush::eRadialGradient , IDOMBrush::eImage , IDOMBrush::eMasked , IDOMBrush::eVisual , IDOMBrush::eSoftMask , IDOMBrush::eTilingPattern , IDOMBrush::eType1ShadingPattern , IDOMBrush::eType2ShadingPattern , IDOMBrush::eType3ShadingPattern , IDOMBrush::eType4567ShadingPattern , IDOMBrush::eNull }

  Brush type enumeration.

- enum eColorInterpolationMode { eSRgbLinearInterpolation , eSCRgbLinearInterpolation }

Color interpolation mode type enumeration.

- enum eSpreadMethod { ePad , eReflect , eRepeat , eNoSpread }

  Spread Method type enumeration.

- enum eViewUnits { eAbsolute }

  View units type enumeration.

- enum eTilingMode { eNoTile , eTile , eFlipX , eFlipY , eFlipXY }

  Tiling mode type enumeration.

6.1.4.2.1 Detailed Description

Interfaces to DOM brush objects.

6.1.4.2.2 Enumeration Type Documentation

eBrushType

`enum IDOMBrush::eBrushType`

Brush type enumeration.

Enumerator

eSolidColor	Solid color brush.
eLinearGradient	Linear gradient brush.
eRadialGradient	Radial gradient brush.
eImage	Image brush.
eMasked	Masked brush.
eVisual	Visual brush.
eSoftMask	Softmask brush.
eTilingPattern	Tiling pattern brush.
eType1ShadingPattern	Type 1 shading pattern brush.
eType2ShadingPattern	Type 2 shading pattern brush.
eType3ShadingPattern	Type 3 shading pattern brush.
eType4567ShadingPattern	Type 4, 5, 6 or 7 shading pattern brush.
eNull	Null brush.

eColorInterpolationMode

`enum eColorInterpolationMode`

Color interpolation mode type enumeration.

Enumerator

eSRgbLinearInterpolation	Linear interpolation for sRGB color space.
eSCRgbLinearInterpolation	Linear interpolation for scRGB color space.

eSpreadMethod

`enum eSpreadMethod`

Spread Method type enumeration.

Enumerator

ePad	Fill the remaining area with the color specified by the final gradient stop.
eReflect	Fill the remaining area by reflecting the gradient, such that the finish point becomes the start of the reflected gradient and the gradient is calculated by stepping back down through the original gradient points.
eRepeat	Fill the remaining area by repeating the gradient from its first gradient stop, starting at the point defined by the last gradient stop.
eNoSpread	Do not fill the remaining area, but allow it to remain transparent.

eTilingMode

`enum eTilingMode`

Tiling mode type enumeration.

Enumerator

eNoTile	No tiling. If the area to be painted is larger than the image, just paint the image once (in the location specified by the brush's viewport), and leave the remaining area transparent.
eTile	Tile image without any flipping or rotating of the image. A square image consisting of a single diagonal line between opposite corners would produce diagonal lines when tiled in this mode.
eFlipX	Tile image such that alternate columns of tiles are flipped horizontally. A square image consisting of a single diagonal line between opposite corners would produce chevrons running horizontally across the area when tiled in this mode.
eFlipY	Tile image such that alternate rows of tiles are flipped vertically. A square image consisting of a single diagonal line between opposite corners would produce chevrons running vertically across the area when tiled in this mode.
eFlipXY	Tile image such that alternate columns of tiles are flipped horizontally AND alternate rows of tiles are flipped vertically. A square image consisting of a single diagonal line between opposite corners would produce a grid of squares balanced on their points when tiled in this mode.

eViewUnits

`enum eViewUnits`

View units type enumeration.

Enumerator

eAbsolute	Absolute units.

6.1.4.3 Fonts

Font handling.

Classes

- class IDOMFontSource

  The font source for the class IDOMFont. This class describes the different ways fonts are constructed.
- class IDOMFontSourceStreamFilter

  An abstract interface for fonts sourced from a font stream filter.
- class IDOMFontSourceFromStream

  The source for IDOMFont when sourced from an existing stream.
- class IDOMFontSourceObfuscationConverter

  Interface for a font sourced from a converter that performs obfuscation and deobfuscation.
- class IDOMFont

  IDOMFont Base Class.
- class IDOMFontOpenType

  IDOMFontOpenType interface.
- class IDOMFontOpenType::CCIDMap

  For TrueType-based CIDFonts, the mapping from CIDs to GlyphIds for non-identity orderings.
- class SignatureID

  Opentype table signatures.
- class IDOMFontOTFTrueType

  Opentype Font.
- class IDOMType3Font

  Representation of a PostScript/PDF Type 3 Font. At present, the stream cannot be set, only retrieved.
- class IFontHeaderWriteSegmentBlockEnumerator

  IFontHeaderWriteSegmentBlockEnumerator Enumerates over the PCLXL Font Header block items for the XL Read↩ FontHeader operator.
- class IFontPCL5WriteSegmentBlockEnumerator

  IFontPCL5WriteSegmentBlockEnumerator Enumerates over the PCL5 font blocks.
- class IDOMFontPCLXL

  This class models PCL XL TrueType and bitmap fonts derived from an OpenType font source.
- class IDOMFontPCL5

  IDOMFontPCL5 (PCL5 Truetype) derived from an OpenType font source.

Typedefs

- typedef uint32 **IDOMFont::CharCode**

  type used to uniquely identify a character code
- typedef std::map< CharCode, IDOMGlyph::GlyphID > **IDOMFont::CCharacterMap**

  Map type used for storing character code to glyph ID mappings.
- typedef uint16 **IDOMFontPCLXL::SymbolSet**

  Type represents the PCL XL Font Symbol Set.
- typedef uint16 **IDOMFontPCL5::SymbolSetIDCode**

  Type represents the PCL5 Font Symbol Set.

Enumerations

- enum IDOMFontSource::eFontSourceType { IDOMFontSource::eFontSourceTypeNone , IDOMFontSource::eFontSourceTypeSt , IDOMFontSource::eFontSourceTypeStream , IDOMFontSource::eFontSourceTypeFont }

  type used to uniquely identify the source type of a font
- enum IDOMFontSourceStreamFilter::eFontStreamFilterType { IDOMFontSourceStreamFilter::eFontStreamFilterTypeNone , IDOMFontSourceStreamFilter::eFontStreamFilterTypeObfuscation }

  An enumeration type used to identify the source type of a font.
- enum IDOMFontSourceObfuscationConverter::eOperation { IDOMFontSourceObfuscationConverter::eObfuscate , IDOMFontSourceObfuscationConverter::eDeobfuscate }

  type used to uniquely identify the conversion direction
- enum IDOMFontPCL5::ePCL5FontType

  Type used to uniquely identify the type of PCL5 font (Currently, only TTF supported).
- enum IDOMFont::eFontType { IDOMFont::eFontTypeUnknown , IDOMFont::eFontTypeOpenType , IDOMFont::eFontType3 , IDOMFont::eFontTypePCLXL , IDOMFont::eFontTypePCL5 }

  type used to uniquely identify the type of font
- enum IDOMFontOpenType::eOpenTypeFontType { IDOMFontOpenType::eOpenTypeFontTypeUnknown , IDOMFontOpenType::eOpenTypeFontTypeTTF , IDOMFontOpenType::eOpenTypeFontTypeCFF , IDOMFontOpenType::eOpenTypeFontTypeTTC }

  Type used to uniquely identify the type of OpenType font.
- enum IDOMFontOpenType::eOriginalFontType { IDOMFontOpenType::eOriginalTypeOpenType , IDOMFontOpenType::eOriginalType1 , IDOMFontOpenType::eOriginalType2 , IDOMFontOpenType::eOriginalType42 , IDOMFontOpenType::eOriginalType9 , IDOMFontOpenType::eOriginalType11 , IDOMFontOpenType::eOriginalPclTrueType }

  Type used to uniquely identify the original type of Font. In the PostScript/PDF inputs, most font types are converted to OpenType before insertion into the DOM. This allows the ability to discover what the original type was.
- enum IDOMFontOpenType::Signature::eSignature { }

  Type used to uniquely identify the known open type signatures.
- enum IFontHeaderWriteSegmentBlockEnumerator::eFontHeaderWriteSegmentItem

  The enumeration of the segment types to be enumerated over.
- enum IDOMFontPCLXL::ePCLXLFontType { IDOMFontPCLXL::ePCLXLFontTypeTTF , IDOMFontPCLXL::ePCLXLFontTypeBitm }

  The enumeration used to identify the type of the PCL XL font.

6.1.4.3.1 Detailed Description

Font handling.

6.1.4.3.2 Enumeration Type Documentation

eFontSourceType

```
enum IDOMFontSource::eFontSourceType
```

type used to uniquely identify the source type of a font

Enumerator

eFontSourceTypeNone	Not specified.
eFontSourceTypeStreamFilter	Font filter (converter)
eFontSourceTypeStream	Font data stream.
eFontSourceTypeFont	Derived from a font.

eFontStreamFilterType

`enum IDOMFontSourceStreamFilter::eFontStreamFilterType`

An enumeration type used to identify the source type of a font.

Enumerator

eFontStreamFilterTypeNone	Not specified.
eFontStreamFilterTypeObfuscation	Obfuscation filter.

eFontType

`enum IDOMFont::eFontType`

type used to uniquely identify the type of font

Enumerator

eFontTypeUnknown	Unknown.
eFontTypeOpenType	OpenType font.
eFontType3	Type3 font.
eFontTypePCLXL	PCL/XL font.
eFontTypePCL5	PCL5 font.

eOpenTypeFontType

`enum IDOMFontOpenType::eOpenTypeFontType`

Type used to uniquely identify the type of OpenType font.

Enumerator

eOpenTypeFontTypeUnknown	Unknown font type.
eOpenTypeFontTypeTTF	TrueType font.
eOpenTypeFontTypeCFF	Compact font format.
eOpenTypeFontTypeTTC	TrueType collections.

eOperation

enum IDOMFontSourceObfuscationConverter::eOperation

type used to uniquely identify the conversion direction

Enumerator

eObfuscate	Obfuscation.
eDeobfuscate	Deobfuscation.

eOriginalFontType

enum IDOMFontOpenType::eOriginalFontType

Type used to uniquely identify the original type of Font. In the PostScript/PDF inputs, most font types are converted to OpenType before insertion into the DOM. This allows the ability to discover what the original type was.

Enumerator

eOriginalTypeOpenType	Originally OpenType format.
eOriginalType1	Type 1 font.
eOriginalType2	Type 2 font - Usually a Type 1 encoded in a PDF in CFF form.
eOriginalType42	Type42/Truetype from PS/PDF.
eOriginalType9	CIDFontType0 (CIDFont based on Type 1 CharStrings)
eOriginalType11	CIDFontType2 (CIDFont based on TrueType glyph descriptions)
eOriginalPclTrueType	A TrueType font originally embedded in a PCL/5 or PCL/XL document.

ePCLXLFontType

enum IDOMFontPCLXL::ePCLXLFontType

The enumeration used to identify the type of the PCL XL font.

Enumerator

ePCLXLFontTypeTTF	TrueType based font.
ePCLXLFontTypeBitmap	Bitmap font.

6.1.4.4 Form XObjects

The children of this node type comprise the contents of a PDF/PS style form. This includes the /Matrix and /BBox (bounds) entries that are normally present in form dictionaries. Here, bounds (if non-empty) is used in preference to calculating the bounds of any children. This node should not be present in the DOM tree as a general node. It must only be used as the contents of an IDOMFormInstance.

Classes

- class IDOMForm

  Interface to DOM node representing PDF-style Form XObjects.
- class IDOMFormInstance

  IDOMFormInstance interface. This describes an instance of an IDOMForm in a DOM tree.

6.1.4.4.1 Detailed Description

The children of this node type comprise the contents of a PDF/PS style form. This includes the /Matrix and /BBox (bounds) entries that are normally present in form dictionaries. Here, bounds (if non-empty) is used in preference to calculating the bounds of any children. This node should not be present in the DOM tree as a general node. It must only be used as the contents of an IDOMFormInstance.

6.1.4.5 Nodes

IDOMNode Interface. An interface to basic DOM node functionality. IDOMNode is the base class for many of the other DOM node types, and defines many of the basic functions of DOM nodes.

Files

- file idomresources.h

  IDOMResources interface.

Classes

- class IDOMCatalog

  IDOMCatalog interface The IDOMCatalog serves as a catalog for addressable DOM nodes, where a DOM node ID is used as the address of the node.
- class IDOMNodeFlags

  *A collection of bit flags used to signal various conditions of the node. For example, the **eNodeRenderFlag** flag identifies nodes that require rendering.*
- class IDOMNode

  Abstract class providing the interface to basic DOM node functionality. IDOMNode is the base class for many of the other DOM node types, and defines many of the basic functions of DOM nodes. Exceptions of type IEDLError are thrown on outright failures.
- class IDOMResource

  Provides an interface to an EDL DOM node representing a generalised resource. A resource represents non-markup document content such as images, fonts and profiles. Resources are generally stream based. This class provides the base class for interfaces to more specialized resource node types.

Enumerations

- enum eEdgeMode { eEMDefault , eEMAliased }

  *Available options for the **RenderOptionsEdgeMode** property, which controls anti-aliasing behavior on anti-aliasing renderers.*
- enum eDOMNodeType {
  eDOMContentRootNode , eDOMDocumentSequenceNode , eDOMDocumentNode , eDOMFixedDocumentNode ,
  eDOMPageNode , eDOMFixedPageNode , eDOMGroupNode , eDOMCharPathGroupNode ,
  eDOMTransparencyGroupNode , eDOMCanvasNode , eDOMGlyphsNode , eDOMGlyphNode ,
  eDOMPathNode , eDOMRefNode , eDOMJobTkContentNode , eDOMJobTkNodeNode ,
  eDOMJobTkValueNode , eDOMJobTkGenericNodeNode , eDOMJobTkGenericCharacterDataNode ,
  eDOMVisualRootNode ,
  eDOMOutlineNode , eDOMOutlineEntryNode , eDOMAnnotationAppearanceNode , eDOMFormNode ,
  eDOMFormInstanceNode , eDOMFragmentNode , eDOMTileNode , eDOMNodeTypeCnt }

  DOM node type enumeration.

6.1.4.5.1 Detailed Description

IDOMNode Interface. An interface to basic DOM node functionality. IDOMNode is the base class for many of the other DOM node types, and defines many of the basic functions of DOM nodes.

6.1.4.5.2 Enumeration Type Documentation

eDOMNodeType

`enum eDOMNodeType`

DOM node type enumeration.

Enumerator

eDOMContentRootNode	Content Root Node.
eDOMDocumentSequenceNode	Document Sequence Node.
eDOMDocumentNode	Document Node.
eDOMFixedDocumentNode	Fixed Document Node.
eDOMPageNode	Page Node.
eDOMFixedPageNode	FixedPage Node.
eDOMGroupNode	Group Node.
eDOMCharPathGroupNode	Char Path Group Node.
eDOMTransparencyGroupNode	Transparency GroupNode.
eDOMCanvasNode	Canvas Node.
eDOMGlyphsNode	Glyphs Node.
eDOMGlyphNode	Glyph Node.
eDOMPathNode	Path Node.
eDOMRefNode	Ref Node.
eDOMJobTkContentNode	JobTicket Content Node.
eDOMJobTkNodeNode	JobTicket Node Node.
eDOMJobTkValueNode	JobTicket Value Node.
eDOMJobTkGenericNodeNode	JobTicket Generic Node Node.
eDOMJobTkGenericCharacterDataNode	JobTicket Generic Character Data Node.
eDOMVisualRootNode	Visual Root Node.
eDOMOutlineNode	Outline Node.
eDOMOutlineEntryNode	Outline Entry Node.
eDOMAnnotationAppearanceNode	Annotation Appearance Node.
eDOMFormNode	Form Node.
eDOMFormInstanceNode	Form Instance Node.
eDOMFragmentNode	Fragment Node.
eDOMTileNode	Tile Node.
eDOMNodeTypeCnt	Node Type Container.

6.1.4.6 Grouping

Classes to create and manage groups of nodes for different purposes.

Classes

- class IDOMCanvas

  A canvas is a special form of an isolated, non-knockout, normal blended transparency group.
- class IDOMGroup

  Interface to DOM node representing Group Elements.
- class IDOMTransparencyGroup

  IDOMTransparencyGroup interface. Analogous to PDF Transparency groups.

6.1.4.6.1 Detailed Description

Classes to create and manage groups of nodes for different purposes.

6.1.4.7 Glyphs

Classes modelling individual characters from a font, or runs of same.

Classes

- class IDOMCharPathGroup

  IDOMCharPathGroup interface Interface to DOM node representing Group Elements that consist of stroked text.
- class IDOMGlyphName
- class IDOMGlyph

  The IDOMGlyph class is an abstract class modelling a single character from a font.
- class IDOMGlyphs

  An abstract class providing an interface to a "Glyphs" node. Glyphs nodes are used to represent a run of uniformly formatted text from a single font. Text runs are broken by line advances and formatting changes. When a text run is broken, a new Glyphs node will be created to describe the text from the change point onwards.
- class CIndicesGlyph

  Utility code allowing the simpler manipulation of glyph glusters represented by the UnicodeString and Indices entries of an IDOMGlyphs object.
- class CGlyphsCluster

  A single cluster generated from the parallel Indices and Unicode strings present in an IDOMGlyphs node. This represents the smallest logical unit of text. Here, a single cluster may represent:
- class CGlyphsClusters

  A single cluster generated from the parallel Indices and Unicode strings present in an IDOMGlyphs node. This represents the smallest logical unit of text. Here, a single cluster may represent:

Typedefs

- typedef uint16 IDOMGlyph::GlyphID

  Holds the glyph ID of the glyph.

Enumerations

- enum IDOMGlyphs::eStyleSimulations

  Specifies a style simulation for a glyphs node.
- enum IDOMGlyph::eGlyphIDSpecial { IDOMGlyph::eGlyphIDNotdef = 0 }

  Enumeration type used to define known, special Glyph IDs.

6.1.4.7.1 Detailed Description

Classes modelling individual characters from a font, or runs of same.

6.1.4.7.2 Typedef Documentation

GlyphID

`IDOMGlyph::GlyphID`

Holds the glyph ID of the glyph.

A glyf table contains the data describing each glyph in the font. A particular glyph is identified by a glyph id, which is used exclusively throughout the font to identify that glyph. For further information on using glyf tables see http://www.microsoft.com/typography/otspec/glyf.htm.

Note: There are some special glyph ids which are reserved by convention as described in the eGlyphIDSpecial type. It is advisable to follow these conventions when modifying fonts. Glyph 0 is normally an open rectangle and is used by rendering systems to substitute for characters that are not present in the font.

6.1.4.7.3 Enumeration Type Documentation

eGlyphIDSpecial

`enum IDOMGlyph::eGlyphIDSpecial`

Enumeration type used to define known, special Glyph IDs.

Enumerator

eGlyphIDNotdef	The glyph ID to subsitute for font undefined glyphs.

6.1.4.8 Geometry

DOM interfaces for path geometry, figures and segments.

Classes

- class PointTmpl< PointType >

  Geometry primitives including: point, rectangle and matrix types supporting both integer and floating point values within.
- class BoxTmpl< PointType >

  Template for a PDF-style box. Similar to a rectangle but specified using a left, bottom, right and top coordinate.
- class IDOMPathNode

  Interface to an EDL path node. A path node specifies a geometry that can be filled with a brush.
- class IDOMPathSegment

  Interface to path segment element. The path segment is the smallest unit in a path geometry.

- class IDOMArcSegment

  Interface to Arc Segment element.
- class IDOMPolyLineSegment

  Interface to a polyline segment node. A polyline segment describes a polygonal drawing containing an arbitrary number of individual vertices. The Points attribute defines the vertices.
- class IDOMPolyBezierSegment

  Interface to a path segment node describing a set of cubic Bézier curves.
- class IDOMPolyQuadraticBezierSegment

  Interface to a polyqualdratic Bézier segment. A polyqualdratic Bézier segment describes a set of quadratic Bézier curves from the starting point defined in the IDOMPathFigure, or from the end point of the previous segment, through a set of vertices, using specified control points. The Points attribute stores an off-curve control point (x2n-1, y2n-1) followed by the end point (x2n, y2n) for each quadratic Bézier curve (where n represents the quadratic Bézier curve).
- class IDOMPathFigure

  Interface to the path figure element. A path figure is a single shape comprised of continuous path segments. One or more path figures collectively define an entire path geometry. A path geometry may define the fill algorithm to be used on the component path figures. Instances of this type use exceptions of IEDLError for error handling.
- class IDOMPathGeometry

  Interface to a path geometry node.
- class IDOMPathGeometryBuilder

  Interface to a path geometry builder.
- class IDOMShape

  Interface to an IDOMShape.
- class IDOMShape::CShapeDetails

  Provides a view into the regions that comprise the shape. A region consists of a series of spans, each representing a series of rectangles that share the same y span.
- struct IDOMShape::CShapeDetails::CSpan

  Representation of a series of rectangles sharing the same y span.

Enumerations

- enum CTransformMatrix< TItem >::eOperationTypes { CTransformMatrix< TItem >::eDoesTranslate = 0x1 , CTransformMatrix< TItem >::eDoesScale = 0x2 , CTransformMatrix< TItem >::eDoesRotate = 0x4 , CTransformMatrix< TItem >::eIsComplex = 0x8 }

  Classification of operation type flags of the transform.
- enum IDOMPathNode::eStrokeLineJoin { IDOMPathNode::eMiterJoin , IDOMPathNode::eBevelJoin , IDOMPathNode::eRoundJoin }

  Specifies the different ways in which the lines in the path could be joined.
- enum IDOMPathNode::eStrokeMiterLimitTreatment { IDOMPathNode::eClipLongMiters , IDOMPathNode::eBevelLongMiters }

  Chooses how miters that extend beyond the miter limit should be treated. ClipLongMiters specifies XPS style behaviour, where miters extending beyond the limit are clipped to the limit. BevelLongMiters specifies PDF/PS style behaviour where miters longer than the limit are instead replaced with a bevel join.
- enum IDOMPathNode::eStrokeLineCap { IDOMPathNode::eFlatCap , IDOMPathNode::eSquareCap , IDOMPathNode::eRoundCap , IDOMPathNode::eTriangleCap }

  Specifies the different types of line end caps available.
- enum IDOMArcSegment::eSweepDirection

  The direction of the sweep-clockwise or counter clockwise-between the start and end points.
- enum IDOMPathGeometry::eFillRule { IDOMPathGeometry::eFREvenOdd , IDOMPathGeometry::eFRNonZero }

  Specifies the algorithm to determine whether or not a point is inside a shape on the canvas.

6.1.4.8.1 Detailed Description

DOM interfaces for path geometry, figures and segments.

6.1.4.8.2 Enumeration Type Documentation

eFillRule

```
enum IDOMPathGeometry::eFillRule
```

Specifies the algorithm to determine whether or not a point is inside a shape on the canvas.

```
"Insideness" is determined by drawing a ray from the point to infinity in any direction and counting
the number of segments from the given shape that the ray crosses.

The EvenOdd algorithm determines the "insideness" of a point on the canvas by drawing a ray
from the point to infinity in any direction and counting the number of segments from the given
shape that the ray crosses. If this number is odd, the point is inside; if it is even, the point is outs
This is the default rule.

The NonZero rule determines the "insideness" of a point on the canvas by drawing a ray from the
point to infinity in any direction and then examining the places where a segment of the shape
crosses the ray. Starting with a count of zero, add one each time a segment crosses the ray from
left to right and subtract one each time a path segment crosses the ray from right to left. After
counting the crossings, if the result is zero then the point is outside the path; otherwise, it is insid
```

Enumerator

eFREvenOdd	Use the even/odd fill algorithm.
eFRNonZero	Use the non-zero fill algorithm.

eOperationTypes

```
template<typename TItem >
enum CTransformMatrix::eOperationTypes
```

Classification of operation type flags of the transform.

Enumerator

eDoesTranslate	Does translate
eDoesScale	Does scale
eDoesRotate	Does rotate
eIsComplex	Is complex

eStrokeLineCap

enum IDOMPathNode::eStrokeLineCap

Specifies the different types of line end caps available.

Enumerator

eFlatCap	Flat end caps.
eSquareCap	Square end caps.
eRoundCap	Round end caps.
eTriangleCap	Triangular end caps.

eStrokeLineJoin

enum IDOMPathNode::eStrokeLineJoin

Specifies the different ways in which the lines in the path could be joined.

Enumerator

eMiterJoin	Use mitered joins.
eBevelJoin	Use bevelled joins.
eRoundJoin	Use rounded joins.

eStrokeMiterLimitTreatment

enum IDOMPathNode::eStrokeMiterLimitTreatment

Chooses how miters that extend beyond the miter limit should be treated. ClipLongMiters specifies XPS style behaviour, where miters extending beyond the limit are clipped to the limit. BevelLongMiters specifies PDF/PS style behaviour where miters longer than the limit are instead replaced with a bevel join.

Enumerator

eClipLongMiters	Clip long miters.
eBevelLongMiters	Bevel long miters.

eSweepDirection

enum IDOMArcSegment::eSweepDirection

The direction of the sweep-clockwise or counter clockwise-between the start and end points.

```
The intersection of two ellipses produces four line segments.  Generally speaking, two segments will
have a sweep of more than 180 degrees, and two segments will have a sweep of less than 180 degrees.
Of each pair of segments, one will be drawn in a clockwise direction from the starting point
the other in a counter clockwise direction.  SweepDirection is used in conjunction with the
Boolean member isLargeArc to specify which of the four arcs is required (see getIsLargeArc()).
```

6.1.4.9 Images

Classes to manage images.

Files

- file iimagecodec.h

Classes

- class IDOMImageProperties

  The IDOMImageProperties interface provides access to an underlying implementation which stores miscellaneous information about the associated image.
- class IDOMImage

  The base class describing an image. This class is subclassed to create a number of more specific image types. Instances of these objects may throw IEDLError exceptions on failure.
- class IDOMJPEGImage

  Interface to a class representing a JPEG (.jpg or .jpeg) image.
- class IDOMPNGImage

  Interface to a class representing a PNG (.png) image.
- class IDOMTIFFImage

  IDOMTIFFImage interface.
- class IDOMPSDImage

  IDOMPSDImage interface.
- class IDOMWMPImage

  IDOMWMPImage interface.
- class IDOMRawImage

  Interface to a class representing a raw image.
- class IDOMPDFAlternateImage

  An encapsulation of a PDF alternate image, which may be referenced from an image or mask brush.
- class IDOMPDFImage

  Interface to a class representing an image extracted from a PDF file. Intended to be only used with the JawsMako APIs.
- class IDOMPCLImage

  Interface to a class representing an image extracted from a PCLXL file.
- class IDOMRecombineImage

  Interface to a class representing a image made up of separate single channel images (each with the same bps, dimensions and resolution) each representing a single component of the entire image, or a mask channel.
- class IDOMRecombineAlphaImage

  Similar to IDOMRecombineImage, but instead combines an image comprising the color components of the image, with a single-channel image that represents the mask or alpha channel. The images must have the same, dimensions, but may have different dimensions. The resolution information will be taken from the color image. Images with Indexed color spaces will be converted to the base spaces.
- class IDOMCompositeImage

  Interface to a class representing a image made up of separate images joined together vertically, appearing as a single image. All images must use the same color space, depth, width, and the same number of channels.
- class IDOMCachedImage

  Interface to a class that when overlayed over an image will cache portions of its output. Useful for cases where images are repeatedly and are relatively expensive to process, and where the source image is certain to never change.
- class IDOMImageChannelSelectorFilter

An image filter that presents optionally an image stripped of alpha, or alternatively a Gray image representing the extra channel (ie Alpha or Mask) or a device color space channel.

- class IDOMImageColorSpaceSubstitutionFilter

  An image filter that presents an identical image, just with the colourspace substituted.
- class IDOMImageColorConverterFilter

  An image filter that presents a colour converted version of an image.
- class IDOMImageBleederFilter

  An image filter that presents an image with the edge pixels repeated. Useful for cases where consumers may interpolate pixels at the edge, creating unwanted artifacts.
- class IDOMImageDownsamplerFilter

  An image filter that presents a downsampled version of an image.
- class IDOMImageMaskExpanderFilter

  *An image filter that presents a image source and color combination as a plain image with an alpha channel, with all pixels colored with the given color. Useful for simplifying an **IDOMMaskedBrush** where the brush masked by the image is a solid color.*
- class IDOMImageDeindexerFilter

  An image filter that presents an image with an Indexed colour space as a simple eight bit image.
- class IDOMImageDeviceNToBaseFilter

  An image filter that presents an image with a DeviceN colour space as a simple image in the alternate space.
- class IDOMImageSpotMergerFilter

  An image filter that merges selected spot components into the process components of an image. This is conceptually similar to IDOMImageDeviceNToBaseFilter, but:
- class IDOMImageInverterFilter

  An image filter that presents a bitwise inverted form of the source image.
- class IDOMDePremultiplyFilter

  An image filter that presents an image with premultiplied alpha as a plain image with plain alpha. It can be applied to any source image, and will do nothing if not required.
- class IDOMMatteRemoverFilter

  An image filter that removes a Matte and undoes premultiplication for a PDF Matte'd image and soft mask. The resulting image does not have alpha, and can be used with the mask to generate the desired result.
- class IDOMImageBitScalerFilter

  An image filter that presents an image as an image with a different bits per sample.
- class IDOMImageColorKeyFilter

  An image filter that presents a masked image where colours within a given range are masked out, analogous to a green screen. The source image must not have a mask or alpha channel.
- class IDOMImageDecodeFilter

  An image filter that applies a PDF/PS style Decode array to the image contents. For details on decode arrays, please see "Decode Arrays" on page 344 of the PDF Reference, version 1.7. The bit depth of the result may be promoted to eight or 16 bits per component depending on the situation.
- class IDOMFilteredImage

  IDOMFilteredImage interface. Provides a method for filtering of an underlying image without requiring converted image data to be stored. It maintains a list of filters that are successively applied.
- class IImageFrame

  IImageFrame encapsulates an EDL image with its details.
- class IImageFrameWriter

  IImageFrameWriter writes an image from an imageframe.
- class IImageDecoder

  IImageDecoder returns IImageFrame objects as requested by the client. This object knows about the imageformat internals and knows how to unpack the image.
- class IImageEncoder

  IImageEncoder accepts IImageFrame objects and streams it out to an image format.

Enumerations

- enum eDOMImageType {
  eDITUnknown = 0 , eDITJPEG , eDITJBIG2 , eDITJPEG2000 ,
  eDITPNG , eDITCCITTFax , eDITFlate , eDITLZW ,
  eDITRunLength , eDITRendered }

  Image type enumeration.
- enum eImageExtraChannelType

  Type used to uniquely identify the type of extra image channel.

6.1.4.9.1 Detailed Description

Classes to manage images.

6.1.4.10 Pages

Classes to create and manage Page and Fixed pages.

Classes

- class IDOMFixedPage

  Represents <FixedPage> element.
- class JawsMako::IPageRaster

  A rasterized equivalent of a page from an IDocument.
- class JawsMako::IPage

  A page from an IDocument, allowing high level page management, and providing on-demand access to page contents.

6.1.4.10.1 Detailed Description

Classes to create and manage Page and Fixed pages.

6.1.4.11 PDF-specific

PDF-specific object classes, such Optional Content Groups.

Topics

- Optional Content Groups

  Declarations of interfaces for querying and manipulating optional content. Optional content is a PDF feature allowing sections of graphical content to be made visible or visible when certain conditions are met. Most often this is used to add visible layers to a PDF document.
- PDF Objects

  PDF Files are essentially an object database consisting of a number of different kinds of objects which can be referred to indirectly by a reference consisting of an object and generation number. This header describes interfaces for these objects in Mako.
- Logical structure

  Declarations of interfaces for querying and manipulating logical structure, tagging, and marked content.
  Logical Structure is a PDF feature, built on marked content, allowing for tagging of content in a PDF file to facilitate reflowing, understanding the logical document structure, and improved accessibility.

Classes

- class IDOMSecurityInfo

  Base DOM security class.
- class IDOMStandardPDFSecurityInfo

  Represents security information from PDF Standard encryption handler.
- class IDOMPublicKeyPDFSecurityInfo

  Represents security information from PDF public key (certificate based) security handling. Inherits from IDOMStandardPDFSecurityInfo, from which it diverges in a modest fashion.

Enumerations

- enum IDOMStandardPDFSecurityInfo::ePermissionsFlags {
  IDOMStandardPDFSecurityInfo::ePrintAllowed = 0x0004 , IDOMStandardPDFSecurityInfo::eEditingAllowed
  = 0x0008 , IDOMStandardPDFSecurityInfo::eCopyingAllowed = 0x0010 , IDOMStandardPDFSecurityInfo::eAnnotationEditingAl
  = 0x0020 ,
  IDOMStandardPDFSecurityInfo::eFormFillingAllowed = 0x0100 , IDOMStandardPDFSecurityInfo::eContentAccessibilityExtracti
  = 0x0200 , IDOMStandardPDFSecurityInfo::eDocumentAssemblyAllowed = 0x0400 , IDOMStandardPDFSecurityInfo::eHighQu
  = 0x0800 ,
  IDOMStandardPDFSecurityInfo::eEverythingAllowed = 0x0ffc }

  Bit values for each permission flag as available from getPermissionFlags.

6.1.4.11.1 Detailed Description

PDF-specific object classes, such Optional Content Groups.

6.1.4.11.2 Enumeration Type Documentation

ePermissionsFlags

`enum IDOMStandardPDFSecurityInfo::ePermissionsFlags`

Bit values for each permission flag as available from getPermissionFlags.

An additional flag over and above that provided by IDOMStandardPDFSecurityInfo.

Enumerator

ePrintAllowed	Allow printing (bit 3)
eEditingAllowed	Allow editing (bit 4)
eCopyingAllowed	Allow copying (bit 5)
eAnnotationEditingAllowed	Allow annotation editing (bit 6)
eFormFillingAllowed	Allow form filling (bit 9)
eContentAccessibilityExtractionAllowed	Allow content extraction in support of accessibility (bit 10)
eDocumentAssemblyAllowed	Allow document assembly (bit 11)
eHighQualityPrintAllowed	Allow high-resolution printing (bit 12)
eEverythingAllowed	All operations permitted

6.1.4.11.3 Optional Content Groups

Declarations of interfaces for querying and manipulating optional content. Optional content is a PDF feature allowing sections of graphical content to be made visible or visible when certain conditions are met. Most often this is used to add visible layers to a PDF document.

Classes

- class JawsMako::IOptionalContent

  Root level optional content information for an entire document.
- class JawsMako::IOptionalContentDetails

  Interface for objects used to tag content as optional. Instances of this class are set in IDOMGroup instances to make those objects optional, linking them to one or more optional content groups.
- class JawsMako::IOptionalContentGroupReference

  A reference to an optional content group.
- class JawsMako::IOptionalContentGroup

  Interface for an optional content group.
- class JawsMako::IOptionalContentGroupUsage

  Usage information for an optional content group, providing context that an application can use to automatically show or hide content in the optional content group. This is optional.
- class JawsMako::IOptionalContentGroupUsageApplication

  Interface for controlling how IOptionalContentGroupUsage is applied, and for what groups.
- class JawsMako::IOptionalContentConfiguration

  A configuration for optional content.
- class JawsMako::IOptionalContentVisibilityExpression

  An interface representing a PDF 1.6+ visibility expression. Please refer to table 4.4.9 of the PDF 1.7 specification for background and detail.

Detailed Description

Declarations of interfaces for querying and manipulating optional content. Optional content is a PDF feature allowing sections of graphical content to be made visible or visible when certain conditions are met. Most often this is used to add visible layers to a PDF document.

Optional content is very powerful, and also complex. However, simple layering features can be added simply. In this header, complex aspects of the optional content scheme have been separated out.

6.1.4.11.4 PDF Objects

PDF Files are essentially an object database consisting of a number of different kinds of objects which can be referred to indirectly by a reference consisting of an object and generation number. This header describes interfaces for these objects in Mako.

Classes

- class JawsMako::IPDFObject

  Abstract interface for a PDF internal object and common interfaces. All non-composite objects are immutable. All PDF objects are hashable.
- class JawsMako::IPDFInteger

  A simple immutable integer PDF object type.
- class JawsMako::IPDFBoolean

  A simple immutable boolean PDF object type.
- class JawsMako::IPDFReal

  A simple immutable floating-point PDF object type.
- class JawsMako::IPDFNull

  A simple immutable "null" pdf object.
- class JawsMako::IPDFString

  A simple immutable string pdf object containing raw unencoded data or PDF Text information.
- class JawsMako::CPDFReference

  A simple concrete class representing indirect reference data.
- class JawsMako::IPDFReference

  A simple class representing an immutable PDF indirect reference.
- class JawsMako::CPDFFarReference

  A simple concrete class representing indirect reference data stored in a remote context, such as (for example) from a different PDF document.
- class JawsMako::IPDFFarReference

  A simple class representing an immutable PDF indirect reference from a remote context such as (for example) from a different PDF document.
- class JawsMako::IPDFArray

  A simple class representing a mutable array of other PDF objects.
- class JawsMako::IPDFDictionary

  A simple class representing a mutable dictionary of key-value pairs where the keys are PDF names and the values are PDF objects.
- class JawsMako::IPDFObjectStore

  A store of IPDFObjects holding a subset/subtree of an entire PDF object database, allowing storage, resolution and editing thereof. Every object store will have a "root" object from which the subtree or subset stems.

Enumerations

- enum JawsMako::ePDFObjectType {
  JawsMako::ePOTInteger = 0 , JawsMako::ePOTBoolean = 1 , JawsMako::ePOTReal = 2 , JawsMako::ePOTString = 3 ,
  JawsMako::ePOTName = 4 , JawsMako::ePOTArray = 5 , JawsMako::ePOTDictionary = 6 , JawsMako::ePOTNull = 7 ,
  JawsMako::ePOTReference = 8 , JawsMako::ePOTFarReference = 9 , JawsMako::ePOTStream = 10 ,
  JawsMako::ePOTOperator = 11 }

  An enumeration of PDF object types.

Detailed Description

PDF Files are essentially an object database consisting of a number of different kinds of objects which can be referred to indirectly by a reference consisting of an object and generation number. This header describes interfaces for these objects in Mako.

Exceptions of type IError are used throughout for error handling.

Enumeration Type Documentation

ePDFObjectType

`enum JawsMako::ePDFObjectType`

An enumeration of PDF object types.

Enumerator

ePOTInteger	A PDF Integer object.
ePOTBoolean	A PDF Boolean object.
ePOTReal	A PDF Real Object.
ePOTString	A PDF String object.
ePOTName	A PDF Name object.
ePOTArray	A PDF Array object.
ePOTDictionary	A PDF Dictionary object.
ePOTNull	A PDF NULL object.
ePOTReference	An indirect reference to a local PDF object stored in the same context.
ePOTFarReference	An indirect reference to a non-local PDF object.
ePOTStream	A PDF Stream object.
ePOTOperator	A PDF/PostScript Operator object.

6.1.4.11.5 Logical structure

Declarations of interfaces for querying and manipulating logical structure, tagging, and marked content.
Logical Structure is a PDF feature, built on marked content, allowing for tagging of content in a PDF file to facilitate reflowing, understanding the logical document structure, and improved accessibility.

Classes

- class JawsMako::IMarkedContentDetails

  Details of Marked Content applied to an IDOMGroup.
- class JawsMako::IMarkedContentStructureDetails

  A subclass of IMarkedContentDetails that is created when the marked content is associated with the document's structure.
- class JawsMako::IMarkedContentArtifactDetails

  A subclass of IMarkedContentDetails that is created when the content is a logical structure Artifact.
- class JawsMako::IStructureElementReference

  A token-like class encapsulating a reference to a structure element.
- class JawsMako::IStructure

  Top level tracking structure describing the logical structure of the document.
- class JawsMako::IStructureElement

  A structure element in the structure tree.
- class JawsMako::IStructureElementChild

  A child of a structure element. Either points to actual marked content, or another structure element.
- class JawsMako::IStructureElementReferenceChild

  A child of a structure element that points to another structure element.

- class JawsMako::IStructureMarkedContentReferenceChild

  A child of a structure element that points to a piece of marked content. Note; to create these, please see IStructureElement::createMarkedContentReferencePair()
- class JawsMako::IStructureObjectReferenceChild

  A child of a structure element that points to a piece of marked content. These cannot be created directly. Instead use IStructureElement::appendObjectReferenceChild() or IStructureElement::insertObjectReferenceChild()

Detailed Description

Declarations of interfaces for querying and manipulating logical structure, tagging, and marked content.
Logical Structure is a PDF feature, built on marked content, allowing for tagging of content in a PDF file to facilitate reflowing, understanding the logical document structure, and improved accessibility.

6.1.4.12 Outlines

Outlines represent indexes to specific locations in the document or an external location.

Classes

- class IDOMOutlineEntry

  Represents an index to a specific location in the document or a specific location external to the document.
- class IDOMOutline

  Represents the outline of the document, which is the collection of bookmarks for the document.
- class IDOMTarget

  Base class for defining hyperlink targets in a document.
- class IDOMExternalTarget

  IDOMExternalTarget interface.
- class IDOMInternalTarget

  The IDOMInternalTarget interface describes the targets of hyperlinks that are in the same document but not on the current page.
- class IDOMPageTarget

  IDOMPageTarget nodes are used to describe hyperlinks on a page to targets on the same page.
- class IDOMPageRectTarget

  IDOMPageRectTarget nodes are used to describe hyperlinks on a page rectangle to targets on the same page.
- class IDOMActionLaunch

  IDOMActionLaunch interface.
- class IDOMActionArray

  IDOMActionArray interface.

Enumerations

- enum IDOMOutlineEntry::eTextStyle { IDOMOutlineEntry::eTextStyleNone = 0x00 , IDOMOutlineEntry::eTextStyleItalic = 0x01 , IDOMOutlineEntry::eTextStyleBold = 0x02 , IDOMOutlineEntry::eTextStyleBoldItalic = eTextStyle↩ Italic | eTextStyleBold }

  Specifies an outline text style.
- enum IDOMTarget::eTargetType { IDOMTarget::eExternal , IDOMTarget::eInternal , IDOMTarget::ePage , IDOMTarget::ePageRect , IDOMTarget::eActionGoToR , IDOMTarget::eActionGoToE , IDOMTarget::eActionLaunch , IDOMTarget::eActionThread , IDOMTarget::eActionSound , IDOMTarget::eActionMovie , IDOMTarget::eActionHide , IDOMTarget::eActionNamed , IDOMTarget::eActionSubmitForm , IDOMTarget::eActionResetForm , IDOMTarget::eActionImportData , IDOMTarget::eActionJavaScript , IDOMTarget::eActionSetOCGState , IDOMTarget::eActionRendition , IDOMTarget::eActionTrans , IDOMTarget::eActionGoTo3DView , IDOMTarget::eActionArray }

  An enumeration of target types.
- enum IDOMPageRectTarget::eFitType

  Destination fit types enumeration.

6.1.4.12.1 Detailed Description

Outlines represent indexes to specific locations in the document or an external location.

6.1.4.12.2 Enumeration Type Documentation

eTargetType

`enum IDOMTarget::eTargetType`

An enumeration of target types.

Enumerator

eExternal	(PDF URI Action) Resolve a uniform resource identifier.
eInternal	XPS internal target.
ePage	XPS page target.
ePageRect	(PDF Goto Action) Go to a destination in the current document.
eActionGoToR	("Go-to remote") Go to a destination in another document.
eActionGoToE	("Go-to embedded"; PDF 1.6) Go to a destination in an embedded file.
eActionLaunch	Launch an application, usually to open a file.
eActionThread	Begin reading an article thread.
eActionSound	(PDF 1.2) Play a sound.
eActionMovie	(PDF 1.2) Play a movie.
eActionHide	(PDF 1.2) Set an annotation's Hidden flag.
eActionNamed	(PDF 1.2) Execute an action predefined by the viewer application.
eActionSubmitForm	(PDF 1.2) Send data to a uniform resource locator.
eActionResetForm	(PDF 1.2) Set fields to their default values.
eActionImportData	(PDF 1.2) Import field values from a file.

Enumerator

eActionJavaScript	(PDF 1.3) Execute a JavaScript script.
eActionSetOCGState	(PDF 1.5) Set the states of optional content groups.
eActionRendition	(PDF 1.5) Controls the playing of multimedia content.
eActionTrans	(PDF 1.5) Updates the display of a document, using a transition dictionary.
eActionGoTo3DView	(PDF 1.6) Set the current view of a 3D annotation
eActionArray	Array of actions.

eTextStyle

`enum IDOMOutlineEntry::eTextStyle`

Specifies an outline text style.

Enumerator

eTextStyleNone	Default text style.
eTextStyleItalic	Italic text style.
eTextStyleBold	Bold text style.
eTextStyleBoldItalic	Bold and italic text style.

6.1.4.13 Metadata

Document metadata, job and print tickets.

Classes

- class IEDLNamespace
  Interface to EDL Namespace class.
- class EDLQName
  Implementation of qualified name class.
- class IDOMJobTkOwner
  Interface to the IDOMJobTkOwner node.
- class IDOMJobTkGenericNode
  Interface to the IDOMJobTkGenericNode node.
- class IDOMJobTkGenericCharacterData
  Interface to the IDOMJobTkGenericCharacterData node.
- class IDOMJobTkNode
  Represents a Job Ticket Node.
- class IDOMJobTkValue
  Represents a Job Ticket value element.
- class IDOMJobTkContent
  Represents the content element of the JobTicket.
- class IDOMJobTk
  Represents an EDL JobTicket.

- class IDOMMetadata

  The IDOMMetadata interface provides access to the metadata attached to the DocumentSequence node. The IDOMMetadata interface is designed to be flexible enough to represent different types of metadata.
- class IDOMPrintTicket

  IDOMPrintTicket interface.

Enumerations

- enum eDOMJobTkLevel {
  eDOMJobTkLevelNotValid = 0 , eDOMJobTkLevelDefault = 1 , eDOMJobTkLevelJob = 2 , eDOMJobTkLevelDoc = 3 ,
  eDOMJobTkLevelPage = 4 , eDOMJobTkLevelCnt = 5 }

  DOMJobTk Job Ticket Level (docseq, doc, page)
- enum IDOMJobTkNode::eDOMJobTkNodeType

  DOMJobTk node (Property, Feature, Option, InitParam, ScoredProperty, ParamRef)
- enum IDOMMetadata::eType {
  IDOMMetadata::eCoreProperties = 0 , IDOMMetadata::eDocumentInfo , IDOMMetadata::eViewerPreferences , IDOMMetadata::ePageView ,
  IDOMMetadata::ePDFInfo , IDOMMetadata::eMetadataTypeCnt }

  Metadata types data type.
- enum IDOMMetadata::eXmpContainerType { IDOMMetadata::eXmpContainer_None = 0 , IDOMMetadata::eXmpContainer_Alt = 1 , IDOMMetadata::eXmpContainer_Bag = 2 , IDOMMetadata::eXmpContainer_Seq = 3 }

  The variant values passed to PValue when storing a CEDLStringVect.

6.1.4.13.1 Detailed Description

Document metadata, job and print tickets.

6.1.4.13.2 Enumeration Type Documentation

eType

`enum IDOMMetadata::eType`

Metadata types data type.

Enumerator

eCoreProperties	Core properties metadata (XPS core properties)
eDocumentInfo	Document information metadata (PDF document information dictionary)
eViewerPreferences	Viewer Preferences entry of PDF catalog dictionary.
ePageView	PageLayout and PageMode entries of PDF catalog dictionary.
ePDFInfo	Version and Linearization entries defined for PDF document.
eMetadataTypeCnt	The number of types defined by eType.

eXmpContainerType

`enum IDOMMetadata::eXmpContainerType`

The variant values passed to PValue when storing a CEDLStringVect.

Enumerator

eXmpContainer_None	The PValue variant that indicates it is not an Xmp container.
eXmpContainer_Alt	The PValue variant for storing a CEDLStringVect representing an Xmp Alt container.
eXmpContainer_Bag	The PValue variant for storing a CEDLStringVect representing an Xmp Bag container.
eXmpContainer_Seq	The PValue variant for storing a CEDLStringVect representing an Xmp Seq container.

6.1.4.14 Object Hashing

Hashes for object identification and comparison.

Classes

- class IDOMHashable

  Abstract interface for objects that can be hashed.
- class JawsMako::IHashable

  Simple interface to provide a consistent hashing method for Mako objects.

6.1.4.14.1 Detailed Description

Hashes for object identification and comparison.

6.1.4.15 Other DOM Objects

Other DOM objects.

Classes

- class IDOMAudioFile

  IDOMAudioFile interface.
- class IDOMRawDataFile

  IDOMRawDataFile interface.
- class IDOMResourceDictionary

  Interface to the EDL DOM's resource dictionary. The resource dictionary is a document resource that is shared between page markups. It holds a reference list of non-markup content that is shared between multiple pages of the document.

6.1.4.15.1 Detailed Description

Other DOM objects.

6.1.5 Input/Output

Classes concerned with reading input and writing output.

Topics

- Streams

  EDL provides a collection of "stream" classes that are supplied to, and returned by, EDL and JawsMako APIs that access files or data streams.
- Input

  Abstract input source that can open files from disk or a stream and create an IDocumentAssembly for the contents.
- Output

  Abstract output sink that can output DOM to a file or stream in a given output format.
- Generators

  Generators and renderers of Mako DOM content to vector formats.

6.1.5.1 Detailed Description

Classes concerned with reading input and writing output.

6.1.5.2 Streams

EDL provides a collection of "stream" classes that are supplied to, and returned by, EDL and JawsMako APIs that access files or data streams.

Classes

- class IEDLStream

  Generic stream. Abstract base class for EDL stream subsystem.
- class IInputStream

  Generic input stream. Abstract base class for all input streams.
- class IRAStream

  Abstract base class for "Random-Access" streams i.e. streams that can be arbitrarily re-positioned.
- class IPushbackStream

  Abstract base class (for input streams only) that provides a "push back" mechanism. When used with random access streams, the pushback buffer is invalidated by setPos().
- class IRAInputStream

  Random Access Input Stream.
- class IInputPushbackStream

  Input Stream with pushback support.
- class IRAInputPushbackStream

  Random-access Input Stream with pushback support.
- class IOutputStream

  Generic output stream. Abstract base class for output streams.
- class IRAOutputStream

  Random Access Output Stream.
- class EDLIFStream

  An ifstream that can deal with UTF8 file names on all platforms.
- class EDLOFStream

  An ofstream that can deal with UTF8 file names on all platforms.

6.1.5.2.1 Detailed Description

EDL provides a collection of "stream" classes that are supplied to, and returned by, EDL and JawsMako APIs that access files or data streams.

6.1.5.3 Input

Abstract input source that can open files from disk or a stream and create an IDocumentAssembly for the contents.

Classes

- class JawsMako::IDistiller

  An instance of the JawsMako Distiller class.
- class JawsMako::IIJPDSInput

  An instance of the JawsMako IJPDS input class.
- class JawsMako::IIJPDSPageRaster

  An instance of the JawsMako IJPDS page raster that contains the source input page number and rip number.
- class JawsMako::IInput

  Abstract input source that can open files from disk or a stream and create an IDocumentAssembly for the contents.
- class JawsMako::IPCL5Input

  An instance of the JawsMako PCL5 input class.
- class JawsMako::IPCLXLInput

  An instance of the JawsMako PCL/XL input class.
- class JawsMako::IPDFInput

  An instance of the JawsMako PDF input class.
- class JawsMako::IPJLParser

  A PJL (Printer Job Language) parser for Mako.
- class JawsMako::IPPMLInput

  An instance of the JawsMako PPML input class.
- class JawsMako::IPRNInput

  An instance of the JawsMako PRN input class.
- class JawsMako::IPSInput

  An instance of the JawsMako PS input class.
- class JawsMako::IXPSInput

  An instance of the JawsMako XPS input class.
- class JawsMako::IOXPSInput

  An instance of the JawsMako OpenXPS input class.

6.1.5.3.1 Detailed Description

Abstract input source that can open files from disk or a stream and create an IDocumentAssembly for the contents.

6.1.5.4 Output

Abstract output sink that can output DOM to a file or stream in a given output format.

Classes

- class JawsMako::IOutput

  Abstract output sink that can output DOM to a file or stream in a given output format.
- class JawsMako::IOutputIntent

  Interface class representing a PDF output intent.
- class JawsMako::IPCL5Output

  Interface for the PCL5 IOutput class.
- class JawsMako::IPCLXLOutput

  Interface for the PCLXL IOutput class.
- class JawsMako::IPDFOutput

  Interface for the PDF IOutput class.
- class JawsMako::IPSOutput

  Interface for the PS IOutput class.
- class JawsMako::IPSInjector

  Interface allowing users of IPSOutput to inject raw PostScript directly into the output stream at strategic points in the output process. Use IPSOutput::setInjector() to install subclasses of this type.
- class JawsMako::IXPSOutput

  Interface for the XPS IOutput class.
- class JawsMako::IOXPSOutput

  Interface for the OpenXPS IOutput class.

6.1.5.4.1 Detailed Description

Abstract output sink that can output DOM to a file or stream in a given output format.

6.1.5.5 Generators

Generators and renderers of Mako DOM content to vector formats.

Classes

- class JawsMako::ISkiaRenderer

  A renderer that can paint XPS compatible DOM into a Skia canvas using the Skia API.
- class JawsMako::ISVGGenerator

  An SVG generator for JawsMako, allowing simple generation of SVG fragments for individual DOM nodes or entire pages.
- class JawsMako::IXAMLGenerator

  A XAML generator for Mako, allowing simple generation of XAML fragments for individual DOM nodes or entire pages.

6.1.5.5.1 Detailed Description

Generators and renderers of Mako DOM content to vector formats.

6.1.6 Interactive features

Interactive objects such as annotations and forms.

Topics

- Annotations

  Comments and other annotations.
- Forms

  Interactive form elements.

6.1.6.1 Detailed Description

Interactive objects such as annotations and forms.

6.1.6.2 Annotations

Comments and other annotations.

Classes

- class JawsMako::IAnnotationReference

  A generic reference to an annotation. The target annotation might not be loaded. Chiefly used to refer to annotations from a Form.
- class JawsMako::IAnnotationAppearance

  An interface class for an annotation appearance, describing the graphical content of an annotation in a given usage and state. Annotation appearances are immutable.
- class JawsMako::CAnnotationBorder

  A class representing an annotation's border (described in the PDF Specification as BorderStyle). The meaning of a border style depends on the annotation type and not all annotation types will support all attributes of this class, and neither will all PDF versions support all attributes. Please refer to the PDF 1.7 specification for the required styles. JawsMako will only store the attributes that are valid for the given type, but will not signal errors for this case.
- class JawsMako::IAnnotation

  An interface class for an annotation.
- class JawsMako::IMarkupAnnotation

  An interface class for markup annotations. It is intended that future releases of JawsMako will extend this interface.
- class JawsMako::CQuadPoint

  A representation of a PDF Quadpoint, in DOM coordinates.
- class JawsMako::CRectInset

  A class which specifies an inset from a rectangle.
- class JawsMako::ITextMarkupAnnotation

  A generic interface class for a text markup annotation It is intended that future releases of JawsMako will extend this interface.
- class JawsMako::ILinkAnnotation

  A generic interface class for a link annotation It is intended that future releases of JawsMako will extend this interface.
- class JawsMako::IFreeTextAnnotation

  A generic interface class for a free text annotation It is intended that future releases of JawsMako will extend this interface.
- class JawsMako::ICaretAnnotation

  A generic interface class for a caret annotation It is intended that future releases of JawsMako will extend this interface.
- class JawsMako::IShapeAnnotation

  A generic interface class for circle and square annotations. It is intended that future releases of JawsMako will extend this interface.
- class JawsMako::IPolyAnnotation

An interface class for a polygon or polyline annotation. It is intended that future releases of JawsMako will extend this interface.

- class JawsMako::ILineAnnotation

  An interface class for a line annotation. It is intended that future releases of JawsMako will extend this interface.
- class JawsMako::IInkAnnotation

  A generic interface class for a ink annotation It is intended that future releases of JawsMako will extend this interface.
- class JawsMako::IPopupAnnotation

  An interface class for a popup annotation, which should not exist as a standalone, but is associated with a Markup Annotation. No appearances can be added to this annotation type. It is intended that future releases of JawsMako will extend this interface.
- class JawsMako::ISoundAnnotation

  An interface class for a sound annotation. Allows access to the sound as a WAV stream if the stream is embedded. It is intended that future releases of JawsMako will extend this interface.
- class JawsMako::ITextAnnotation

  A generic interface class for a text (sticky note) annotation.
- class JawsMako::IRedactionAnnotation

  A generic interface class for a redaction annotation.
- class JawsMako::IStampAnnotation

  A generic interface class for a stamp annotation.
- class JawsMako::IThreads

  An interface class for document threads, Currently a stub interface.
- class JawsMako::IAnnotationUtils
- class JawsMako::INamedDestination

  A named destination in a PDF Document.

Enumerations

- enum JawsMako::eAppearanceUsage { JawsMako::eAUNormal , JawsMako::eAURollover , JawsMako::eAUDown }

  The usage scenario for an annotation appearance.

6.1.6.2.1 Detailed Description

Comments and other annotations.

6.1.6.2.2 Enumeration Type Documentation

eAppearanceUsage

```
enum JawsMako::eAppearanceUsage
```

The usage scenario for an annotation appearance.

Enumerator

eAUNormal	The normal appearance is used when the annotation is not interacting with the user. This appearance is also used for printing the annotation.
eAURollover	The rollover appearance is used when the user moves the cursor into the annotation's active area without pressing the mouse button.
eAUDown	The down appearance is used when the mouse button is pressed or held down within the annotation's active area.

6.1.6.3 Forms

Interactive form elements.

Classes

- class JawsMako::CFieldOption

  Class to store form field option array entries.
- class JawsMako::CXFAPacket

  A class encapsulating an entry in an XFA array.
- class JawsMako::IFormField

  An interface class for a form field. A form field may have multiple child fields and widget annotations, arranged in a tree.
- class JawsMako::IForm

  An interface class for an interactive form, which tracks a tree of IFormFields and widgets.
- class JawsMako::IWidgetAppearanceCharacteristics

  Appearance information for a widget annotation.
- class JawsMako::IWidgetAnnotation

  An interface class for a widget annotation It is intended that future releases of JawsMako will extend this interface.

Enumerations

- enum JawsMako::eFieldType { JawsMako::eFTInherited , JawsMako::eFTButton , JawsMako::eFTText , JawsMako::eFTChoice , JawsMako::eFTSignature }

  The type of a form field. These map to field types present in the pdf specification.
- enum JawsMako::eFieldFlags

  Flags used in form fields.
- enum JawsMako::eQuadding

  The type of quadding (justification) used for variable text in a form field.
- enum JawsMako::IWidgetAnnotation::eHighlightMode

  The highlight mode for a Widget annotation.
- enum JawsMako::IWidgetAnnotation::eFontEncoding

  Encoding scheme used for form fonts.

6.1.6.3.1 Detailed Description

Interactive form elements.

6.1.6.3.2 Enumeration Type Documentation

eFieldType

```
enum JawsMako::eFieldType
```

The type of a form field. These map to field types present in the pdf specification.

Enumerator

eFTInherited	Inherited.
eFTButton	Button.
eFTText	Text box.
eFTChoice	Checkbox.
eFTSignature	Digital signature.

6.1.7 Utility

Utility classes.

Topics

- Platform-dependent

  Platform-dependent functions that are visible through the EDL API.
- Transforms

  Transforms.
- Others

  Other utility classes and functions.

6.1.7.1 Detailed Description

Utility classes.

6.1.7.2 Platform-dependent

Platform-dependent functions that are visible through the EDL API.

Files

- file platform.h

  Platform-dependent defines, enumerations, types etc. that are visible through the EDL API.

Functions

- _BEGIN_EDL_NAMESPACE EDL_API int edlMkdir (const char ∗dir)

  Creates a subdirectory path (including parents) in an existing directory tree.
- EDL_API int edlRmdir (const char ∗dir)

  Removes a subdirectory, if it is empty.
- EDL_API char ∗ edlGetTemporaryDirectory (void)

  Get the platform-specific temporary directory path. This is created by consulting platform specific APIs or through the TEMP or TMP environment variables.
- EDL_API char ∗ edlMakeTempDir (const char ∗subdir, int pidNo)

  Creates an EDL temporary directory as specified by the two parameters. The temporary directory will be created as a subdirectory of the root temporary directory, which is specified by either the `TEMP` or `TMP` environment variables on most platforms, with platform specific methods used on iOS and UWP instead. For example, if `TMP` is set to c:\tmp and you invoke `edlMakeTempDir("edl", 2)` then it will create c:\tmp\edl\2.
- EDL_API char ∗ edlExclusiveMakeTempDir (const char ∗subdir, int pidNo)

  As edlMakeTempDir, but fails if the directory exists.
- EDL_API char ∗ edlMakeTempDirProvidingSubDirPath (const char ∗subdir, int pidNo, char ∗∗fullSubDir)

  As edlMakeTempDir, but also provides the UTF8 path to the subdir to the pointer at address fullSubDir. This must be freed by the caller using `free()`. This is useful if the user wishes to delete that directory. If the subdir parameter was an empty string or nothing but path separators, then fullSubDir will be NULL on exit.
- EDL_API int edlGetProcessId ()

  Gets the process id of the calling process. This is useful if the user wants to provide separate temporary directories for each instance of EDL.
- EDL_API FILE ∗ edlFopen (const char ∗filename, const char ∗mode)

  Open a file as per fopen, handling UTF8 file names on all platforms.
- EDL_API int edlVsnprintf (char ∗buffer, size_t n, const char ∗format, va_list ap)

  Implementation of vsnprintf that always uses the C locale, to write formatted data from variable argument list to sized buffer Parameters as per vsnprintf()
- EDL_API int edlSnprintf (char ∗buffer, size_t n, const char ∗format,...)

  Implementation of snprintf that always uses the C locale. Parameters as per snprintf().
- EDL_API double edlStrtod (const char ∗str, char ∗∗endptr)

  Implementation of strtod that always uses the C locale to parse the C string str interpreting its content as a floating point number as per strtod()

6.1.7.2.1 Detailed Description

Platform-dependent functions that are visible through the EDL API.

6.1.7.2.2 Function Documentation

edlExclusiveMakeTempDir()

```
EDL_API char * edlExclusiveMakeTempDir (
            const char * subdir,
            int pidNo )
```

As edlMakeTempDir, but fails if the directory exists.

Parameters

subdir	Specifies the first part of the path to the temporary directory to be created. The path is treated as a UTF8 string.
pidNo	Specifies the second part of the path to the temporary directory to be created. This parameter allows multiple instances of the application to run on the same machine whilst maintaining independent temporary directories.

Returns

A pointer to a string containing the path of the created directory. If the call fails, the pointer will be NULL. It is the responsibility of the caller to release the memory pointed to by the return value by calling `free()`.

edlFopen()

```
EDL_API FILE * edlFopen (
            const char * filename,
            const char * mode )
```

Open a file as per fopen, handling UTF8 file names on all platforms.

Parameters

filename	The path to the file.
mode	An fopen mode string.

Returns

FILE A file pointer, or NULL on failure.

edlGetProcessId()

```
EDL_API int edlGetProcessId ( )
```

Gets the process id of the calling process. This is useful if the user wants to provide separate temporary directories for each instance of EDL.

Returns

int The process id of the calling process.

edlGetTemporaryDirectory()

```
EDL_API char * edlGetTemporaryDirectory (
            void  )
```

Get the platform-specific temporary directory path. This is created by consulting platform specific APIs or through the TEMP or TMP environment variables.

Returns

A pointer to a string containing the temporary directory path. If the call fails NULL is returned. It is the responsibility of the caller to release the memory pointed to by the return value by calling free().

edlMakeTempDir()

```
EDL_API char * edlMakeTempDir (
            const char * subdir,
            int pidNo )
```

Creates an EDL temporary directory as specified by the two parameters. The temporary directory will be created as a subdirectory of the root temporary directory, which is specified by either the `TEMP` or `TMP` environment variables on most platforms, with platform specific methods used on iOS and UWP instead. For example, if `TMP` is set to c:\tmp and you invoke `edlMakeTempDir`("edl", 2) then it will create c:\tmp\edl\2.

Parameters

subdir	Specifies the first part of the path to the temporary directory to be created. The path is treated as a UTF8 string.
pidNo	Specifies the second part of the path to the temporary directory to be created. This parameter allows multiple instances of the application to run on the same machine whilst maintaining independent temporary directories.

Returns

A pointer to a string containing the path of the created directory. If the call fails, the pointer will be NULL. It is the responsibility of the caller to release the memory pointed to by the return value by calling `free()`.

edlMakeTempDirProvidingSubDirPath()

```
EDL_API char * edlMakeTempDirProvidingSubDirPath (
            const char * subdir,
            int pidNo,
            char ** fullSubDir )
```

As edlMakeTempDir, but also provides the UTF8 path to the subdir to the pointer at address fullSubDir. This must be freed by the caller using `free()`. This is useful if the user wishes to delete that directory. If the subdir parameter was an empty string or nothing but path separators, then fullSubDir will be NULL on exit.

Parameters

subdir	Specifies the first part of the path to the temporary directory to be created. The path is treated as a UTF8 string.
pidNo	Specifies the second part of the path to the temporary directory to be created. This parameter allows multiple instances of the application to run on the same machine whilst maintaining independent temporary directories.
fullSubDir	The pointer to the address of a UTF8 path to the subdir

Returns

A pointer to a string containing the path of the created directory. If the call fails, the pointer will be NULL. It is the responsibility of the caller to release the memory pointed to by the return value by calling `free()`.

edlMkdir()

```
_BEGIN_EDL_NAMESPACE EDL_API int edlMkdir (
            const char * dir )
```

Creates a subdirectory path (including parents) in an existing directory tree.

Parameters

dir	Specifies the path to be created. If the parameter is specified as /foo/bar/dir, and the path /foo/bar already exists, the directory dir will be created within bar. If the parameter is specified as /foo/bar and /foo already exists, then both bar and bar/dir are created, and so forth. The path is treated as a UTF8 string.

Returns

int Non-zero on success, zero if the call fails.

edlRmdir()

```
EDL_API int edlRmdir (
             const char * dir )
```

Removes a subdirectory, if it is empty.

Parameters

dir	Specifies the path to, and name of, the directory to be removed. The path is treated as a UTF8 string.

Returns

int Non-zero on success, zero if the call fails.

edlSnprintf()

```
EDL_API int edlSnprintf (
             char * buffer,
             size_t n,
             const char * format,
              ...  )
```

Implementation of snprintf that always uses the C locale. Parameters as per snprintf().

Parameters

buffer	C string to accept the result
n	Buffer size
format	Format string

Returns

int Either the number of characters printed (not including the null terminator) or the number of characters that would have been printed if the target buffer was unlimited in size.

edlStrtod()

```
EDL_API double edlStrtod (
             const char * str,
             char ** endptr )
```

Implementation of strtod that always uses the C locale to parse the C string str interpreting its content as a floating point number as per strtod()

Parameters

str	C string with the representation of a floating-point number.
endptr	Reference to an already allocated object of type char *, whose value is set by the function to the next character in str after the numerical value. This parameter can also be a null pointer, in which case it is not used.

Returns

double Either the converted floating point number, or 0.0 if the conversion failed.

edlVsnprintf()

```
EDL_API int edlVsnprintf (
              char * buffer,
              size_t n,
              const char * format,
              va_list ap )
```

Implementation of vsnprintf that always uses the C locale, to write formatted data from variable argument list to sized buffer Parameters as per vsnprintf()

Parameters

buffer	C string to accept the result
n	Buffer size
format	Format string
ap	Variable argument list

Returns

int Either the number of characters printed (not including the null terminator) or the number of characters that would have been printed if the target buffer was unlimited in size.

6.1.7.3 Transforms

Transforms.

Classes

- class JawsMako::ICustomTransform

  A transform that allows the implementation to be provided externally.
- class JawsMako::CTransformState

  Class for tracking the graphics state leading to the point where a transform is applied.
- class JawsMako::ITransform

  ITransforms provide a method of applying common operations on DOM objects such as brushes, nodes, colors, colorspaces or entire trees. Not all transforms will operate on all kinds of objects, as noted in their descriptions.
- class JawsMako::ITransformChain

ITransformChain represents a change of ITransforms, and provides a method of applying a range of transforms to an entire DOM tree. Instances of this type attempt to ensure that shared resources are modified once only.

- class JawsMako::IImageEncoderTransform

  A simple transform for image encoding. Most useful for encoding abstract images such as IDOMRecombineImage, IDOMRawImage and IDOMFilteredImage as PNG, Tiff or Jpeg. Images may be color converted if they are not compatible with the desired image type.

- class JawsMako::IImageDownsamplerTransform

  A transform for downsampling images above a given effective resolution to a desired target effective resolution.

- class JawsMako::IColorConverterTransform

  A transform for color conversion, converting all appropriate DOM contents to a desired target color space.

- class JawsMako::IComplexColorSimplifierTransform

  A simple transform that looks for DeviceN or Indexed color spaces, and where found, simplifies the hosting objects to use the underlying color space (for Indexed cases) or the alternate color space (for DeviceN cases). Useful in particular for consumers that do not support such color spaces. DOM brushes using only the /None colorant in a DeviceN colorspace may be dropped entirely.

- class JawsMako::IImageMergerTransform

  A simple transform that looks for nearby images and attempts to glom them together in a single image. Some producers can break images up into images consisting of a single scanline; this transform attempts to put them back together again. This transform can handle images with a mask channel, but does not attempt to merge images with an alpha channel.

- class JawsMako::IOptionalContentFixerTransform

  A simple transform that strips the DOM of any PDF optional content that is not visible for the given document use. This transform also selects from any alternate images, if present.

- class JawsMako::ICFFCIDSplitterTransform

  A simple transform that looks for CID CFF Fonts containing multiple SubFonts. Some viewers do not support these fonts, or do so poorly. If found, this transform will split out the sub fonts into individual font streams, and adjust the Glyphs nodes where they are used accordingly.

- class JawsMako::ICFFSanitizerTransform

  A simple transform that scans and sanitises CFF Fonts for wider interoperability.

- class JawsMako::IStrokerTransform

  A transform for converting some or all stroked paths into plain filled paths.

- class JawsMako::IFormUnpackerTransform

  A transform for unpacking an IDOMFormInstance directly into the DOM tree. That is, in the DOM tree the IDOMFormInstance is replaced with the unpacked contents of the referenced IDOMForm.

- class JawsMako::IRendererTransform

  A transform for selective rendering of sections of a DOM tree, replacing the rendered items with an image representation. Currently only operates on IDOMFixedPages; this restriction should be eased in future versions.

- class JawsMako::IRedactorTransform

  A transform for applying redaction redactions.

- class JawsMako::IType3UnpackerTransform

  A transform for unpacking glyphs using a Type 3 font into regular DOM.

- class JawsMako::IOverprintSimulationTransform

  A transform that modifies DOM such that any overprint present in the DOM will be visible when written or rendered in an environment that does not support overprint.

- class JawsMako::IPatternConverterTransform

  Transform to convert PDF/PS tiling/shading patterns to XPS- compatible forms where possible.

- class JawsMako::IFontProcessorTransform

  A transform for performing font subsetting and merging.

- class JawsMako::IFontProcessorDeferredTransform

  A transform for performing font subsetting and merging, but it only replaces modified fonts with placeholders. These fonts will /not/ be usable until finaliseFonts() is called.

- class JawsMako::IBlendSimplifierTransform

  A transform to transform linear or radial gradients with repeat or reflect padding to simpler types. For linear gradients this means a simple linear gradient inside a tiling visual brush. For radial gradients, the stops must be repeated as required.

- class JawsMako::ISoftMaskConverterTransform

  A transform that converts PDF style soft masks to opacity masks, suitable for XPS.
- class JawsMako::IFormDeduplicatorTransform

  A transform that attempts to deduplicate graphically identical IDOMForm objects. Note that non-graphical properties, such as additional dictionary entries, structure information, etc, are not consulted when looking for identical forms.
- class IDOMMatrix

  Defines the render transform matrix.

Typedefs

- typedef CEDLVector< ITransformPtr > **JawsMako::CTransformVect**

  A vector of transform instances.

Enumerations

- enum JawsMako::eBrushUsage { JawsMako::eBUNone = 0 , JawsMako::eBUGeneral , JawsMako::eBUFill , JawsMako::eBUStroke , JawsMako::eBUOpacityMask , JawsMako::eBUAlternateImage }

  When a CTransformState has been pushed for a brush, this indicates the usage of that brush. If we descend into a brush, this allows the transform to know what kind of brush it is in.

Functions

- static JAWSMAKO_API ICustomTransformPtr JawsMako::ICustomTransform::create (const IJawsMakoPtr &jawsMako, IImplementation *implementation, const IAbortPtr &abort=IAbortPtr(), bool dependsOnClip↩ Bounds=true, bool dependsOnGroupSpace=true, bool dependsOnRenderingIntent=true, bool depends↩ OnTransform=true, bool dependsOnBrushUsage=true, bool dependsOnEdgeMode=true, bool dependsOn↩ UncoloredTilingBrush=true)

  Create an ICustomTransform object whose underlying methods are supported by the implementation object. Note - this creation routine will be removed in a future version of Mako and replaced with a version that is equivalent to calling this function with default arguments.
- static JAWSMAKO_API ICustomTransformPtr JawsMako::ICustomTransform::create (const IJawsMakoPtr &jawsMako, IImplementation *implementation, const IAbortPtr &abort, bool dependsOnClipBounds, bool dependsOnGroupSpace, bool dependsOnRenderingIntent, bool dependsOnTransformScale, bool depends↩ OnTransformOffset, bool dependsOnBrushUsage, bool dependsOnEdgeMode, bool dependsOnUncolored↩ TilingBrush)

  Create an ICustomTransform object whose underlying methods are supported by the implementation object.

6.1.7.3.1 Detailed Description

Transforms.

6.1.7.3.2 Enumeration Type Documentation

eBrushUsage

`enum JawsMako::eBrushUsage`

When a CTransformState has been pushed for a brush, this indicates the usage of that brush. If we descend into a brush, this allows the transform to know what kind of brush it is in.

Enumerator

eBUNone	No usage indicated.
eBUGeneral	General brush.
eBUFill	Indicates a fill.
eBUStroke	Indicates a stroke.
eBUOpacityMask	Indicates an opacity mask.
eBUAlternateImage	Indicates an alternate image.

6.1.7.3.3 Function Documentation

create() [1/2]

```
static JAWSMAKO_API ICustomTransformPtr JawsMako::ICustomTransform::create (
            const IJawsMakoPtr & jawsMako,
            IImplementation * implementation,
            const IAbortPtr & abort,
            bool dependsOnClipBounds,
            bool dependsOnGroupSpace,
            bool dependsOnRenderingIntent,
            bool dependsOnTransformScale,
            bool dependsOnTransformOffset,
            bool dependsOnBrushUsage,
            bool dependsOnEdgeMode,
            bool dependsOnUncoloredTilingBrush )  [static]
```

Create an ICustomTransform object whose underlying methods are supported by the implementation object.

Note that internally the transform mechanism uses a caching scheme such that your callback functions are called as infrequently as possible. For example, if your callback edits an IDOMForm, the transform mechanism will attempt to cache the result and avoid calling transformForm() for the same form again if it believes that the results will be the same. This is the purpose of the dependsOn parameters which tell the mechanism whether or not your transform will produce the same results under these circumstances. This will also affect the information passed to the callbacks in the CTransformState.

In this form of the creation routine, there are no default arguments for these dependency arguments. This is to ensure that should the number of dependency parameters change there will be no surprises for existing code.

Parameters

jawsMako	A smart pointer to the JawsMako instance.
implementation	The custom transformation implementation.
abort	An abort callback to allow your transformation to be aborted.
dependsOnClipBounds	If your transform implementation will return the same results regardless of the clip affecting the object being transformed, set this to false. Otherwise set this to true.
dependsOnGroupSpace	If your transform implementation will return the same results regardless of the transparency group color space affecting the object being transformed, set this to false. Otherwise set this to true.
dependsOnRenderingIntent	If your transform implementation will return the same results regardless of the current rendering intent affecting the object being transformed, set this to false. Otherwise set this to true.

Parameters

dependsOnTransformScale	If your transform implementation will return the same results regardless of how the object being scaled, rotated, or skewed, (that is, transformed by the xx, xy, yx and yy portions of the effective render transform affecting the object) set this to false. Otherwise set this to true.
dependsOnTransformOffset	If your transform implementation will return the same results regardless of how the object is being offset (that is translated by the dx and dy portions of the render transform affecting the object). Otherwise set this to true. Normally, if dependsOnTransformOffset is set, then dependsOnTransformScale is typically also set.
dependsOnBrushUsage	If your transform implementation will return the same results for a brush regardless of what purpose the brush is used for, set this to false. Otherwise set this to true.
dependsOnEdgeMode	If your transform implementation will return the same results regardless of what edge mode is affecting the object being transformed, set this to false. Otherwise set this to true.
dependsOnUncoloredTilingBrush	If your transform implementation will return the same results regardless of whether or not the object is in a colored or uncolored tiling pattern brush, set this to false. Otherwise set this to true.

Returns

ICustomTransformPtr A smart pointer to the ICustomTransform object.

create() [2/2]

```
static JAWSMAKO_API ICustomTransformPtr JawsMako::ICustomTransform::create (
              const IJawsMakoPtr & jawsMako,
              IImplementation * implementation,
              const IAbortPtr & abort = IAbortPtr (),
              bool dependsOnClipBounds = true,
              bool dependsOnGroupSpace = true,
              bool dependsOnRenderingIntent = true,
              bool dependsOnTransform = true,
              bool dependsOnBrushUsage = true,
              bool dependsOnEdgeMode = true,
              bool dependsOnUncoloredTilingBrush = true )  [inline], [static]
```

Create an ICustomTransform object whose underlying methods are supported by the implementation object. Note - this creation routine will be removed in a future version of Mako and replaced with a version that is equivalent to calling this function with default arguments.

For fine control over dependencies, please see the alternate form below.

Note that internally the transform mechanism uses a caching scheme such that your callback functions are called as infrequently as possible. For example, if your callback edits an IDOMForm, the transform mechanism will attempt to cache the result and avoid calling transformForm() for the same form again if it believes that the results will be the same. This is the purpose of the dependsOn parameters which tell the mechanism whether or not your transform will produce the same results under these circumstances. This will also affect the information passed to the callbacks in the CTransformState.

Parameters

jawsMako	A smart pointer to the JawsMako instance.
implementation	The custom transformation implementation.
abort	An abort callback to allow your transformation to be aborted.
dependsOnClipBounds	If your transform implementation will return the same results regardless of the clip affecting the object being transformed, set this to false. Otherwise set this to true.
dependsOnGroupSpace	If your transform implementation will return the same results regardless of the transparency group color space affecting the object being transformed, set this to false. Otherwise set this to true.
dependsOnRenderingIntent	If your transform implementation will return the same results regardless of the current rendering intent affecting the object being transformed, set this to false. Otherwise set this to true.
dependsOnTransform	If your transform implementation will return the same results regardless of how the object being transformed is positioned, rotated, or scaled, set this to false. Otherwise set this to true. Note that setting this to true has the same effect as setting dependsOnTransformScale and dependsOnTransformOffset set to true in the alternate form of the creator below.
dependsOnBrushUsage	If your transform implementation will return the same results for a brush regardless of what purpose the brush is used for, set this to false. Otherwise set this to true.
dependsOnEdgeMode	If your transform implementation will return the same results regardless of what edge mode is affecting the object being transformed, set this to false. Otherwise set this to true.
dependsOnUncoloredTilingBrush	If your transform implementation will return the same results regardless of whether or not the object is in a colored or uncolored tiling pattern brush, set this to false. Otherwise set this to true.

Returns

ICustomTransformPtr A smart pointer to the ICustomTransform object.

6.1.7.4 Others

Other utility classes and functions.

Classes

- class IDOMFunction

  Base class for PDF/PS Style functions All function instances throw IEDLError exceptions on failure.
- class IDOMSampledFunction

  Interface for sampled functions. See section 3.9.1 of the PDF 1.7 Reference. Default values are as per described in that reference.
- class IDOMExponentialFunction

  Interface for exponential functions. See section 3.9.2 of the PDF 1.7 Reference. Default values are as per described in that reference. There can be only one input for this function type.
- class IDOMStitchingFunction

  Interface for stitching functions. See section 3.9.3 of the PDF 1.7 Reference. Default values are as per described in that reference. There can only be one input for this function, and the functions contained therein must also handle one input.

- class IDOMGroupingFunction

  Interface to encapsulate an array of x-input-1-output functions.
- class IDOMPostScriptCalculatorFunction

  Interface for PostScript calculator functions. See section 3.9.4 of the PDF 1.7 Reference. Default values are as per described in that reference.

6.1.7.4.1 Detailed Description

Other utility classes and functions.

6.1.8 Rendering

Classes and methods for rendering content.

Topics

- Transparency blend modes

  An enum for transparency blend modes.
- Halftones

  An abstract base class for communicating halftone information to the Jaws renderer, for use with renderScreened() and renderScreenedToFrameBuffers()

Classes

- class JawsMako::IJawsRenderer

  A renderer that uses the Jaws RIP to create images from arbitrary DOM.

6.1.8.1 Detailed Description

Classes and methods for rendering content.

6.1.8.2 Transparency blend modes

An enum for transparency blend modes.

Enumerations

- enum eBlendMode {
  eBlendModeUnspecified , eBlendModeNormal , eBlendModeMultiply , eBlendModeScreen ,
  eBlendModeOverlay , eBlendModeSoftLight , eBlendModeHardLight , eBlendModeColorDodge ,
  eBlendModeColorBurn , eBlendModeDarken , eBlendModeLighten , eBlendModeDifference ,
  eBlendModeExclusion , eBlendModeHue , eBlendModeSaturation , eBlendModeColor ,
  eBlendModeLuminosity }

  An enum for transparency blend modes.

6.1.8.2.1 Detailed Description

An enum for transparency blend modes.

6.1.8.2.2 Enumeration Type Documentation

eBlendMode

```
enum eBlendMode
```

An enum for transparency blend modes.

Enumerator

eBlendModeUnspecified	Unspecified blend mode.
eBlendModeNormal	Selects the source color, ignoring the backdrop.
eBlendModeMultiply	Multiplies the backdrop and source color values.
eBlendModeScreen	Multiplies the complements of the backdrop and source color values, then complements the result
eBlendModeOverlay	The backdrop color is not replaced but is mixed with the source color to reflect the lightness or darkness of the backdrop
eBlendModeSoftLight	Darkens or lightens the colors, depending on the source color value.
eBlendModeHardLight	Multiplies or screens the colors, depending on the source color value.
eBlendModeColorDodge	Brightens the backdrop color to reflect the source color.
eBlendModeColorBurn	Darkens the backdrop color to reflect the source color.
eBlendModeDarken	Selects the darker of the backdrop and source colors.
eBlendModeLighten	Selects the lighter of the backdrop and source colors.
eBlendModeDifference	Subtracts the darker of the two constituent colors from the lighter color.
eBlendModeExclusion	Produces an effect similar to that of the Difference mode but lower in contrast
eBlendModeHue	Creates a color with the hue of the source color and the saturation and luminosity of the backdrop color
eBlendModeSaturation	Creates a color with the saturation of the source color and the hue and luminosity of the backdrop color
eBlendModeColor	Creates a color with the hue and saturation of the source color and the luminosity of the backdrop color
eBlendModeLuminosity	Creates a color with the luminosity of the source color and the hue and saturation of the backdrop color

6.1.8.3 Halftones

An abstract base class for communicating halftone information to the Jaws renderer, for use with renderScreened() and renderScreenedToFrameBuffers()

Classes

- class JawsMako::IJawsRenderer::CSpotHalftone

Description of a simple spot halftone, at 45 degrees, using Jaws's default spot function. Used for monochrome rendering.

- class JawsMako::IJawsRenderer::CColorSpotHalftone

  Description of spot halftones, using Jaws's default spot function. Used for color halftoned rendering. Analogous to a PostScript Type 2 Halftone; please refer to section 7.4.6 of the PostScript language reference manual. When rendering to RGB, there must be three angles. (Red, Green and Blue respectively). For CMYK output, there must be four (Cyan, Magenta, Yellow and Black respectively).
- class JawsMako::IJawsRenderer::CThresholdArrayHalftone

  Description of a Type 3 8-bit threshold array halftone for use with monochrome rendering. Please refer to section 7.4.5 of the PostScript language reference manual, 3rd edition.
- class JawsMako::IJawsRenderer::CColorThresholdArrayHalftone

  As per CThresholdArrayHalftone, but for use with color rendering. There is a single width and height, but a threshold array for each color being rendering. One threshold array must be specified for each color being rendered. When rendering to RGB, there must be three thresholds (Red, Green and Blue respectively). For CMYK output, there must be four thresholds (Cyan, Magenta, Yellow and Black respectively).
- class JawsMako::IJawsRenderer::CThresholdHalftone

  A halftone representing a simple threshold. Used for monochrome rendering.
- class JawsMako::IJawsRenderer::CEDSHalftone

  A halftone representing an error diffusion screen. Allows the production of results containing a variable number of gray levels per channel using a range of error diffusion screens.

6.1.8.3.1 Detailed Description

An abstract base class for communicating halftone information to the Jaws renderer, for use with renderScreened() and renderScreenedToFrameBuffers()

6.1.9 Layout

Classes to perform layout of text and produce fixed content DOM.

Classes

- class IFrame

  A frame, into which text can be flowed by the layout engine.
- class JawsMako::ILayoutParagraph

  A paragraph, consisting of a number of runs of content.
- class JawsMako::ILayoutRun

  A run of content to be added to an ILayoutParagraph.
- class JawsMako::ILayoutTextRun

  A run of text content to be added to an ILayoutParagraph.
- class JawsMako::ILayoutImageRun

  A run of image content to be added to an ILayoutParagraph.
- class JawsMako::ILayoutFontWeight

  Font weights used by ILayoutFont for font selection.
- class JawsMako::ILayoutFont

  ILayoutFont interface.
- class JawsMako::ILayout

  An instance of a Layout engine that can layout and flow text into one or more frames, resulting in DOM that describes the final laid-out text.

6.1.9.1 Detailed Description

Classes to perform layout of text and produce fixed content DOM.

6.1.10 Colors

Classes and methods for dealing with color spaces and values.

Topics

- Color value

  A single color value. The color values themselves are held as floating point values for all color spaces. For some spaces (such as indexed color spaces) the values will be integral, but still stored as floats.
- Color space

  A color space describes a range of colors appropriate to a given context.
  A device color space describes the range of colors that a camera can see, a printer can print, or a monitor can display, and is tied to the device it describes.
  Editing color spaces, on the other hand, such as Adobe RGB or sRGB, are device-independent, determining a color range you can work in. They are designed to be gray balanced, such that colors with equal amounts of red, green, and blue appear neutral, and allow you to edit images in a controlled, consistent manner. Editing spaces also are perceptually uniform; that is, changes to lightness, hue, or saturation are applied equally to all the colors in the image.
- Color profiles

  A color profile provides an accurate description of the characteristics of a digital device or working color space.

6.1.10.1 Detailed Description

Classes and methods for dealing with color spaces and values.

6.1.10.2 Color value

A single color value. The color values themselves are held as floating point values for all color spaces. For some spaces (such as indexed color spaces) the values will be integral, but still stored as floats.

Classes

- class IDOMColor

  Holds a single color value. The color values themselves are held as floating point values for all color spaces. For some spaces (such as indexed color spaces) the values will be integral, but still stored as floats.

Enumerations

- enum eBlackPointCompensation { eBPCDefault = 0 , eBPCOn = 1 , eBPCOff = 2 }

  Black point compensation enumeration.
- enum eRenderingIntent { ePerceptual = 0 , eRelativeColorimetric = 1 , eSaturation = 2 , eAbsoluteColorimetric = 3 }

  Rendering intent enumeration.

6.1.10.2.1 Detailed Description

A single color value. The color values themselves are held as floating point values for all color spaces. For some spaces (such as indexed color spaces) the values will be integral, but still stored as floats.

The color values themselves cannot be interpreted without reference to the referenced color space.

IDOMColor nodes are used to specify the colors of lines and strokes, not of images. Hence if an ArcSegment is drawn with a blue line, the blue is specified by an IDOMColor node.

6.1.10.3 Color space

A color space describes a range of colors appropriate to a given context.
A device color space describes the range of colors that a camera can see, a printer can print, or a monitor can display, and is tied to the device it describes.
Editing color spaces, on the other hand, such as Adobe RGB or sRGB, are device-independent, determining a color range you can work in. They are designed to be gray balanced, such that colors with equal amounts of red, green, and blue appear neutral, and allow you to edit images in a controlled, consistent manner. Editing spaces also are perceptually uniform; that is, changes to lightness, hue, or saturation are applied equally to all the colors in the image.

Classes

- class IDOMColorSpace

  IDOMColorSpace interface.
- class IDOMColorSpacesRGB

  Represents the RGB color space.
- class IDOMColorSpacesGray

  Represents a gray color space using the sRGB gamma and WhitePoint.
- class IDOMColorSpacescRGB

  Represents the scRGB color space.
- class IDOMColorSpaceDeviceRGB

  IDOMColorSpaceDeviceRGB interface.
- class IDOMColorSpaceDeviceGray

  IDOMColorSpaceDeviceGray interface.
- class IDOMColorSpaceDeviceCMYK

  Represents the default CMYK color space.
- class IDOMColorSpaceDeviceCMY

  Represents the default CMY color space. NOTE: Currently for internal use only; Do not use this color space in your own applications.
- class IDOMColorSpaceICCBased

  Represents a color space described by an ICC profile.
- class IDOMColorSpaceIndexed

  This color space is analogous to the PostScript/PDF Indexed color space.
- class IDOMDeviceNColorant

  This class enables the specification of colorant information for PDF style NChannel variants of DeviceN color spaces.
- class IDOMColorSpaceDeviceN

  This color space is analogous to the PostScript/PDF DeviceN/Separation color spaces.
- class IDOMColorSpaceLAB

  This color space is as described in section 4.5.4 of the PDF 1.7 Reference Manual.

Enumerations

- enum IDOMColorSpace::eColorSpaceType {
  IDOMColorSpace::eDeviceRGB , IDOMColorSpace::eDeviceGray , IDOMColorSpace::eDeviceCMYK , IDOMColorSpace::esRGB ,
  IDOMColorSpace::esGray , IDOMColorSpace::escRGB , IDOMColorSpace::eICCBased , IDOMColorSpace::eIndexed ,
  IDOMColorSpace::eDeviceN , IDOMColorSpace::eLAB , IDOMColorSpace::eDeviceCMY , **eNumColorSpaceTypes** = 11 }

  Color spaces type enumeration.

6.1.10.3.1 Detailed Description

A color space describes a range of colors appropriate to a given context.
A device color space describes the range of colors that a camera can see, a printer can print, or a monitor can display, and is tied to the device it describes.
Editing color spaces, on the other hand, such as Adobe RGB or sRGB, are device-independent, determining a color range you can work in. They are designed to be gray balanced, such that colors with equal amounts of red, green, and blue appear neutral, and allow you to edit images in a controlled, consistent manner. Editing spaces also are perceptually uniform; that is, changes to lightness, hue, or saturation are applied equally to all the colors in the image.

6.1.10.3.2 Enumeration Type Documentation

eColorSpaceType

`enum IDOMColorSpace::eColorSpaceType`

Color spaces type enumeration.

Enumerator

eDeviceRGB	Device RGB color space.
eDeviceGray	Device Gray color space.
eDeviceCMYK	Device CMYK color space.
esRGB	XPS sRGB color space with input range scaled to between 0 and 1.
esGray	Gray with sRGB gamma color space.
escRGB	XPS scRGB color space.
eICCBased	Color space with an ICC profile.
eIndexed	A color space analogous to the PostScript language/PDF Indexed color space,
eDeviceN	representing a mapping from an integral component value to components of an underlying color space. See section 4.8.4 of the PostScript Language Reference, 3rd Edition, and section 4.5.5 of the PDF Reference, version 1.7 A color space analogous to the PostScript language/PDF DeviceN/Separation color
eLAB	spaces as described in section 4.5.5 of the PDF 1.7 Reference Manual Lab color space as described in section 4.5.4 of the PDF 1.7 Reference Manual
eDeviceCMY	Device CMY color space to support PCL input. Internal use only.

6.1.10.4 Color profiles

A color profile provides an accurate description of the characteristics of a digital device or working color space.

Classes

- class IColorManager

  Public interface to the EDL color manager. There is only one instance of the color manager for each factory. It can be retrieved using the IEDLFactory::getSingleton method, or by using the get() static function.
- class IDOMICCProfile

  IDOMICCProfile interface.

6.1.10.4.1 Detailed Description

A color profile provides an accurate description of the characteristics of a digital device or working color space.

6.1.11 Validate

Validator classes. These provide interfaces allowing the validation of documents to published standards.

Classes

- class JawsMako::IPDFValidator

  A class for validating PDF documents against published PDF standards such as PDF/X.

6.1.11.1 Detailed Description

Validator classes. These provide interfaces allowing the validation of documents to published standards.

6.1.12 Text

JawsMako Text Conveniences.

Classes

- class JawsMako::IUnicodeHelper

  An interface into language specific unicode helpers.
- class JawsMako::ITextRun

  A run of text, containing unicode information, the position, transformation and bounds of the text.
- class JawsMako::IPageLayoutData

  Provides a representation of the analyzed page layout by organizing and allowing access to collections of IPage↩ LayoutNodes.
- class JawsMako::IPageLayoutNode

  Simple data type representing a part of an analyzed page.
- class JawsMako::IPageLayout

  Analyze the layout of a FixedPage, grouping together text deemed to be in horizontal and/or vertical blocks. Useful for text search and selection.
- class JawsMako::ITextSearch

  Perform text searching using the page information obtained from an IPageLayout.
- class JawsMako::ITextSelect

  Perform text selection using the page information obtained from an IPageLayout.

6.1.12.1 Detailed Description

JawsMako Text Conveniences.

Provides interfaces for extracting text information from a page.

6.1.13 Types

Common types and required headers for the JawsMako interface.

Typedefs

- typedef IEDLError **JawsMako::IError**

  An error type used for exceptions. Synonymous with IEDLError.
- typedef EDLString **JawsMako::String**

  A wide character string (UTF-16 on Windows, UTF-32 on all other platforms)
- typedef EDLSysString **JawsMako::U8String**

  A UTF-8 String.
- typedef EDLSysString **JawsMako::RawString**

  A raw, 8 bit string. Encoding depends on context.
- typedef EDLU16String **JawsMako::U16String**

  An explicit UTF-16 string, regardless of platform.
- typedef EDLU32String **JawsMako::U32String**

  An explicit UTF-32 string, regardless of platform.

6.1.13.1 Detailed Description

Common types and required headers for the JawsMako interface.

6.2 Separator

Separator Features.

Classes

- class JawsMako::ISeparator

  An instance of the JawsMako Separator class.

6.2.1 Detailed Description

Separator Features.

This header describes the Separator interface. Instances of this type provide a method for separating page content into pages created in monochrome.

7 Class Documentation

7.1 BoxTmpl< PointType > Class Template Reference

Template for a PDF-style box. Similar to a rectangle but specified using a left, bottom, right and top coordinate.

`#include <edlgeom.h>`

7.1.1 Detailed Description

template<typename PointType>
class BoxTmpl< PointType >

Template for a PDF-style box. Similar to a rectangle but specified using a left, bottom, right and top coordinate.

The documentation for this class was generated from the following file:

- edlgeom.h

7.2 JawsMako::CAnnotationBorder Class Reference

A class representing an annotation's border (described in the PDF Specification as BorderStyle). The meaning of a border style depends on the annotation type and not all annotation types will support all attributes of this class, and neither will all PDF versions support all attributes. Please refer to the PDF 1.7 specification for the required styles. JawsMako will only store the attributes that are valid for the given type, but will not signal errors for this case.

`#include <interactive.h>`

Public Types

- enum eBorderType {
  eBTSolid , eBTDashed , eBTBeveled , eBTInset ,
  eBTUnderline }

  Types of border.

Public Member Functions

- CAnnotationBorder (float inWidth=1.0f/72.0f ∗96.0f, const eBorderType &inType=eBTSolid, const CEDLVector< float > &inDash=CEDLVector< float >())

  Constructor.

Public Attributes

- float width
- eBorderType **type**

  The type of border.
- CEDLVector< float > dash

7.2.1 Detailed Description

A class representing an annotation's border (described in the PDF Specification as BorderStyle). The meaning of a border style depends on the annotation type and not all annotation types will support all attributes of this class, and neither will all PDF versions support all attributes. Please refer to the PDF 1.7 specification for the required styles. JawsMako will only store the attributes that are valid for the given type, but will not signal errors for this case.

7.2.2 Member Enumeration Documentation

eBorderType

```
enum JawsMako::CAnnotationBorder::eBorderType
```

Types of border.

Enumerator

eBTSolid	A solid rectangle surrounding the annotation.
eBTDashed	A dashed rectangle surrounding the annotation.
eBTBeveled	A simulated embossed rectangle that appears to be raised above the surface of the page.
eBTInset	A simulated engraved rectangle that appears to be recessed below the surface of the page.
eBTUnderline	A single line along the bottom of the annotation rectangle.

7.2.3 Constructor & Destructor Documentation

CAnnotationBorder()

```
JawsMako::CAnnotationBorder::CAnnotationBorder (
            float inWidth = 1.0f / 72.0f * 96.0f,
            const eBorderType & inType = eBTSolid,
            const CEDLVector< float > & inDash = CEDLVector<float>() )  [inline]
```

Constructor.

Parameters

inWidth	Border width. Default width is 1 PDF unit, or 1/72nd of an inch
inType	Border type. Default is eBTSolid
inDash	A vector of floats describing a eBTDashed dash pattern

7.2.4 Member Data Documentation

dash

```
CEDLVector<float> JawsMako::CAnnotationBorder::dash
```

If the border type is eBTDashed, this is the dash pattern, expressed in standard JawsMako units (96ths of an inch)

width

```
float JawsMako::CAnnotationBorder::width
```

Width of the border in standard JawsMako units (96ths of an inch); a width of zero indicates that the border is not drawn.

The documentation for this class was generated from the following file:

- interactive.h

7.3 JawsMako::IXAMLGenerator::CAnnotationXAML Class Reference

Class for receiving XAML generated for annotation appearances in a bulk fashion.

```
#include <xamlgenerator.h>
```

Public Attributes

- IAnnotationPtr **annotation**

  The annotation for which the XAML was generated.
- IAnnotationAppearancePtr **appearance**

  The appearance for which the XAML was generated.
- IRAInputStreamPtr **stream**

  The generated XAML stream.

7.3.1 Detailed Description

Class for receiving XAML generated for annotation appearances in a bulk fashion.

The documentation for this class was generated from the following file:

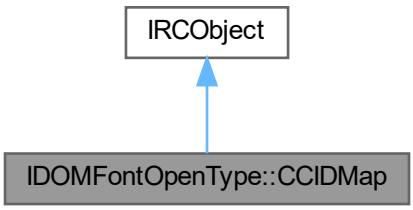
- xamlgenerator.h

7.4 IDOMFontOpenType::CCIDMap Class Reference

For TrueType-based CIDFonts, the mapping from CIDs to GlyphIds for non-identity orderings.

```
#include <idomfont.h>
```

Inheritance diagram for IDOMFontOpenType::CCIDMap:



Public Attributes

- EDLRawString registry
- EDLRawString ordering
- int32 supplement
- CEDLVector< uint16 > mapping

Additional Inherited Members

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.4.1 Detailed Description

For TrueType-based CIDFonts, the mapping from CIDs to GlyphIds for non-identity orderings.

7.4.2 Member Data Documentation

mapping

```
CEDLVector<uint16> IDOMFontOpenType::CCIDMap::mapping
```

The mapping from CID to GlyphID for all CIDs reachable in the font

ordering

```
EDLRawString IDOMFontOpenType::CCIDMap::ordering
```

The name of the ordering, eg Japan1, Korea1, CNS1, GB1, etc

registry

```
EDLRawString IDOMFontOpenType::CCIDMap::registry
```

The registry in which this CID ordering is registered

supplement

```
int32 IDOMFontOpenType::CCIDMap::supplement
```

The supplement number of the character collection

The documentation for this class was generated from the following file:

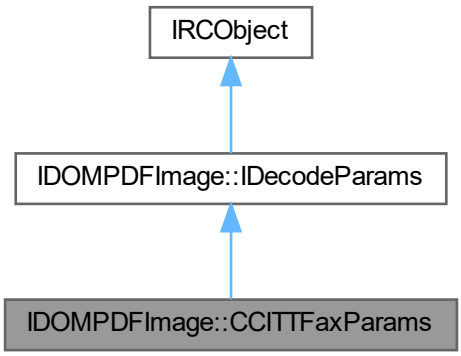
- idomfont.h

7.5 IDOMPDFImage::CCITTFaxParams Class Reference

Class to hold filter parameters for CCITTFax-compressed image data. Please see the PDF specification for the meaning of these parameters.

```
#include <idomimageresource.h>
```

Inheritance diagram for IDOMPDFImage::CCITTFaxParams:



Additional Inherited Members

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()
  Virtual destructor.

7.5.1 Detailed Description

Class to hold filter parameters for CCITTFax-compressed image data. Please see the PDF specification for the meaning of these parameters.

The documentation for this class was generated from the following file:

- idomimageresource.h

7.6 CClassID Class Reference

An object to represent a 128-bit globally unique ID.

`#include <objclassid.h>`

Public Member Functions

- **CClassID** ()
  Constructor.
- CClassID (const EDLSysString &str)
  Converts hexadecimal representation of a CClassID to a CClassID.
- CClassID (const CClassID &other)
  Copy to another CClassID.
- void operator= (const CClassID &other)
  operator =
- CClassID (uint32 dw0, uint32 dw1, uint32 dw2, uint32 dw3)
  Construct CClassID from 4 x uint32.
- bool equal (const CClassID &id) const
  Compare to another CClassID.

7.6.1 Detailed Description

An object to represent a 128-bit globally unique ID.

7.6.2 Constructor & Destructor Documentation

CClassID() [1/3]

```
CClassID::CClassID (
            const EDLSysString & str )  [inline]
```

Converts hexadecimal representation of a CClassID to a CClassID.

Parameters

str	String containing the hexadecimal value

CClassID() [2/3]

```
CClassID::CClassID (
            const CClassID & other )  [inline]
```

Copy to another CClassID.

Parameters

other	CClassID to copy to

CClassID() [3/3]

```
CClassID::CClassID (
            uint32 dw0,
            uint32 dw1,
            uint32 dw2,
            uint32 dw3 )  [inline]
```

Construct CClassID from 4 x uint32.

Parameters

dw0	First unsigned, 32-bit integer
dw1	Second unsigned, 32-bit integer
dw2	Third unsigned, 32-bit integer
dw3	Fourth unsigned, 32-bit integer

7.6.3 Member Function Documentation

equal()

```
bool CClassID::equal (
            const CClassID & id ) const  [inline]
```

Compare to another CClassID.

Parameters

id	CClassID with which to compare

Returns

bool True if equal, false if not

operator=()

```
void CClassID::operator= (
            const CClassID & other )  [inline]
```

operator =

Parameters

other	CClassID

The documentation for this class was generated from the following file:

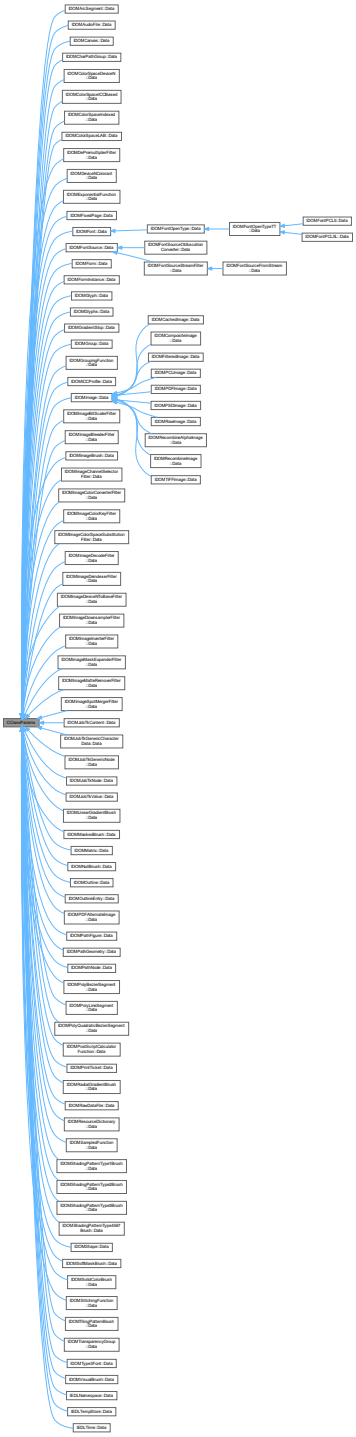
- objclassid.h

7.7 CClassParams Class Reference

EDL Object Interface.

```
#include <iedlobject.h>
```

Inheritance diagram for CClassParams:



7.7.1 Detailed Description

EDL Object Interface.

When an EDL object is created via a class factory the created object can be passed a collection of initialization parameters. This collection is passed into the EDL class factory object creation method as a pointer to a sub-class of a CClassParams

The documentation for this class was generated from the following file:

- iedlobject.h

7.8 JawsMako::IJawsRenderer::CColorSpotHalftone Class Reference

Description of spot halftones, using Jaws's default spot function. Used for color halftoned rendering. Analogous to a PostScript Type 2 Halftone; please refer to section 7.4.6 of the PostScript language reference manual. When rendering to RGB, there must be three angles. (Red, Green and Blue respectively). For CMYK output, there must be four (Cyan, Magenta, Yellow and Black respectively).

`#include <jawsmako.h>`

Inherits JawsMako::IJawsRenderer::IHalftone.

Public Attributes

- float **frequency**

  The halftone frequency in LPI to be used.
- bool **useFullResolutionForFlattening**

  Normally, the resolution at which flattening for transparent.
- CEDLVector< float > **angles**

  The vector of angles to use.

7.8.1 Detailed Description

Description of spot halftones, using Jaws's default spot function. Used for color halftoned rendering. Analogous to a PostScript Type 2 Halftone; please refer to section 7.4.6 of the PostScript language reference manual. When rendering to RGB, there must be three angles. (Red, Green and Blue respectively). For CMYK output, there must be four (Cyan, Magenta, Yellow and Black respectively).

Typically for CMYK rendering, the spot angles would be:

- Cyan 15 degrees
- Magenta 75 degrees
- Yellow 0 degrees
- Black 45 degrees

The documentation for this class was generated from the following file:

- jawsmako.h

7.9 JawsMako::IJawsRenderer::CColorThresholdArrayHalftone Class Reference

As per CThresholdArrayHalftone, but for use with color rendering. There is a single width and height, but a threshold array for each color being rendering. One threshold array must be specified for each color being rendered. When rendering to RGB, there must be three thresholds (Red, Green and Blue respectively). For CMYK output, there must be four thresholds (Cyan, Magenta, Yellow and Black respectively).

`#include <jawsmako.h>`

Inherits JawsMako::IJawsRenderer::IHalftone.

Public Attributes

- uint32 **width**

  The width of the halftone cell in pixels.
- uint32 **height**

  the height of the halftone cell in pixels
- CEDLVector< CEDLVector< uint8 > > **thresholdArrays**

  The threshold arrays. Must be width x height bytes in size, and no larger than 65535 bytes.

7.9.1 Detailed Description

As per CThresholdArrayHalftone, but for use with color rendering. There is a single width and height, but a threshold array for each color being rendering. One threshold array must be specified for each color being rendered. When rendering to RGB, there must be three thresholds (Red, Green and Blue respectively). For CMYK output, there must be four thresholds (Cyan, Magenta, Yellow and Black respectively).

The documentation for this class was generated from the following file:

- jawsmako.h

7.10 JawsMako::IPDFValidator::CContentError Class Reference

A class describing validation errors present in a particular location on a page.

`#include <validate.h>`

7.10.1 Detailed Description

A class describing validation errors present in a particular location on a page.

Please note that the location may be an empty rectangle if the object in question does not mark, is entirely clipped out, or is a modifier for a composite object (such as when a colorspace is used for an isolated transparency group).

The documentation for this class was generated from the following file:

- validate.h

7.11 JawsMako::IJawsRenderer::CEDSHalftone Class Reference

A halftone representing an error diffusion screen. Allows the production of results containing a variable number of gray levels per channel using a range of error diffusion screens.

`#include <jawsmako.h>`

Inherits JawsMako::IJawsRenderer::IHalftone.

Public Attributes

- uint8 **dropSizes**

  The number of distinct drop sizes to produce (not including white). Please refer to the class description for details.
- uint8 **rows**

  The number of rows in the error weighting matrix - 5 maximum.
- uint8 **columns**

  The number of columns in the error weighting matrix - 4 maximum.
- uint8 **pixelColumn**

  The pixel column.
- uint32 **denominator**

  The denominator parameter, set to the sum of the weights.
- uint8 **weights** [20]

  The weight matrix; maximum of 20 entries (5 x 4)
- bool **useSerpentine**

  If true, the algorithm reverses scanning direction with each alternate scanline to reduce pattern artifacts.
- float **perturbation**

  The weight of random perturbation to appy to each pre-screened pixel which may reduce pattern artifacts.

7.11.1 Detailed Description

A halftone representing an error diffusion screen. Allows the production of results containing a variable number of gray levels per channel using a range of error diffusion screens.

May be used for both monochrome and color rendering with renderScreened() and renderScreenedToFrameBuffers().

The dropSizes parameter defines the number of gray levels that will be produced:

- 1 specifies a monochrome result
- 2 specifies two different drop sizes, producing no ink, 50% ink, and 100% ink
- 3 specifies three different drop sizes, producing no ink, 33% ink, 66% ink and 100% ink and so forth up to a maximum dropSizes parameter of 8.

For a dropSizes parameter of 1, the renderScreened() and renderScreenedToFrameBuffers() will produce 1bpc output. For other dropSizes, these APIs will currently produce 4bpc results which may be post-processed as desired.

The default is a simple Atkinson monochrome error diffusion screen

The EDS Workbench application (see http://documentation.globalgraphics.com) can be used to explore the possibilities and generate parameters for this class.

The documentation for this class was generated from the following file:

- jawsmako.h

7.12 JawsMako::CFieldOption Class Reference

Class to store form field option array entries.

```
#include <interactive.h>
```

7.12.1 Detailed Description

Class to store form field option array entries.

The documentation for this class was generated from the following file:

- interactive.h

7.13 JawsMako::IJawsRenderer::CFrameBufferInfo Class Reference

Description of a frame buffer for use with renderSeparationsToFrameBuffers.

`#include <jawsmako.h>`

Public Attributes

- void * **buffer**

  Pointer to the first pixel in the frame buffer.
- int32 **rowStride**

  The distance, in bytes, from one scanline in the frame buffer to the next. May be negative.
- int32 pixelStride

7.13.1 Detailed Description

Description of a frame buffer for use with renderSeparationsToFrameBuffers.

7.13.2 Member Data Documentation

pixelStride

`int32 JawsMako::IJawsRenderer::CFrameBufferInfo::pixelStride`

The distance, in bytes, from one sample to the next. May be negative. If zero, the pixels are assumed to be packed. That is, the next sample for the separation is placed next to the last.

The documentation for this class was generated from the following file:

- jawsmako.h

7.14 JawsMako::IPDFValidator::CGeneralError Class Reference

A class describing validation errors found in a PDF that are not tied to a location on a page.

`#include <validate.h>`

7.14.1 Detailed Description

A class describing validation errors found in a PDF that are not tied to a location on a page.

The documentation for this class was generated from the following file:

- validate.h

7.15 CGlyphsCluster Class Reference

A single cluster generated from the parallel Indices and Unicode strings present in an IDOMGlyphs node. This represents the smallest logical unit of text. Here, a single cluster may represent:

`#include <iglyphsclusters.h>`

Public Member Functions

- bool similar (const CGlyphsCluster &other, float eps) const

  Check to see if this cluster is similar to another with regard to metrics. Non-metrics information (such as glyph ids and unicode) must be equivalent for two clusters to be considered similar.

Public Attributes

- CEDLVector< uint32, 5 > **unicode**

  The unicode information for the cluster as UTF-32, if present.
- CIndicesGlyphList **glyphs**

  The glyph information, if present.

7.15.1 Detailed Description

A single cluster generated from the parallel Indices and Unicode strings present in an IDOMGlyphs node. This represents the smallest logical unit of text. Here, a single cluster may represent:

- a 1:1 mapping between a unicode value and a glyph
- a N:1 mapping between a number of unicode values and a glyph, such as where unicode combining characters are used, but represented with a single glyph
- a 1:N mapping where a single unicode value is represented by multiple glyphs, such as where composite characters are used
- a N:N mapping where multiple unicode characters are represented with multiple glyphs for complex language situations
- cases where a unicode value is present but no glyph is specified (instead a glyph is selected from the font via the unicode character map)
- cases where a glyph is specified but with no unicode information

7.15.2 Member Function Documentation

similar()

```
bool CGlyphsCluster::similar (
            const CGlyphsCluster & other,
            float eps ) const  [inline]
```

Check to see if this cluster is similar to another with regard to metrics. Non-metrics information (such as glyph ids and unicode) must be equivalent for two clusters to be considered similar.

Parameters

other	The cluster to compare.
eps	The epsilon limit for metrics similarity, in hundredths of an em.

Returns

bool True if similar, false otherwise.

The documentation for this class was generated from the following file:

- iglyphsclusters.h

7.16 CGlyphsClusters Class Reference

A single cluster generated from the parallel Indices and Unicode strings present in an IDOMGlyphs node. This represents the smallest logical unit of text. Here, a single cluster may represent:

```
#include <iglyphsclusters.h>
```

7.16.1 Detailed Description

A single cluster generated from the parallel Indices and Unicode strings present in an IDOMGlyphs node. This represents the smallest logical unit of text. Here, a single cluster may represent:

- a 1:1 mapping between a unicode value and a glyph
- a N:1 mapping between a number of unicode values and a glyph, such as where unicode combining characters are used, but represented with a single glyph
- a 1:N mapping where a single unicode value is represented by multiple glyphs, such as where composite characters are used
- a N:N mapping where multiple unicode characters are represented with multiple glyphs for complex language situations
- cases where a unicode value is present but no glyph is specified (instead a glyph is selected from the font via the unicode character map)
- cases where a glyph is specified but with no unicode information

The documentation for this class was generated from the following file:

- iglyphsclusters.h

7.17 CIndicesGlyph Class Reference

Utility code allowing the simpler manipulation of glyph glusters represented by the UnicodeString and Indices entries of an IDOMGlyphs object.

```
#include <iglyphsclusters.h>
```

Public Member Functions

- bool similar (const CIndicesGlyph &other, float eps) const

  Check to see if this indices glyph entry is similar to another with regard to metrics. Non-metrics information (such as glyph ids) must be equivalent for two indices glyphs to be considered similar.

Public Attributes

- bool **hasGlyphId**

  True if the glyphId entry is valid.
- uint16 **glyphId**

  The glyph ID of the glyph, if hasGlyphId is true.
- bool **hasAdvance**

  True if the advance entry is valid.
- float **advance**

  The advance, in hundredths of an em, if hasAdvance is true.
- bool **hasUOffset**

  True if the uOffset entry is valid.
- float **uOffset**

  The u offset of the glyph, in hundredths of an em, if hasUOffset is true.
- bool **hasVOffset**

  True if the vOffset entry is valid.
- float **vOffset**

  The v offset of the glyph, in hundredths of an em, if hasVOffset is true.

7.17.1 Detailed Description

Utility code allowing the simpler manipulation of glyph glusters represented by the UnicodeString and Indices entries of an IDOMGlyphs object.

A glyph unit, populated from an entry in an Indices string. All entries are optional, and their presence is flagged with hasXxxxx entries.

7.17.2 Member Function Documentation

similar()

```
bool CIndicesGlyph::similar (
            const CIndicesGlyph & other,
            float eps ) const  [inline]
```

Check to see if this indices glyph entry is similar to another with regard to metrics. Non-metrics information (such as glyph ids) must be equivalent for two indices glyphs to be considered similar.

Parameters

other	The indices glyph to compare.
eps	The epsilon limit for metrics similarity, in hundredths of an em.

Returns

bool True if similar, false otherwise.

The documentation for this class was generated from the following file:

- iglyphsclusters.h

7.18 JawsMako::ILayoutFont::CLayoutFontItem Class Reference

An association of a font and it's font index.

`#include <layout.h>`

7.18.1 Detailed Description

An association of a font and it's font index.

The documentation for this class was generated from the following file:

- layout.h

7.19 IDOMShadingPatternType4567Brush::CMeshEntry Class Reference

A entry in the shading pattern's mesh. The interpretation of each entry depends on the shading type, and potentially on per-entry flags. Please see the PDF specification for details.

`#include <idombrush.h>`

Public Attributes

- uint8 **flag**

  A flag for the mesh entry. Used for Type 4, 6, and 7 shading patterns.
- CEDLVector< FPoint > points
- CEDLVector< CEDLVector< float > > colors

7.19.1 Detailed Description

A entry in the shading pattern's mesh. The interpretation of each entry depends on the shading type, and potentially on per-entry flags. Please see the PDF specification for details.

7.19.2 Member Data Documentation

colors

`CEDLVector<CEDLVector<float> > IDOMShadingPatternType4567Brush::CMeshEntry::colors`

A variable number of colors. The interpretation of these colors depends on the shading type and (for all but Type 4 shades) the flag entry. For shading patterns that use functions, each color will be a single component that is then passed through the function to obtain a final color value.

points

```
CEDLVector<FPoint> IDOMShadingPatternType4567Brush::CMeshEntry::points
```

A variable number of points. The interpretation of these points depends on the shading type and (for all but Type 4 shades) the flag entry.

The documentation for this class was generated from the following file:

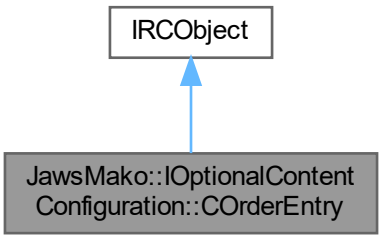
- idombrush.h

7.20 JawsMako::IOptionalContentConfiguration::COrderEntry Class Reference

Class for presenting the order that groups should be displayed in a user interface. May be arranged in a tree.

```
#include <optionalcontent.h>
```

Inheritance diagram for JawsMako::IOptionalContentConfiguration::COrderEntry:



Public Member Functions

- COrderEntryPtr **clone** ()

  Create a deep copy.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Attributes

- IOptionalContentGroupReferencePtr **groupRef**

  Only valid if isGroup is true.
- U8String **name**

  May be an empty string. Only valid if isGroup is false.
- COrderEntryVect **children**

  Only valid if isGroup is false.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.20.1 Detailed Description

Class for presenting the order that groups should be displayed in a user interface. May be arranged in a tree.

The documentation for this class was generated from the following file:

- optionalcontent.h

7.21 JawsMako::IPDFValidator::CPageErrors Class Reference

A collection of all the errors associated with a page.

`#include <validate.h>`

7.21.1 Detailed Description

A collection of all the errors associated with a page.

The documentation for this class was generated from the following file:

- validate.h

7.22 JawsMako::CPDFFarReference Class Reference

A simple concrete class representing indirect reference data stored in a remote context, such as (for example) from a different PDF document.

`#include <pdfobjects.h>`

Public Member Functions

- CPDFFarReference (DOMid docId, DOMid allocatorId, int32 objectNum, int32 generation)

  Construct the reference data from remote document, allocator, object number and generator.
- DOMid getDocumentId () const

  Obtain the source document ID for this document.
- DOMid getAllocatorId () const

  Obtain the id of the entity that allocated the object. If the same as the document id, the referenced object is present in the original PDF document with that Id. Otherwise it has been allocated anew.
- int32 getObjectNum () const

  Obtain the object number of this reference data.
- int32 getGeneration () const

  Obtain the generation number of this reference data.

7.22.1 Detailed Description

A simple concrete class representing indirect reference data stored in a remote context, such as (for example) from a different PDF document.

7.22.2 Constructor & Destructor Documentation

CPDFFarReference()

```
JawsMako::CPDFFarReference::CPDFFarReference (
            DOMid docId,
            DOMid allocatorId,
            int32 objectNum,
            int32 generation )  [inline]
```

Construct the reference data from remote document, allocator, object number and generator.

Parameters

docId	The id of the source document for the referred object
allocatorId	A unique allocator ID used to differentiate between updated objects and unedited objects from the source document.
objectNum	The object number to use
generation	The generation number to use

7.22.3 Member Function Documentation

getAllocatorId()

```
DOMid JawsMako::CPDFFarReference::getAllocatorId ( ) const  [inline]
```

Obtain the id of the entity that allocated the object. If the same as the document id, the referenced object is present in the original PDF document with that Id. Otherwise it has been allocated anew.

Returns

DOMid the allocator ID.

getDocumentId()

```
DOMid JawsMako::CPDFFarReference::getDocumentId ( ) const  [inline]
```

Obtain the source document ID for this document.

Returns

DOMid the source document ID.

getGeneration()

```
int32 JawsMako::CPDFFarReference::getGeneration ( ) const  [inline]
```

Obtain the generation number of this reference data.

Returns

int32 The generation number.

getObjectNum()

```
int32 JawsMako::CPDFFarReference::getObjectNum ( ) const  [inline]
```

Obtain the object number of this reference data.

Returns

int32 The object number.

The documentation for this class was generated from the following file:

- pdfobjects.h

7.23 JawsMako::IPDFInput::CPdfFontInfo Class Reference

Information about a font in a PDF file, obtained by scanning the PDF font structures.

```
#include <pdfinput.h>
```

Public Attributes

- bool **embedded**

  True if the PDF provides an embedded font stream for this font.
- bool **subset**

  True if the PDF indicates the font is a subset (that is, the font name has a subset tag)
- RawString **fontType**

  The PDF Font Type as a string.
- RawString **fontName**

  The name of the font, according to the font's FontDescriptor entry, if present.
- RawString **baseFontName**

  The "base" (PostScript) name of the font according to the Font dictionary.
- int32 **objectNumber**

  If the font dictionary is an indirect object, this is the object number (negative otherwise)
- int32 **generation**

  If the font dictionary is an indirect object, this is the generation (negative otherwise)
- int32 **subFontObjectNumber**

  If the font dictionary is a CIDFont and the subfont is an indirect object, this is the object number (negative otherwise)
- int32 **subFontGeneration**

  If the font dictionary is a CIDFont and the subfont is an indirect object, this is the object number (negative otherwise)
- CUInt32Vect **referencedPages**

  A vector of 0-indexed page numbers on which this font is referenced.
- IDOMFontPtr font

7.23.1 Detailed Description

Information about a font in a PDF file, obtained by scanning the PDF font structures.

7.23.2 Member Data Documentation

font

```
IDOMFontPtr JawsMako::IPDFInput::CPdfFontInfo::font
```

Pointer to an IDOMFont generated from the PDF font. NULL if scanPdfForFonts() was invoked with the domFont parameter set to false.

The documentation for this class was generated from the following file:

- pdfinput.h

7.24 JawsMako::CPDFReference Class Reference

A simple concrete class representing indirect reference data.

```
#include <pdfobjects.h>
```

Public Member Functions

- CPDFReference (int32 objectNum, int32 generation)

  Construct the reference data from an object number and generation.
- int32 getObjectNum () const

  Obtain the object number of this reference data.
- int32 getGeneration () const

  Obtain the generation number of this reference data.

7.24.1 Detailed Description

A simple concrete class representing indirect reference data.

7.24.2 Constructor & Destructor Documentation

CPDFReference()

```
JawsMako::CPDFReference::CPDFReference (
            int32 objectNum,
            int32 generation )  [inline]
```

Construct the reference data from an object number and generation.

Parameters

objectNum	The object number to use
generation	The generation number to use

7.24.3 Member Function Documentation

getGeneration()

```
int32 JawsMako::CPDFReference::getGeneration ( ) const  [inline]
```

Obtain the generation number of this reference data.

Returns

int32 The generation number.

getObjectNum()

```
int32 JawsMako::CPDFReference::getObjectNum ( ) const  [inline]
```

Obtain the object number of this reference data.

Returns

int32 The object number.

The documentation for this class was generated from the following file:

- pdfobjects.h

7.25 JawsMako::IPDFInput::CPdfScannedInk Class Reference

Basic information about an ink used in a PDF file, obtained by scanning the PDF page tree.

`#include <pdfinput.h>`

Public Attributes

- RawString **inkName**

  The name of the ink.
- CUInt32Vect pageNums

7.25.1 Detailed Description

Basic information about an ink used in a PDF file, obtained by scanning the PDF page tree.

The returned information does not provide details about the ink itself; instead, this information may used to scan the appropriate page in the PDF where the ink is encountered using IRendererTransform::findInks()

7.25.2 Member Data Documentation

pageNums

`CUInt32Vect JawsMako::IPDFInput::CPdfScannedInk::pageNums`

The page numbers where this ink is encountered (where 0 indicates the first page)

The documentation for this class was generated from the following file:

- pdfinput.h

7.26 JawsMako::IPJLParser::CPjlAttributeValue Class Reference

A captured PJL attribute.

`#include <pjl.h>`

Public Attributes

- RawString modifier
- RawString key
- RawString value

7.26.1 Detailed Description

A captured PJL attribute.

7.26.2 Member Data Documentation

key

```
RawString JawsMako::IPJLParser::CPjlAttributeValue::key
```

The parsed PJL key, if present. Always presented as upper case.

modifier

```
RawString JawsMako::IPJLParser::CPjlAttributeValue::modifier
```

The parsed PJL modifier, if present. Always presented as upper case without spaces.

value

```
RawString JawsMako::IPJLParser::CPjlAttributeValue::value
```

The PJL value, if present. Unmodified.

The documentation for this class was generated from the following file:

- pjl.h

7.27 JawsMako::CQuadPoint Class Reference

A representation of a PDF Quadpoint, in DOM coordinates.

```
#include <interactive.h>
```

7.27.1 Detailed Description

A representation of a PDF Quadpoint, in DOM coordinates.

The documentation for this class was generated from the following file:

- interactive.h

7.28 JawsMako::CRectInset Class Reference

A class which specifies an inset from a rectangle.

```
#include <interactive.h>
```

7.28.1 Detailed Description

A class which specifies an inset from a rectangle.

The documentation for this class was generated from the following file:

- interactive.h

7.29 JawsMako::ISVGGenerator::CResourceEntry Class Reference

Resource entry.

`#include <svggenerator.h>`

7.29.1 Detailed Description

Resource entry.

The documentation for this class was generated from the following file:

- svggenerator.h

7.30 JawsMako::IXAMLGenerator::CResourceEntry Class Reference

Resource entry.

`#include <xamlgenerator.h>`

7.30.1 Detailed Description

Resource entry.

The documentation for this class was generated from the following file:

- xamlgenerator.h

7.31 IDOMShape::CShapeDetails Class Reference

Provides a view into the regions that comprise the shape. A region consists of a series of spans, each representing a series of rectangles that share the same y span.

`#include <idomshape.h>`

Classes

- struct CSpan

  Representation of a series of rectangles sharing the same y span.

Public Attributes

- uint32 **numSpans**

  The number of spans. The last span is used solely to indicate the y endpoint of the penultimate span.
- CSpan * **spans**

  The spans as an array. The memory is owned by the IDOMShape. Do not free.

7.31.1 Detailed Description

Provides a view into the regions that comprise the shape. A region consists of a series of spans, each representing a series of rectangles that share the same y span.

The documentation for this class was generated from the following file:

- idomshape.h

7.32 IDOMShape::CShapeDetails::CSpan Struct Reference

Representation of a series of rectangles sharing the same y span.

```
#include <idomshape.h>
```

Public Attributes

- int32 **y**

  The y point of the top of this span (inclusive). The ending y coordinate of the span is the y entry in the next span.
- int32 **length**

  The number of x points in xVals. Always a multiple of two.
- int32 * **xVals**

  Points representing the start (inclusive) and end (exclusive) in x of the rectangles comprising the span. The memory is owned by the IDOMShape. Do not free.

7.32.1 Detailed Description

Representation of a series of rectangles sharing the same y span.

The documentation for this struct was generated from the following file:

- idomshape.h

7.33 JawsMako::IJawsRenderer::CSpotHalftone Class Reference

Description of a simple spot halftone, at 45 degrees, using Jaws's default spot function. Used for monochrome rendering.

`#include <jawsmako.h>`

Inherits JawsMako::IJawsRenderer::IHalftone.

Public Attributes

- float **frequency**

  The frequency of the spot function, in lines per inch.
- bool **useFullResolutionForFlattening**

  Normally, the resolution at which flattening for transparent.

7.33.1 Detailed Description

Description of a simple spot halftone, at 45 degrees, using Jaws's default spot function. Used for monochrome rendering.

The documentation for this class was generated from the following file:

- jawsmako.h

7.34 JawsMako::CTemporaryStoreParameters Class Reference

Allows the temporary storage parameters to be optionally overridden.

`#include <types.h>`

Public Member Functions

- CTemporaryStoreParameters (uint64 memoryLimit=0, uint64 diskLimit=0, uint32 blockSize=0)

  Initialise the temporary store parameter block.

7.34.1 Detailed Description

Allows the temporary storage parameters to be optionally overridden.

7.34.2 Constructor & Destructor Documentation

CTemporaryStoreParameters()

```
JawsMako::CTemporaryStoreParameters::CTemporaryStoreParameters (
            uint64 memoryLimit = 0,
            uint64 diskLimit = 0,
            uint32 blockSize = 0 )  [inline]
```

Initialise the temporary store parameter block.

Parameters

memoryLimit	Set the limit, in bytes, on data stored in memory by the temporary store (IEDLTempStore). Pass zero to use the default for the current platform. The maximum is 32GB, though of course on 32 bit systems, it will need to be less than 4GB. The value given will be rounded down to a multiple of the block size.
diskLimit	Set the limit, in bytes, on data stored on disk by the temporary store (IEDLTempStore). Pass zero to use the default for the current platform. The maximum is currently 256GB, however for Android this is currently limited to 2GB. The value given will be rounded down to a multiple of the block size.
blockSize	Set the block size to use. The default is currently 4096 bytes. Each file in the temporary store will occupy a multiple of the block size. Note that currently, the total number of blocks must be less than 1. That is, (diskLimit + memoryLimit) / blockSize must be less than 2147483647.

The documentation for this class was generated from the following file:

- types.h

7.35 JawsMako::IJawsRenderer::CThresholdArrayHalftone Class Reference

Description of a Type 3 8-bit threshold array halftone for use with monochrome rendering. Please refer to section 7.4.5 of the PostScript language reference manual, 3rd edition.

`#include <jawsmako.h>`

Inherits JawsMako::IJawsRenderer::IHalftone.

Public Attributes

- uint32 **width**

  The width of the halftone cell in pixels.
- uint32 **height**

  the height of the halftone cell in pixels
- CEDLVector< uint8 > **thresholdArray**

  The threshold array. Must be width x height bytes in size, and no larger than 65535 bytes.

7.35.1 Detailed Description

Description of a Type 3 8-bit threshold array halftone for use with monochrome rendering. Please refer to section 7.4.5 of the PostScript language reference manual, 3rd edition.

The documentation for this class was generated from the following file:

- jawsmako.h

7.36 JawsMako::IJawsRenderer::CThresholdHalftone Class Reference

A halftone representing a simple threshold. Used for monochrome rendering.

`#include <jawsmako.h>`

Inherits JawsMako::IJawsRenderer::IHalftone.

Public Attributes

- uint8 **threshold**
  Range is 8 bits, 1 to 254 inclusive.

7.36.1 Detailed Description

A halftone representing a simple threshold. Used for monochrome rendering.

The documentation for this class was generated from the following file:

- jawsmako.h

7.37 CTransformMatrix< TItem > Class Template Reference

Matrix class - special 3x2 matrix.

`#include <edlgeom.h>`

Public Types

- enum eOperationTypes { eDoesTranslate = 0x1 , eDoesScale = 0x2 , eDoesRotate = 0x4 , eIsComplex = 0x8 }
  Classification of operation type flags of the transform.

Public Member Functions

- **CTransformMatrix** ()
  Creates zero matrix.
- CTransformMatrix (TItem _xx, TItem _xy, TItem _yx, TItem _yy, TItem _dx, TItem _dy)
  Creates the matrix elementary.
- CTransformMatrix (const CTransformMatrix< TItem > &m)
  Copy constructor.
- CTransformMatrix & operator= (const CTransformMatrix< TItem > &m)
  Assignment operator.
- CTransformMatrix (const RectTmpl< TItem > &sourceRect, const RectTmpl< TItem > &destRect)
  Creates a matrix that transforms from one rectangle to another.
- void set (TItem _xx=1, TItem _xy=0, TItem _yx=0, TItem _yy=1, TItem _dx=0, TItem _dy=0)
  Create blank matrix.
- TItem xx () const

Returns xx component.

- TItem xy () const

  Returns xy component.

- TItem yx () const

  Returns yx component.

- TItem yy () const

  Returns yy component.

- TItem dx () const

  Returns dx component.

- TItem dy () const

  Returns dy component.

- void setXX (TItem x)

  Sets xx component.

- void setXY (TItem x)

  Sets xy component.

- void setYX (TItem x)

  Sets yx component.

- void setYY (TItem x)

  Sets yy component.

- void setDX (TItem x)

  Sets dx component.

- void setDY (TItem x)

  Sets dy component.

- bool equal (const CTransformMatrix< TItem > &matrix, bool ignoreDXDY=false) const

  Compare to another matrix.

- bool identity (bool ignoreDXDY=false) const

  Determine if identity matrix.

- CTransformMatrix< TItem > & preMul (const CTransformMatrix< TItem > &matrix)

  Premultiply by given matrix.

- CTransformMatrix< TItem > & postMul (const CTransformMatrix< TItem > &matrix)

  Postmultiply by given matrix.

- bool degenerate ()

  Check to see if the matrix is degenerate (0 scale)

- bool invert ()

  Invert the matrix.

- void transform (PointTmpl< TItem > &result, const PointTmpl< TItem > &point, bool ignoreDXDY=false) const

  Transform a point.

- PointTmpl< TItem > transform (const PointTmpl< TItem > &point, bool ignoreDXDY=false) const

  Transform a point.

- bool iTransform (PointTmpl< TItem > &result, const PointTmpl< TItem > &point, bool ignoreDXDY=false) const

  Transform a point by the inverse of the matrix.

- std::pair< bool, PointTmpl< TItem > > iTransform (const PointTmpl< TItem > &point, bool ignore↩ DXDY=false) const

  Transform a point by the inverse of the matrix.

- void rotate (double radians)

  Add a rotation, clockwise, in radians.

- void scale (TItem xscale, TItem yscale)

  Scale.

- void translate (TItem dx, TItem dy)

Translate.

- TItem determinant () const

  Find the determinant of the matrix.

- void transformRect (RectTmpl< TItem > &rect, bool ignoreDXDY=false) const

  Transform a rectangle.

- uint32 classify () const

  Classify the transform.

- void decompose (PointTmpl< TItem > &translate, double &rotationAngle, double &shearAngle, FPoint &scale) const

  Similar to the other form of decompose, but does the decomposition in a different, potentially more useful order. It's designed to more naturally extract rotation information when objects are scaled in a rotated context.

- void compose (const PointTmpl< TItem > &translateAmount, double rotationAngle, double shearAngle, const FPoint &scaleAmount)

  Undo the second form of decompose above. It starts with an identity matrix and applies the transforms in this order:

- template<typename AType >
  void asArray (AType *array) const

  Retrieve the matrix as an array of the given type.

7.37.1 Detailed Description

template<typename TItem>
class CTransformMatrix< TItem >

Matrix class - special 3x2 matrix.

7.37.2 Constructor & Destructor Documentation

CTransformMatrix() [1/3]

```
template<typename TItem >
CTransformMatrix< TItem >::CTransformMatrix (
            TItem _xx,
            TItem _xy,
            TItem _yx,
            TItem _yy,
            TItem _dx,
            TItem _dy )  [inline]
```

Creates the matrix elementary.

Parameters

_xx	xx component
_xy	xy component
_yx	yx component
_yy	yy component
_dx	dx component
_dy	dy component

CTransformMatrix() [2/3]

```
template<typename TItem >
CTransformMatrix< TItem >::CTransformMatrix (
            const CTransformMatrix< TItem > & m )  [inline]
```

Copy constructor.

Parameters

m	Transform matrix

CTransformMatrix() [3/3]

```
template<typename TItem >
CTransformMatrix< TItem >::CTransformMatrix (
            const RectTmpl< TItem > & sourceRect,
            const RectTmpl< TItem > & destRect )  [inline]
```

Creates a matrix that transforms from one rectangle to another.

Parameters

sourceRect	Source rectangle
destRect	Destination rectangle

7.37.3 Member Function Documentation

asArray()

```
template<typename TItem >
template<typename AType >
void CTransformMatrix< TItem >::asArray (
            AType * array ) const  [inline]
```

Retrieve the matrix as an array of the given type.

Parameters

array	Pointer to a six-entry array of the given type

classify()

```
template<typename TItem >
uint32 CTransformMatrix< TItem >::classify ( ) const  [inline]
```

Classify the transform.

Returns

uint32 Operation flags

compose()

```
template<typename TItem >
void CTransformMatrix< TItem >::compose (
             const PointTmpl< TItem > & translateAmount,
             double rotationAngle,
             double shearAngle,
             const FPoint & scaleAmount )  [inline]
```

Undo the second form of decompose above. It starts with an identity matrix and applies the transforms in this order:

- Translate
- Rotate
- Shear
- Scale

Parameters

translateAmount	The desired translation component in x and y.
rotationAngle	The desired rotation angle in radians.
shearAngle	The desired shear angle in radians.
scaleAmount	The desired scale in x and y.

decompose()

```
template<typename TItem >
void CTransformMatrix< TItem >::decompose (
             PointTmpl< TItem > & translate,
             double & rotationAngle,
             double & shearAngle,
             FPoint & scale ) const  [inline]
```

Similar to the other form of decompose, but does the decomposition in a different, potentially more useful order. It's designed to more naturally extract rotation information when objects are scaled in a rotated context.

Here, once decomposed, the original matrix (within the limitations of floating point roundoff and trig functions) can be reconstructed through successive premultiplies in the following order:

- Translate
- Rotate
- Shear
- Scale

See compose, which demonstrates how to perform these steps.

Parameters

translate	Reference to receive the translation component in x and y.
rotationAngle	Reference to receive the rotation angle in radians. Please note that the rotation direction is not the same as the other decompose method, and instead uses the same semantics as the rotate member function (clockwise).
shearAngle	Reference to receive the shear angle in radians.
scale	Reference to receive the scale component in x and y.

degenerate()

```
template<typename TItem >
bool CTransformMatrix< TItem >::degenerate ( )  [inline]
```

Check to see if the matrix is degenerate (0 scale)

Returns

bool True if matrix is degenerate, false otherwise.

determinant()

```
template<typename TItem >
TItem CTransformMatrix< TItem >::determinant ( ) const  [inline]
```

Find the determinant of the matrix.

Returns

TItem the determinant

dx()

```
template<typename TItem >
TItem CTransformMatrix< TItem >::dx ( ) const  [inline]
```

Returns dx component.

Returns

TItem dx component

dy()

```
template<typename TItem >
TItem CTransformMatrix< TItem >::dy ( ) const  [inline]
```

Returns dy component.

Returns

TItem dy component

equal()

```
template<typename TItem >
bool CTransformMatrix< TItem >::equal (
            const CTransformMatrix< TItem > & matrix,
            bool ignoreDXDY = false ) const  [inline]
```

Compare to another matrix.

Parameters

matrix	Matrix to compare current instance to
ignoreDXDY	Ignore DX & DY for this comparison

Returns

bool True if equal, false if not

identity()

```
template<typename TItem >
bool CTransformMatrix< TItem >::identity (
            bool ignoreDXDY = false ) const  [inline]
```

Determine if identity matrix.

Parameters

ignoreDXDY	Ignore DX & DY for this operation

Returns

bool True if identity matrix, false if not

invert()

```
template<typename TItem >
bool CTransformMatrix< TItem >::invert ( )  [inline]
```

Invert the matrix.

Returns

bool True if matrix was inverted, false if inversion was not possible

iTransform() [1/2]

```
template<typename TItem >
std::pair< bool, PointTmpl< TItem > > CTransformMatrix< TItem >::iTransform (
            const PointTmpl< TItem > & point,
            bool ignoreDXDY = false ) const  [inline]
```

Transform a point by the inverse of the matrix.

Parameters

point	The point
ignoreDXDY	Ignore DX & DY for this operation

Returns

std::pair<**bool**, PointTmpl<TItem> > A pair whose first element is true if the point was transformed while the second element is the resulting point

iTransform() [2/2]

```
template<typename TItem >
bool CTransformMatrix< TItem >::iTransform (
            PointTmpl< TItem > & result,
            const PointTmpl< TItem > & point,
            bool ignoreDXDY = false ) const  [inline]
```

Transform a point by the inverse of the matrix.

Parameters

result	PointTmpl<TItem> to accept result
point	The point
ignoreDXDY	Ignore DX & DY for this operation

Returns

bool True if point was transformed, false if transformation was not possible

operator=()

```
template<typename TItem >
CTransformMatrix & CTransformMatrix< TItem >::operator= (
            const CTransformMatrix< TItem > & m )  [inline]
```

Assignment operator.

Parameters

m	Transform matrix

postMul()

```
template<typename TItem >
CTransformMatrix< TItem > & CTransformMatrix< TItem >::postMul (
            const CTransformMatrix< TItem > & matrix )  [inline]
```

Postmultiply by given matrix.

Parameters

matrix	Matrix postmultiplier

Returns

CTransformMatrix<**TItem**> Resulting matrix

preMul()

```
template<typename TItem >
CTransformMatrix< TItem > & CTransformMatrix< TItem >::preMul (
            const CTransformMatrix< TItem > & matrix )  [inline]
```

Premultiply by given matrix.

Parameters

matrix	Matrix premultiplier

Returns

CTransformMatrix<**TItem**> Resulting matrix

rotate()

```
template<typename TItem >
void CTransformMatrix< TItem >::rotate (
            double radians )  [inline]
```

Add a rotation, clockwise, in radians.

Parameters

radians	Angle in radians

scale()

```
template<typename TItem >
void CTransformMatrix< TItem >::scale (
            TItem xscale,
            TItem yscale )  [inline]
```

Scale.

Parameters

xscale	X-axis scaling factor
yscale	Y-axis scaling factor

set()

```
template<typename TItem >
void CTransformMatrix< TItem >::set (
            TItem _xx = 1,
            TItem _xy = 0,
            TItem _yx = 0,
            TItem _yy = 1,
            TItem _dx = 0,
            TItem _dy = 0 )  [inline]
```

Create blank matrix.

Parameters

_xx	xx component
_xy	xy component
_yx	yx component
_yy	yy component
_dx	dx component
_dy	dy component

setDX()

```
template<typename TItem >
void CTransformMatrix< TItem >::setDX (
            TItem x )  [inline]
```

Sets dx component.

Parameters

x	value to set

setDY()

```
template<typename TItem >
void CTransformMatrix< TItem >::setDY (
            TItem x )  [inline]
```

Sets dy component.

Parameters

x	value to set

setXX()

```
template<typename TItem >
void CTransformMatrix< TItem >::setXX (
            TItem x )  [inline]
```

Sets xx component.

Parameters

x	value to set

setXY()

```
template<typename TItem >
void CTransformMatrix< TItem >::setXY (
            TItem x )  [inline]
```

Sets xy component.

Parameters

x	value to set

setYX()

```
template<typename TItem >
void CTransformMatrix< TItem >::setYX (
            TItem x )  [inline]
```

Sets yx component.

Parameters

x	value to set

setYY()

```
template<typename TItem >
void CTransformMatrix< TItem >::setYY (
            TItem x )  [inline]
```

Sets yy component.

Parameters

x	value to set

transform() [1/2]

```
template<typename TItem >
PointTmpl< TItem > CTransformMatrix< TItem >::transform (
            const PointTmpl< TItem > & point,
            bool ignoreDXDY = false ) const  [inline]
```

Transform a point.

Parameters

point	The point before transformation
ignoreDXDY	Ignore DX & DY for this operation

Returns

CTransformMatrix<**TItem**> Resulting matrix

transform() [2/2]

```
template<typename TItem >
void CTransformMatrix< TItem >::transform (
            PointTmpl< TItem > & result,
            const PointTmpl< TItem > & point,
            bool ignoreDXDY = false ) const  [inline]
```

Transform a point.

Parameters

result	The point after transformation
point	The point before transformation
ignoreDXDY	Ignore DX & DY for this operation

transformRect()

```
template<typename TItem >
void CTransformMatrix< TItem >::transformRect (
            RectTmpl< TItem > & rect,
            bool ignoreDXDY = false ) const  [inline]
```

Transform a rectangle.

Transform a rectangle

Parameters

rect	Rectangle to transform
ignoreDXDY	Ignore DX & DY for this operation

translate()

```
template<typename TItem >
void CTransformMatrix< TItem >::translate (
             TItem dx,
             TItem dy )  [inline]
```

Translate.

Parameters

dx	x-axis transformation value
dy	y-axis transformation value

xx()

```
template<typename TItem >
TItem CTransformMatrix< TItem >::xx ( ) const  [inline]
```

Returns xx component.

Returns

TItem xx component

xy()

```
template<typename TItem >
TItem CTransformMatrix< TItem >::xy ( ) const  [inline]
```

Returns xy component.

Returns

TItem xy component

yx()

```
template<typename TItem >
TItem CTransformMatrix< TItem >::yx ( ) const  [inline]
```

Returns yx component.

Returns

TItem yx component

yy()

```
template<typename TItem >
TItem CTransformMatrix< TItem >::yy ( ) const  [inline]
```

Returns yy component.

Returns

TItem yy component

The documentation for this class was generated from the following file:

- edlgeom.h

7.38 JawsMako::CTransformState Class Reference

Class for tracking the graphics state leading to the point where a transform is applied.

`#include <transforms.h>`

Public Member Functions

- **CTransformState** (const IDOMNodePtr &node)

  Initialise to the state that would be active inside the given node. An atttempt will be made to determine the complete state by contatenating the state of all the parents of the given node. The best results will occur when the node's parents lead to an IDOMFixedPage.
- CTransformState stateInsideNode (const IDOMNodePtr &node, bool updateTransform=true, bool update↩ Clip=true, bool updateRenderingIntent=true, bool updateEdgeMode=true) const

  Return a new state consisting of this state concatenated with the state implied by the given node.
- CTransformState stateInsideBrush (const IDOMBrushPtr &brush, eBrushUsage brushUsage) const

  Return a new state consisting of this state concatenated with the state implied by the given brush.

Public Attributes

- FMatrix **transform**

  The current transformation from default coordinates.
- bool **hasClipBounds**

  True if clipBounds is valid.
- FRect **clipBounds**

  The bounds of the current clipping area.
- IDOMColorSpacePtr **groupColorSpace**

  The current group color space.
- bool **hasRenderingIntent**

  True if an explicit rendering intent has been applied.
- eRenderingIntent **renderingIntent**

  Ignored if hasRenderingIntent is false.
- eBlackPointCompensation **blackPointCompensation**

  The current black point compensation.
- eEdgeMode **edgeMode**

  The current edge mode.
- eBrushUsage **brushUsage**

  Set when we descend into a brush.
- bool inUncoloredTilingBrush
- void * transformPriv

7.38.1 Detailed Description

Class for tracking the graphics state leading to the point where a transform is applied.

Consider for example the IImageDownsamplerTransform described later in this header. In order to determine how to downsample an image, the transform needs to know how large the image will eventually be. The CTransformState provides this information by providing the combined transform that applies to the image based on the RenderTransforms of all the nodes entered leading to the point where the image is actually encountered.

Other transforms need access to other information, such as the approximate clip area, the current group color space, the renderingIntent (if present), the current antialiasing mode (edge mode) and/or how a brush is used.

7.38.2 Member Function Documentation

stateInsideBrush()

```
CTransformState JawsMako::CTransformState::stateInsideBrush (
            const IDOMBrushPtr & brush,
            eBrushUsage brushUsage ) const
```

Return a new state consisting of this state concatenated with the state implied by the given brush.

Parameters

brush	The brush whose state we wish to apply.
brushUsage	The way the brush would be used; ie for filling, stroking, or for use as an opacity mask.

Returns

The new state.

stateInsideNode()

```
CTransformState JawsMako::CTransformState::stateInsideNode (
            const IDOMNodePtr & node,
            bool updateTransform = true,
            bool updateClip = true,
            bool updateRenderingIntent = true,
            bool updateEdgeMode = true ) const
```

Return a new state consisting of this state concatenated with the state implied by the given node.

Parameters

node	The node whose state we wish to apply.
updateTransform	If true, update the transform entry, and also the related black point compensation entry.
updateClip	If true, update the clipBounds entry
updateRenderingIntent	If true, update the rendering intent.

Returns

The new state.

7.38.3 Member Data Documentation

inUncoloredTilingBrush

```
bool JawsMako::CTransformState::inUncoloredTilingBrush
```

Set to true when we descend into an uncolored tiling brush (An IDOMTilingPatternBrush with PaintType == 2)

transformPriv

```
void* JawsMako::CTransformState::transformPriv
```

Private contextual information for use by the transform implementation

The documentation for this class was generated from the following file:

- transforms.h

7.39 JawsMako::CXFAPacket Class Reference

A class encapsulating an entry in an XFA array.

```
#include <interactive.h>
```

Public Attributes

- U8String **name**

  If empty and this is the only packet in the array, it represents the entire XFA package.
- IInputStreamPtr **stream**

  The stream containing the XFA packet.

7.39.1 Detailed Description

A class encapsulating an entry in an XFA array.

The documentation for this class was generated from the following file:

- interactive.h

7.40 JawsMako::IAnnotationUtils::CXMLResource Class Reference

Simple class for tracking streams associated with XML generated by generateXMLForDocument()

```
#include <interactive.h>
```

7.40.1 Detailed Description

Simple class for tracking streams associated with XML generated by generateXMLForDocument()

The documentation for this class was generated from the following file:

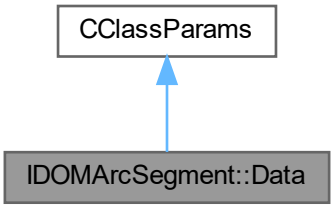
- interactive.h

7.41 IDOMArcSegment::Data Class Reference

Initialization data.

`#include <idompathgeometry.h>`

Inheritance diagram for IDOMArcSegment::Data:



7.41.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

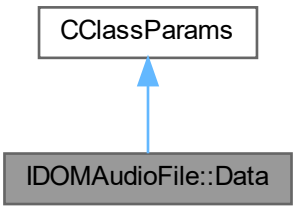
- idompathgeometry.h

7.42 IDOMAudioFile::Data Class Reference

Initialization data.

`#include <idomresources.h>`

Inheritance diagram for IDOMAudioFile::Data:



7.42.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

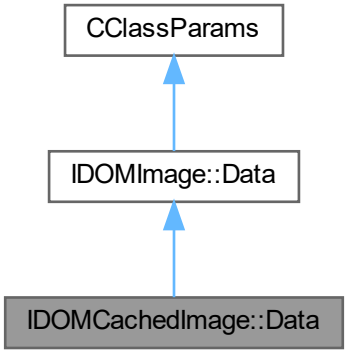
- idomresources.h

7.43 IDOMCachedImage::Data Class Reference

Initialization data.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMCachedImage::Data:



7.43.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

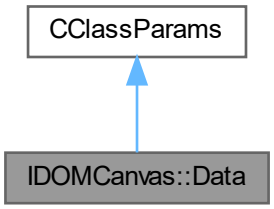
- idomimageresource.h

7.44 IDOMCanvas::Data Class Reference

Initialization data.

`#include <idomcanvas.h>`

Inheritance diagram for IDOMCanvas::Data:



7.44.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

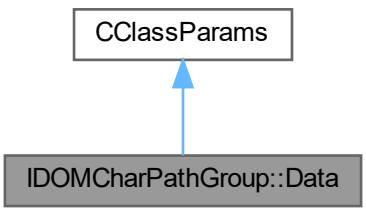
- idomcanvas.h

7.45 IDOMCharPathGroup::Data Class Reference

Initialization data.

`#include <idomcharpathgroup.h>`

Inheritance diagram for IDOMCharPathGroup::Data:



7.45.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

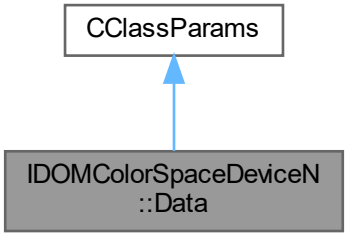
- idomcharpathgroup.h

7.46 IDOMColorSpaceDeviceN::Data Class Reference

Initialization data.

`#include <idomcolorspace.h>`

Inheritance diagram for IDOMColorSpaceDeviceN::Data:



7.46.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

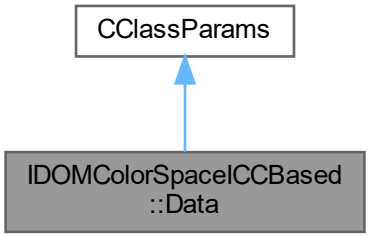
- idomcolorspace.h

7.47 IDOMColorSpaceICCBased::Data Class Reference

Initialization data.

`#include <idomcolorspace.h>`

Inheritance diagram for IDOMColorSpaceICCBased::Data:



7.47.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

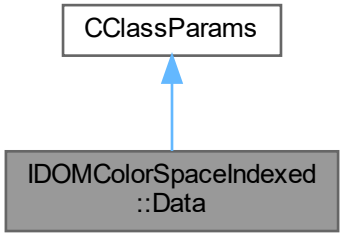
- idomcolorspace.h

7.48 IDOMColorSpaceIndexed::Data Class Reference

Initialization data.

`#include <idomcolorspace.h>`

Inheritance diagram for IDOMColorSpaceIndexed::Data:



7.48.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

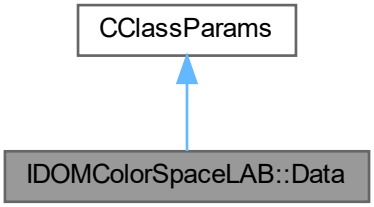
- idomcolorspace.h

7.49 IDOMColorSpaceLAB::Data Class Reference

Initialization data.

```
#include <idomcolorspace.h>
```

Inheritance diagram for IDOMColorSpaceLAB::Data:



7.49.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

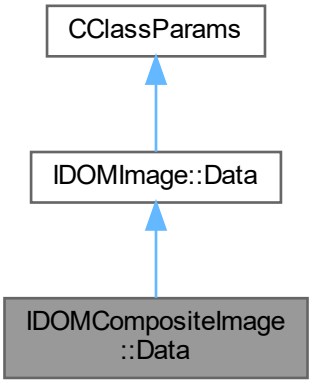
- idomcolorspace.h

7.50 IDOMCompositeImage::Data Class Reference

Initialization data.

```
#include <idomimageresource.h>
```

Inheritance diagram for IDOMCompositeImage::Data:



7.50.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

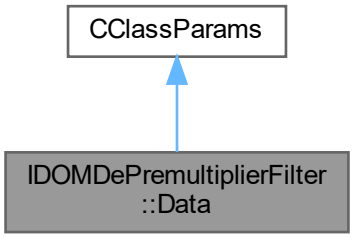
- idomimageresource.h

7.51 IDOMDePremultiplierFilter::Data Class Reference

Initialization data.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMDePremultiplierFilter::Data:



7.51.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

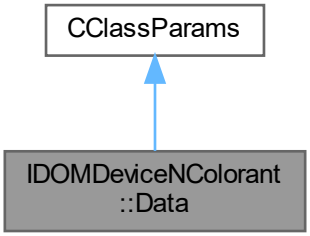
- idomimageresource.h

7.52 IDOMDeviceNColorant::Data Class Reference

Initialization data.

`#include <idomcolorspace.h>`

Inheritance diagram for IDOMDeviceNColorant::Data:



7.52.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

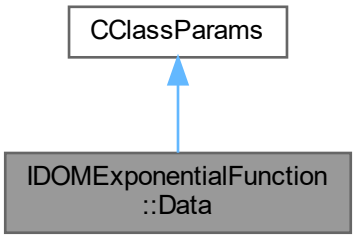
- idomcolorspace.h

7.53 IDOMExponentialFunction::Data Class Reference

Initialization data.

`#include <idomfunction.h>`

Inheritance diagram for IDOMExponentialFunction::Data:



7.53.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

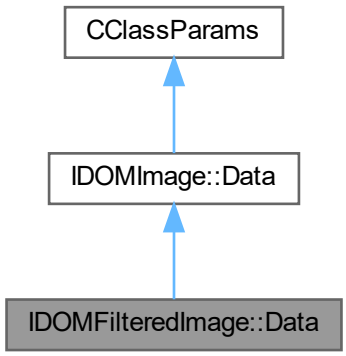
- idomfunction.h

7.54 IDOMFilteredImage::Data Class Reference

Initialization data.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMFilteredImage::Data:



7.54.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

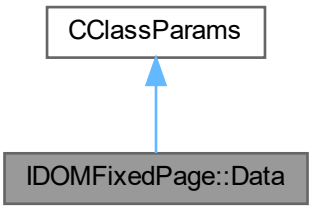
- idomimageresource.h

7.55 IDOMFixedPage::Data Class Reference

Initialization data.

`#include <idompage.h>`

Inheritance diagram for IDOMFixedPage::Data:



7.55.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

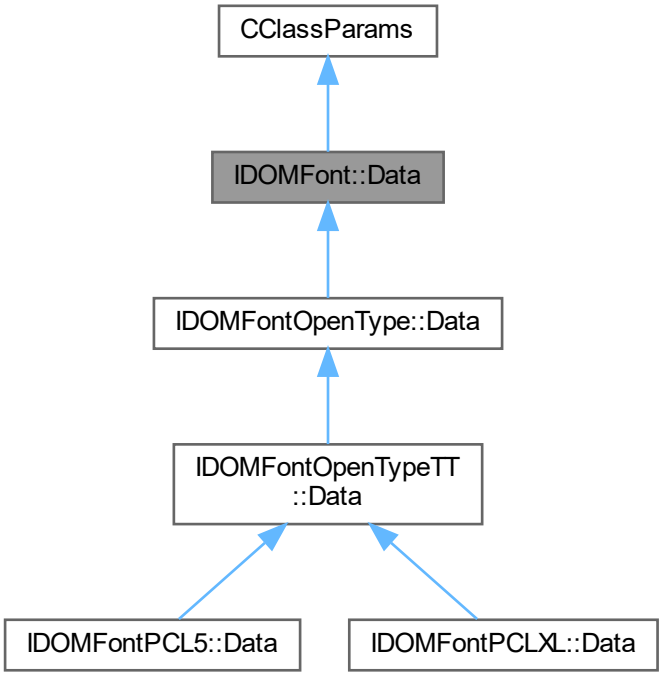
- idompage.h

7.56 IDOMFont::Data Class Reference

Initialization data.

`#include <idomfont.h>`

Inheritance diagram for IDOMFont::Data:



7.56.1 Detailed Description

Initialization data.

```
@defgroup IDOMFontData IDOMFont data
@ingroup  IFont
```

The documentation for this class was generated from the following file:

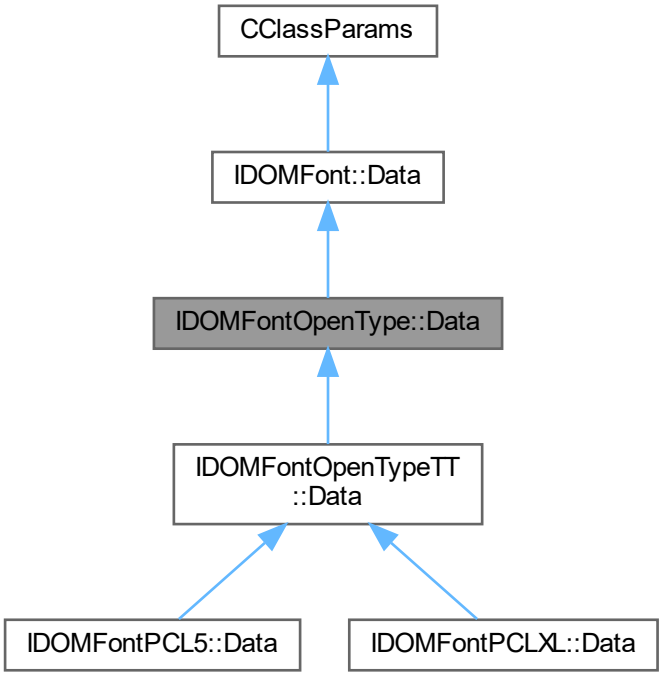
- idomfont.h

7.57 IDOMFontOpenType::Data Class Reference

Initialization data.

```
#include <idomfont.h>
```

Inheritance diagram for IDOMFontOpenType::Data:



7.57.1 Detailed Description

Initialization data.

```
@defgroup IDOMFontOpenTypeData IDOMFontOpenType initialization data
@ingroup  IFont
```

The documentation for this class was generated from the following file:

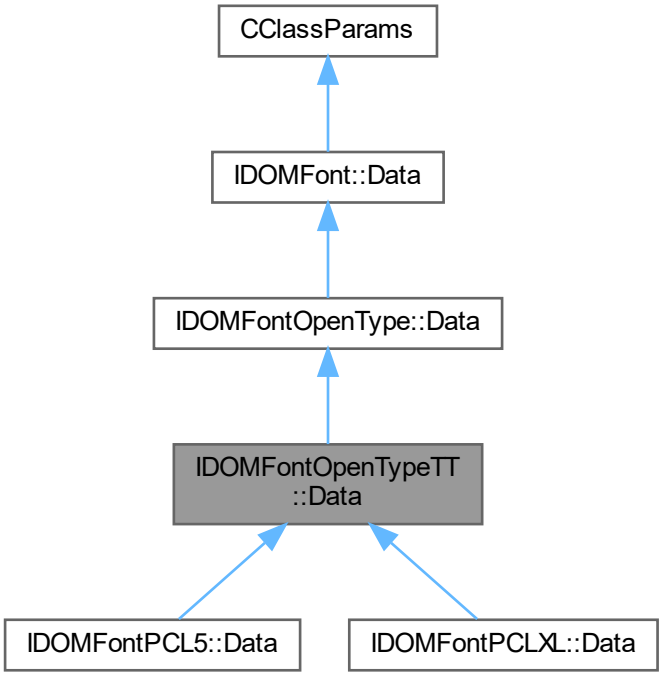
- idomfont.h

7.58 IDOMFontOpenTypeTT::Data Class Reference

Initialization data.

```
#include <idomfont.h>
```

Inheritance diagram for IDOMFontOpenTypeTT::Data:



7.58.1 Detailed Description

Initialization data.

```
@defgroup IDOMFontOpenTypeData IDOMFontOpenType initialization data
@ingroup  IFont
```

The documentation for this class was generated from the following file:

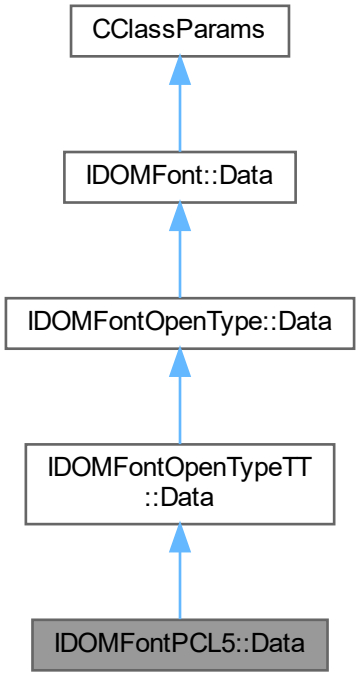
- idomfont.h

7.59 IDOMFontPCL5::Data Class Reference

Initialization data.

```
#include <idomfontpcl.h>
```

Inheritance diagram for IDOMFontPCL5::Data:



7.59.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

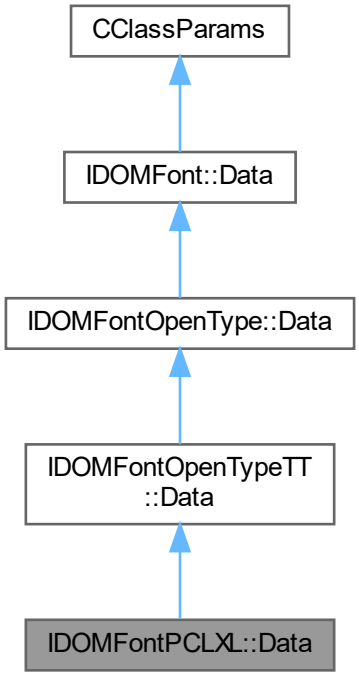
- idomfontpcl.h

7.60 IDOMFontPCLXL::Data Class Reference

Initialization data.

`#include <idomfontpcl.h>`

Inheritance diagram for IDOMFontPCLXL::Data:



7.60.1 Detailed Description

Initialization data.

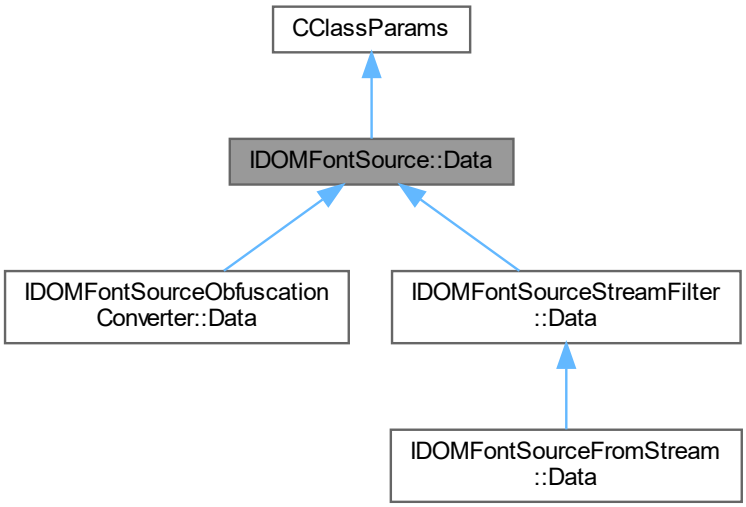
The documentation for this class was generated from the following file:

- idomfontpcl.h

7.61 IDOMFontSource::Data Class Reference

```
#include <idomfont.h>
```

Inheritance diagram for IDOMFontSource::Data:



7.61.1 Detailed Description

```
@defgroup IDOMFontSourceData IDOMFontSource data
@ingroup  IFont
@{
```

The documentation for this class was generated from the following file:

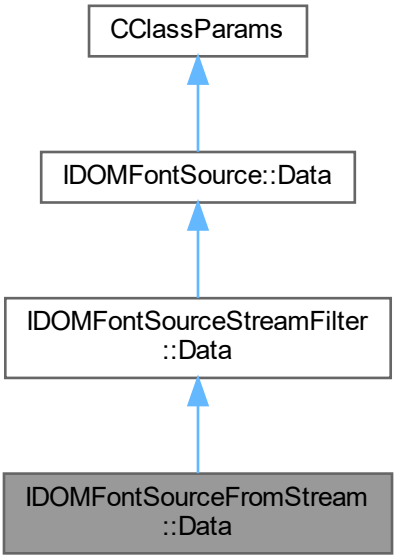
- idomfont.h

7.62 IDOMFontSourceFromStream::Data Class Reference

Initialization data.

```
#include <idomfont.h>
```

Inheritance diagram for IDOMFontSourceFromStream::Data:



7.62.1 Detailed Description

Initialization data.

```
@defgroup IDOMFontSourceFromStreamData IDOMFontSourceFromStream initialization data
@ingroup  IFont
```

The documentation for this class was generated from the following file:

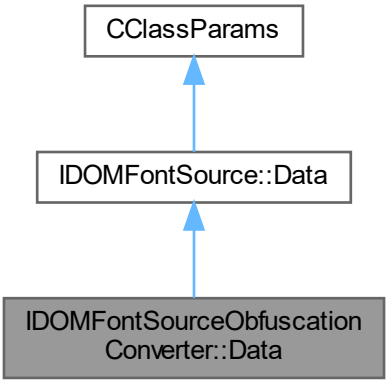
- idomfont.h

7.63 IDOMFontSourceObfuscationConverter::Data Class Reference

Initialization data.

```
#include <idomfont.h>
```

Inheritance diagram for IDOMFontSourceObfuscationConverter::Data:



Public Attributes

- IDOMFontSourcePtr **inputFontSource**

  The input font source for the converter.
- eOperation **operation**

  The obfuscation or deobfuscation operation.

7.63.1 Detailed Description

Initialization data.

```
@defgroup IDOMFontSourceObfuscationConverterData IDOMFontSourceObfuscationConverter data
@ingroup  IFont
```

The documentation for this class was generated from the following file:

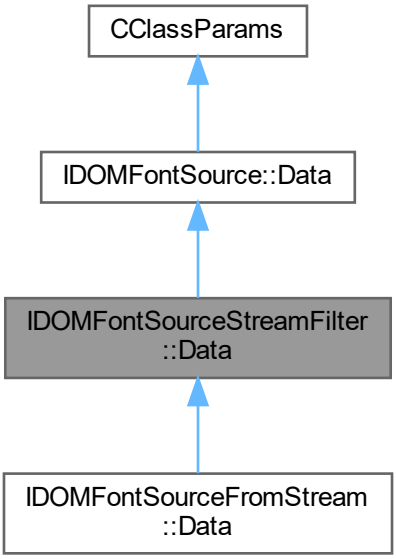
- idomfont.h

7.64 IDOMFontSourceStreamFilter::Data Class Reference

Initialization data.

```
#include <idomfont.h>
```

Inheritance diagram for IDOMFontSourceStreamFilter::Data:



7.64.1 Detailed Description

Initialization data.

```
@defgroup IDOMFontSourceStreamFilterData IDOMFontSourceStreamFilter data
@ingroup  IFont
```

The documentation for this class was generated from the following file:

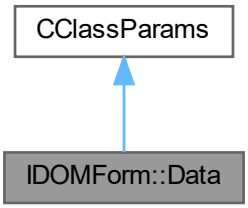
- idomfont.h

7.65 IDOMForm::Data Class Reference

Initialization data.

```
#include <idomform.h>
```

Inheritance diagram for IDOMForm::Data:



7.65.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

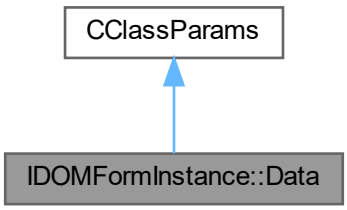
- idomform.h

7.66 IDOMFormInstance::Data Class Reference

Initialization data.

`#include <idomform.h>`

Inheritance diagram for IDOMFormInstance::Data:



7.66.1 Detailed Description

Initialization data.

`@class Data`

The documentation for this class was generated from the following file:

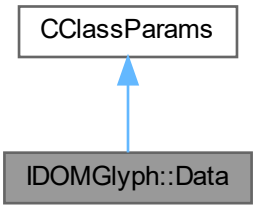
- idomform.h

7.67 IDOMGlyph::Data Class Reference

Initialization data.

`#include <idomglyph.h>`

Inheritance diagram for IDOMGlyph::Data:



7.67.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

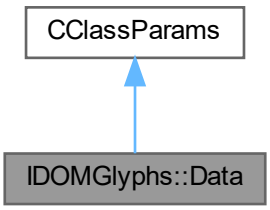
- idomglyph.h

7.68 IDOMGlyphs::Data Class Reference

Initialization data.

`#include <idomglyphs.h>`

Inheritance diagram for IDOMGlyphs::Data:



7.68.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

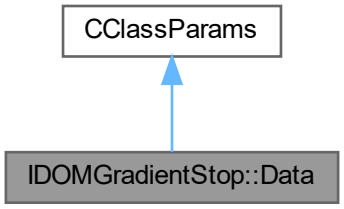
- idomglyphs.h

7.69 IDOMGradientStop::Data Class Reference

Initialization data.

`#include <idombrush.h>`

Inheritance diagram for IDOMGradientStop::Data:



7.69.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

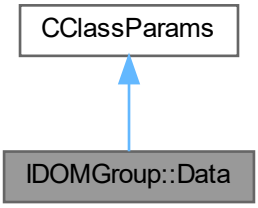
- idombrush.h

7.70 IDOMGroup::Data Class Reference

Initialization data.

`#include <idomgroup.h>`

Inheritance diagram for IDOMGroup::Data:



7.70.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

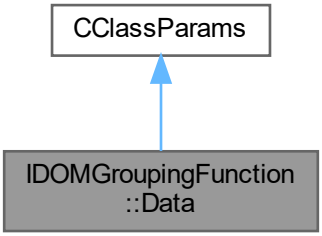
- idomgroup.h

7.71 IDOMGroupingFunction::Data Class Reference

Initialization data.

`#include <idomfunction.h>`

Inheritance diagram for IDOMGroupingFunction::Data:



7.71.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

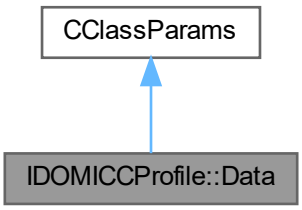
- idomfunction.h

7.72 IDOMICCProfile::Data Class Reference

Initialization data.

`#include <idomresources.h>`

Inheritance diagram for IDOMICCProfile::Data:



7.72.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

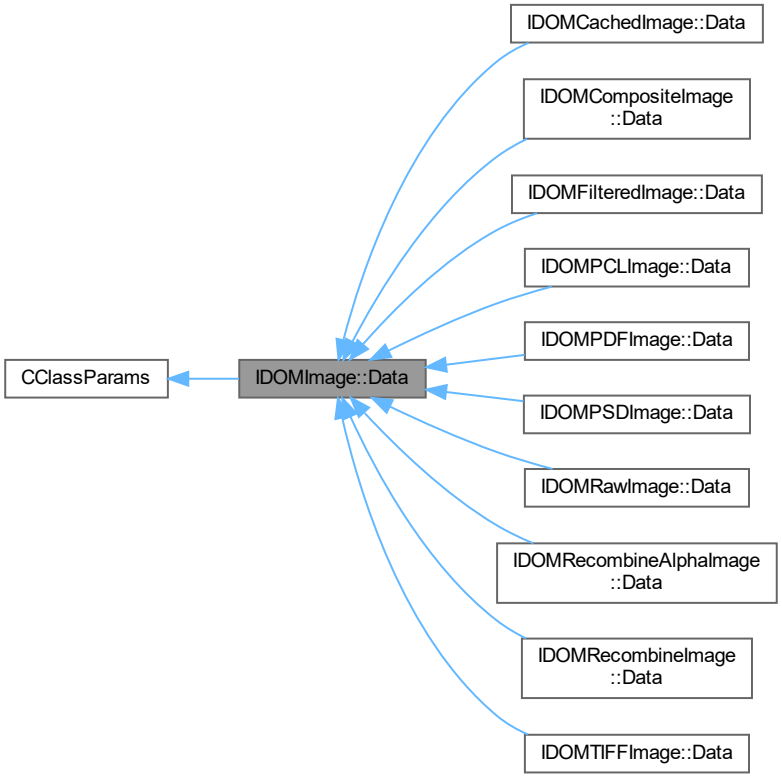
- idomresources.h

7.73 IDOMImage::Data Class Reference

Initialization data.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMImage::Data:



7.73.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

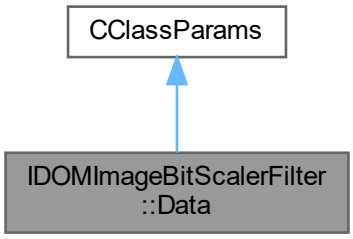
- idomimageresource.h

7.74 IDOMImageBitScalerFilter::Data Class Reference

Initialization data.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMImageBitScalerFilter::Data:



7.74.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

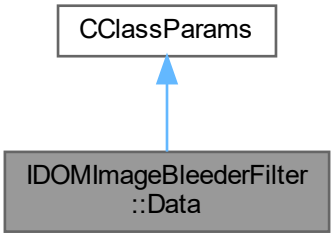
- idomimageresource.h

7.75 IDOMImageBleederFilter::Data Class Reference

Initialization data.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMImageBleederFilter::Data:



7.75.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

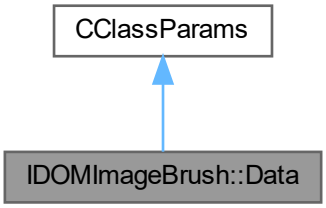
- idomimageresource.h

7.76 IDOMImageBrush::Data Class Reference

Initialization data.

`#include <idombrush.h>`

Inheritance diagram for IDOMImageBrush::Data:



7.76.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

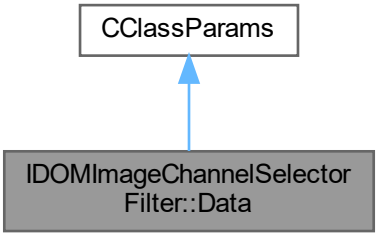
- idombrush.h

7.77 IDOMImageChannelSelectorFilter::Data Class Reference

Initialization data.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMImageChannelSelectorFilter::Data:



7.77.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

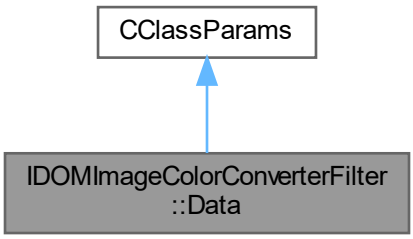
- idomimageresource.h

7.78 IDOMImageColorConverterFilter::Data Class Reference

Initialization data.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMImageColorConverterFilter::Data:



7.78.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

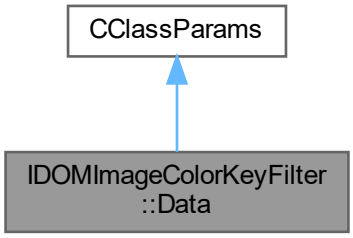
- idomimageresource.h

7.79 IDOMImageColorKeyFilter::Data Class Reference

Initialization data.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMImageColorKeyFilter::Data:



7.79.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

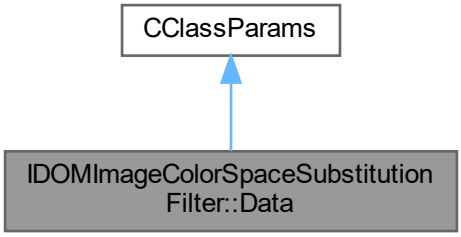
- idomimageresource.h

7.80 IDOMImageColorSpaceSubstitutionFilter::Data Class Reference

Initialization data.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMImageColorSpaceSubstitutionFilter::Data:



7.80.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

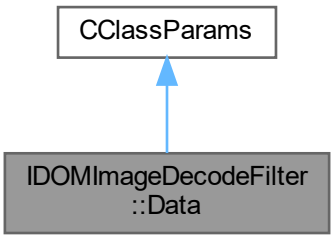
- idomimageresource.h

7.81 IDOMImageDecodeFilter::Data Class Reference

Initialization data.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMImageDecodeFilter::Data:



7.81.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

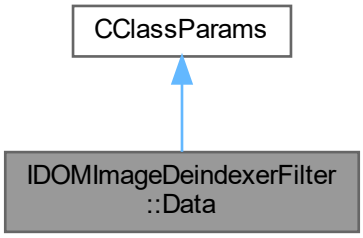
- idomimageresource.h

7.82 IDOMImageDeindexerFilter::Data Class Reference

Initialization data.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMImageDeindexerFilter::Data:



7.82.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

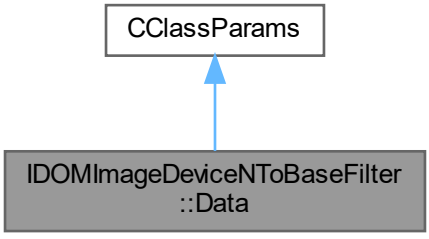
- idomimageresource.h

7.83 IDOMImageDeviceNToBaseFilter::Data Class Reference

Initialization data.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMImageDeviceNToBaseFilter::Data:



7.83.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

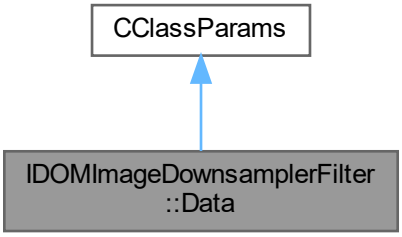
- idomimageresource.h

7.84 IDOMImageDownsamplerFilter::Data Class Reference

Initialization data.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMImageDownsamplerFilter::Data:



7.84.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

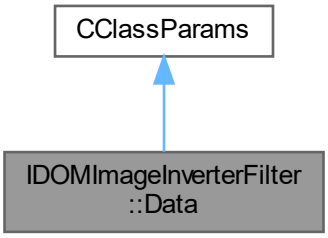
- idomimageresource.h

7.85 IDOMImageInverterFilter::Data Class Reference

Initialization data.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMImageInverterFilter::Data:



7.85.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

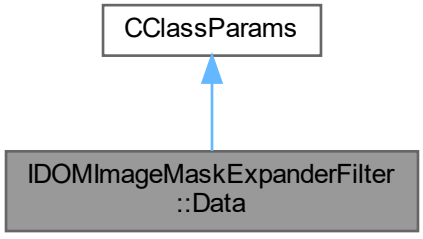
- idomimageresource.h

7.86 IDOMImageMaskExpanderFilter::Data Class Reference

Initialization data.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMImageMaskExpanderFilter::Data:



7.86.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

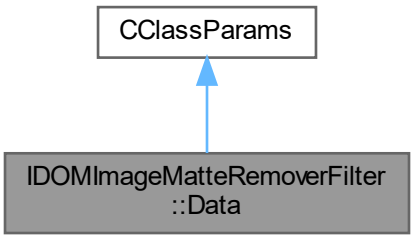
- idomimageresource.h

7.87 IDOMImageMatteRemoverFilter::Data Class Reference

Initialization data.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMImageMatteRemoverFilter::Data:



7.87.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

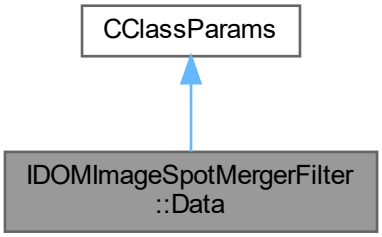
- idomimageresource.h

7.88 IDOMImageSpotMergerFilter::Data Class Reference

Initialization data.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMImageSpotMergerFilter::Data:



7.88.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

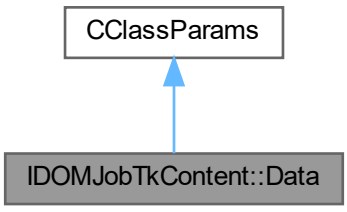
- idomimageresource.h

7.89 IDOMJobTkContent::Data Class Reference

Initialization data.

`#include <idomjobtk.h>`

Inheritance diagram for IDOMJobTkContent::Data:



Public Attributes

- eDOMJobTkLevel **level**

  The level of the JobTicket content (DocumentSequence, FixedDocument, Fixedpage).
- double **version**

  Version of the JobTicket content.
- bool **modified**

  User should set this flag if job ticked content was modified.

7.89.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

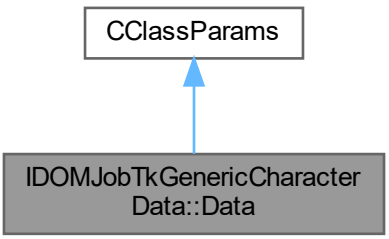
- idomjobtk.h

7.90 IDOMJobTkGenericCharacterData::Data Class Reference

Initialization data.

`#include <idomjobtk.h>`

Inheritance diagram for IDOMJobTkGenericCharacterData::Data:



7.90.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

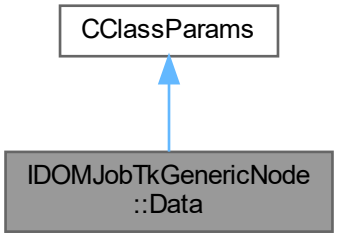
- idomjobtk.h

7.91 IDOMJobTkGenericNode::Data Class Reference

Initialization data.

`#include <idomjobtk.h>`

Inheritance diagram for IDOMJobTkGenericNode::Data:



7.91.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

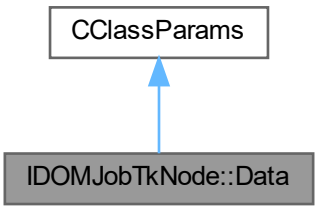
- idomjobtk.h

7.92 IDOMJobTkNode::Data Class Reference

Initialization data.

`#include <idomjobtk.h>`

Inheritance diagram for IDOMJobTkNode::Data:



Public Attributes

- EDLQName **name**

  Name of the node.
- eDOMJobTkNodeType **jobTkNodeType**

  Job Ticket node type.

7.92.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

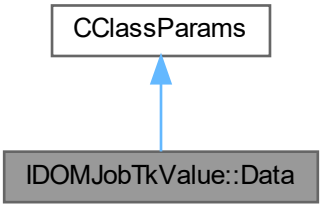
- idomjobtk.h

7.93 IDOMJobTkValue::Data Class Reference

Initialization data.

`#include <idomjobtk.h>`

Inheritance diagram for IDOMJobTkValue::Data:



7.93.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

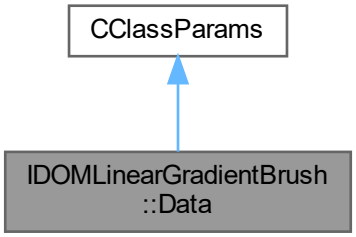
- idomjobtk.h

7.94 IDOMLinearGradientBrush::Data Class Reference

Initialization data.

`#include <idombrush.h>`

Inheritance diagram for IDOMLinearGradientBrush::Data:



7.94.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

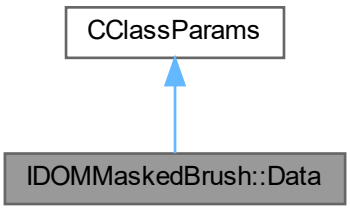
- idombrush.h

7.95 IDOMMaskedBrush::Data Class Reference

Initialization data.

`#include <idombrush.h>`

Inheritance diagram for IDOMMaskedBrush::Data:



7.95.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

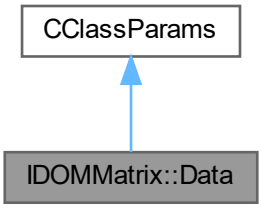
- idombrush.h

7.96 IDOMMatrix::Data Class Reference

Initialization data.

`#include <idomresources.h>`

Inheritance diagram for IDOMMatrix::Data:



7.96.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

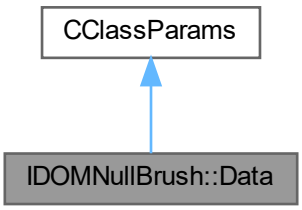
- idomresources.h

7.97 IDOMNullBrush::Data Class Reference

Initialization data.

`#include <idombrush.h>`

Inheritance diagram for IDOMNullBrush::Data:



7.97.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

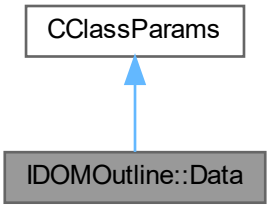
- idombrush.h

7.98 IDOMOutline::Data Class Reference

Initialization data.

`#include <idomoutline.h>`

Inheritance diagram for IDOMOutline::Data:



7.98.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

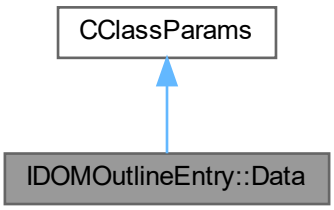
- idomoutline.h

7.99 IDOMOutlineEntry::Data Class Reference

Initialization data.

`#include <idomoutline.h>`

Inheritance diagram for IDOMOutlineEntry::Data:



7.99.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

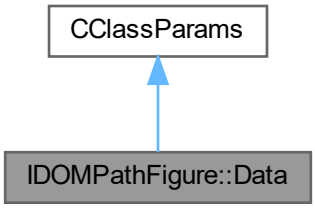
- idomoutline.h

7.100 IDOMPathFigure::Data Class Reference

Initialization data.

`#include <idompathgeometry.h>`

Inheritance diagram for IDOMPathFigure::Data:



7.100.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

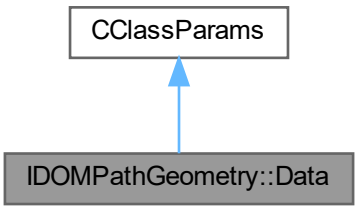
- idompathgeometry.h

7.101 IDOMPathGeometry::Data Class Reference

Initialization data.

`#include <idompathgeometry.h>`

Inheritance diagram for IDOMPathGeometry::Data:



7.101.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

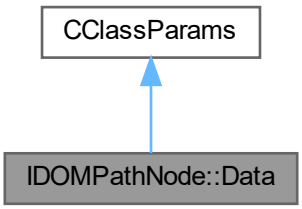
- idompathgeometry.h

7.102 IDOMPathNode::Data Class Reference

Initialization data.

`#include <idompath.h>`

Inheritance diagram for IDOMPathNode::Data:



7.102.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

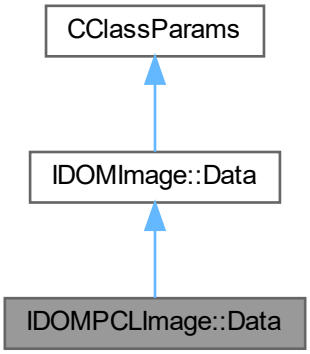
- idompath.h

7.103 IDOMPCLImage::Data Class Reference

Initialization data.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMPCLImage::Data:



7.103.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

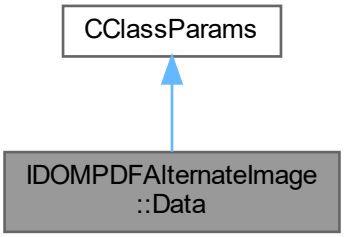
- idomimageresource.h

7.104 IDOMPDFAlternateImage::Data Class Reference

Initialization data.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMPDFAlternateImage::Data:



7.104.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

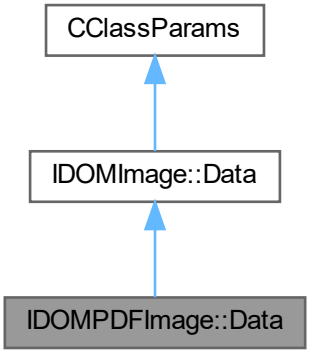
- idomimageresource.h

7.105 IDOMPDFImage::Data Class Reference

Initialization data.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMPDFImage::Data:



7.105.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

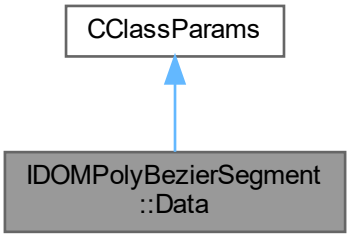
- idomimageresource.h

7.106 IDOMPolyBezierSegment::Data Class Reference

Initialization data.

`#include <idompathgeometry.h>`

Inheritance diagram for IDOMPolyBezierSegment::Data:



7.106.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

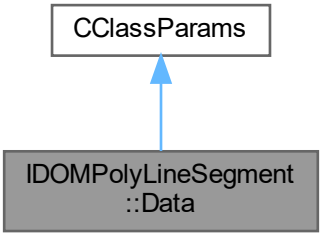
- idompathgeometry.h

7.107 IDOMPolyLineSegment::Data Class Reference

Initialization data.

`#include <idompathgeometry.h>`

Inheritance diagram for IDOMPolyLineSegment::Data:



7.107.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

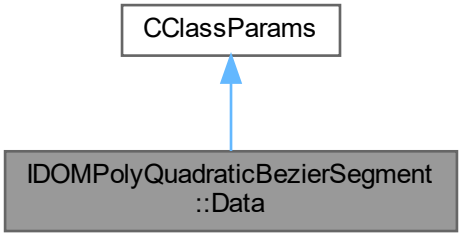
- idompathgeometry.h

7.108 IDOMPolyQuadraticBezierSegment::Data Class Reference

Initialization data.

`#include <idompathgeometry.h>`

Inheritance diagram for IDOMPolyQuadraticBezierSegment::Data:



7.108.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

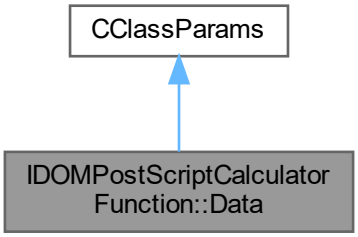
- idompathgeometry.h

7.109 IDOMPostScriptCalculatorFunction::Data Class Reference

Initialization data.

`#include <idomfunction.h>`

Inheritance diagram for IDOMPostScriptCalculatorFunction::Data:



7.109.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

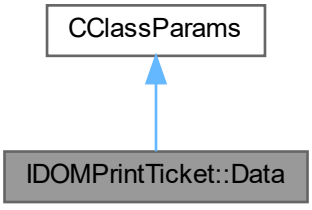
- idomfunction.h

7.110 IDOMPrintTicket::Data Class Reference

Initialization data.

`#include <idomresources.h>`

Inheritance diagram for IDOMPrintTicket::Data:



7.110.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

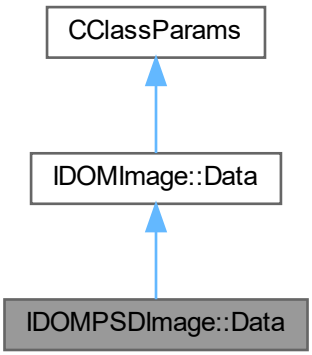
- idomresources.h

7.111 IDOMPSDImage::Data Class Reference

Initialization data.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMPSDImage::Data:



7.111.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

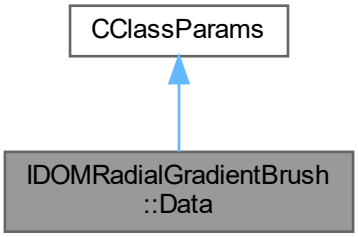
- idomimageresource.h

7.112 IDOMRadialGradientBrush::Data Class Reference

Initialization data.

`#include <idombrush.h>`

Inheritance diagram for IDOMRadialGradientBrush::Data:



7.112.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

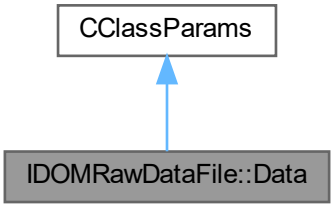
- idombrush.h

7.113 IDOMRawDataFile::Data Class Reference

Initialization data.

`#include <idomresources.h>`

Inheritance diagram for IDOMRawDataFile::Data:



7.113.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

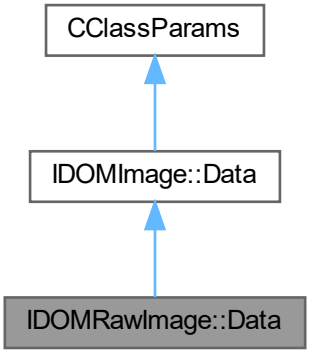
- idomresources.h

7.114 IDOMRawImage::Data Class Reference

Initialization data.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMRawImage::Data:



7.114.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

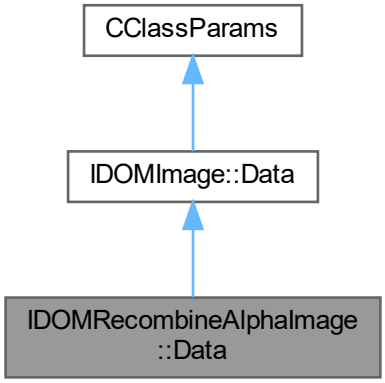
- idomimageresource.h

7.115 IDOMRecombineAlphaImage::Data Class Reference

Initialization data.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMRecombineAlphaImage::Data:



7.115.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

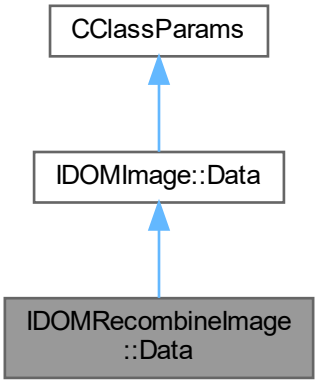
- idomimageresource.h

7.116 IDOMRecombineImage::Data Class Reference

Initialization data.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMRecombineImage::Data:



7.116.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

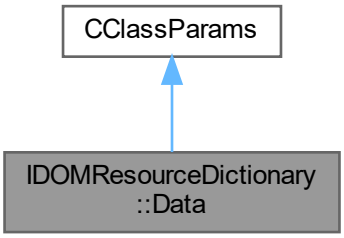
- idomimageresource.h

7.117 IDOMResourceDictionary::Data Class Reference

Initialization data.

`#include <idomresources.h>`

Inheritance diagram for IDOMResourceDictionary::Data:



7.117.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

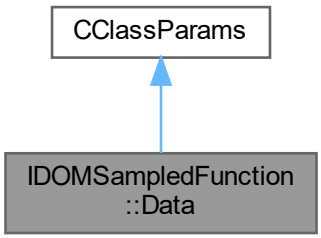
- idomresources.h

7.118 IDOMSampledFunction::Data Class Reference

Initialization data.

`#include <idomfunction.h>`

Inheritance diagram for IDOMSampledFunction::Data:



7.118.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

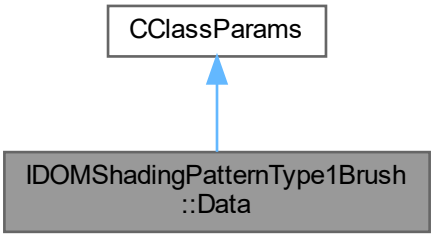
- idomfunction.h

7.119 IDOMShadingPatternType1Brush::Data Class Reference

Initialization data.

`#include <idombrush.h>`

Inheritance diagram for IDOMShadingPatternType1Brush::Data:



7.119.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

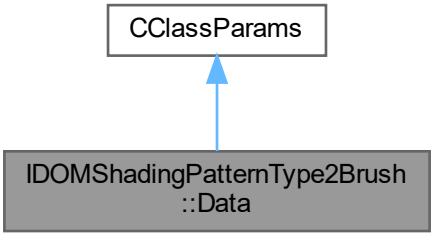
- idombrush.h

7.120 IDOMShadingPatternType2Brush::Data Class Reference

Initialization data.

`#include <idombrush.h>`

Inheritance diagram for IDOMShadingPatternType2Brush::Data:



7.120.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

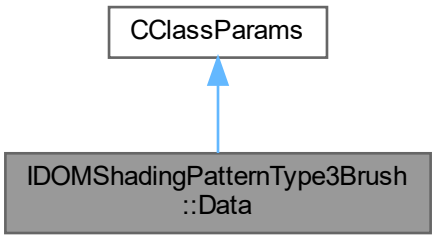
- idombrush.h

7.121 IDOMShadingPatternType3Brush::Data Class Reference

Initialization data.

`#include <idombrush.h>`

Inheritance diagram for IDOMShadingPatternType3Brush::Data:



7.121.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

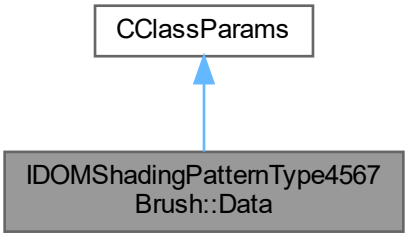
- idombrush.h

7.122 IDOMShadingPatternType4567Brush::Data Class Reference

Initialization data.

`#include <idombrush.h>`

Inheritance diagram for IDOMShadingPatternType4567Brush::Data:



7.122.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

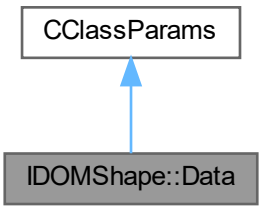
- idombrush.h

7.123 IDOMShape::Data Class Reference

Initialization data.

`#include <idomshape.h>`

Inheritance diagram for IDOMShape::Data:



7.123.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

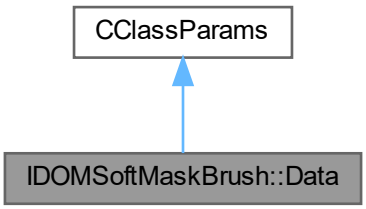
- idomshape.h

7.124 IDOMSoftMaskBrush::Data Class Reference

Initialization data.

`#include <idombrush.h>`

Inheritance diagram for IDOMSoftMaskBrush::Data:



7.124.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

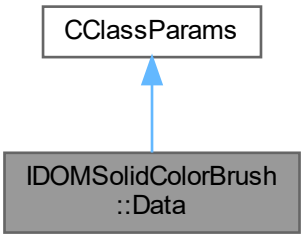
- idombrush.h

7.125 IDOMSolidColorBrush::Data Class Reference

Initialization data.

`#include <idombrush.h>`

Inheritance diagram for IDOMSolidColorBrush::Data:



7.125.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

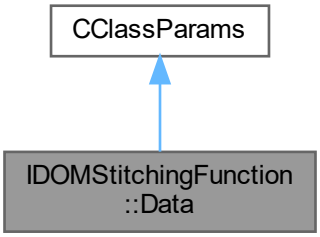
- idombrush.h

7.126 IDOMStitchingFunction::Data Class Reference

Initialization data.

`#include <idomfunction.h>`

Inheritance diagram for IDOMStitchingFunction::Data:



7.126.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

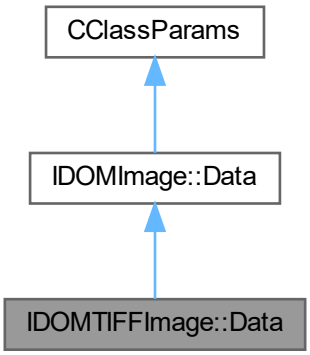
- idomfunction.h

7.127 IDOMTIFFImage::Data Class Reference

Initialization data.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMTIFFImage::Data:



7.127.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

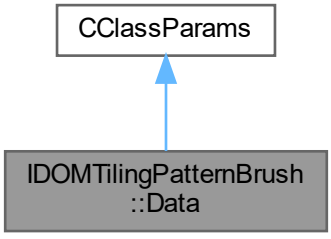
- idomimageresource.h

7.128 IDOMTilingPatternBrush::Data Class Reference

Initialization data.

`#include <idombrush.h>`

Inheritance diagram for IDOMTilingPatternBrush::Data:



7.128.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

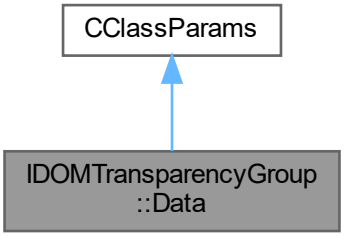
- idombrush.h

7.129 IDOMTransparencyGroup::Data Class Reference

Initialization data.

`#include <idomgroup.h>`

Inheritance diagram for IDOMTransparencyGroup::Data:



7.129.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

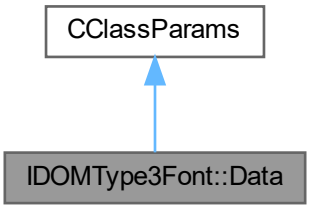
- idomgroup.h

7.130 IDOMType3Font::Data Class Reference

Initialization data.

`#include <idomfont.h>`

Inheritance diagram for IDOMType3Font::Data:



7.130.1 Detailed Description

Initialization data.

```
@defgroup IDOMType3FontData IDOMType3Font initialization data
@ingroup  IFont
```

The documentation for this class was generated from the following file:

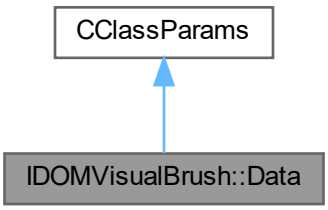
- idomfont.h

7.131 IDOMVisualBrush::Data Class Reference

Initialization data.

`#include <idombrush.h>`

Inheritance diagram for IDOMVisualBrush::Data:



7.131.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

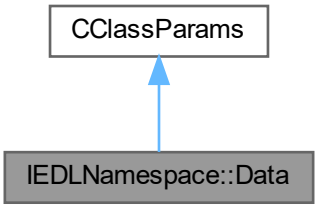
- idombrush.h

7.132 IEDLNamespace::Data Class Reference

Initialization data.

`#include <edlqname.h>`

Inheritance diagram for IEDLNamespace::Data:



7.132.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

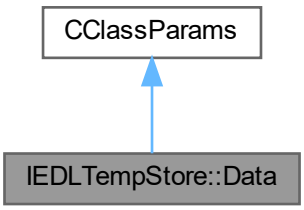
- edlqname.h

7.133 IEDLTempStore::Data Class Reference

Initialization data.

`#include <iedltempstore.h>`

Inheritance diagram for IEDLTempStore::Data:



7.133.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

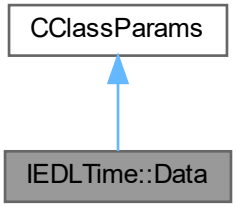
- iedltempstore.h

7.134 IEDLTime::Data Class Reference

Initialization data.

`#include <edltime.h>`

Inheritance diagram for IEDLTime::Data:



7.134.1 Detailed Description

Initialization data.

The documentation for this class was generated from the following file:

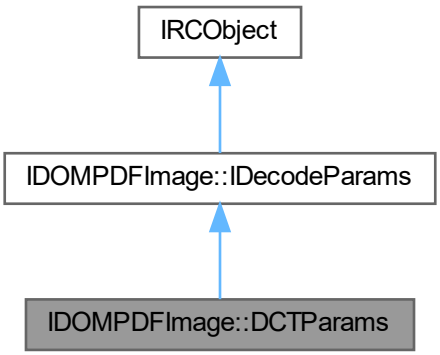
- edltime.h

7.135 IDOMPDFImage::DCTParams Class Reference

Class to hold filter parameters for DCT-compressed image data. Please see the PDF specification for the meaning of these parameters.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMPDFImage::DCTParams:



Additional Inherited Members

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.135.1 Detailed Description

Class to hold filter parameters for DCT-compressed image data. Please see the PDF specification for the meaning of these parameters.

The documentation for this class was generated from the following file:

- idomimageresource.h

7.136 EDLIFStream Class Reference

An ifstream that can deal with UTF8 file names on all platforms.

`#include <edlstream.h>`

Inherits std::ifstream.

7.136.1 Detailed Description

An ifstream that can deal with UTF8 file names on all platforms.

The documentation for this class was generated from the following file:

- edlstream.h

7.137 EDLOFStream Class Reference

An ofstream that can deal with UTF8 file names on all platforms.

`#include <edlstream.h>`

Inherits std::ofstream.

7.137.1 Detailed Description

An ofstream that can deal with UTF8 file names on all platforms.

The documentation for this class was generated from the following file:

- edlstream.h

7.138 EDLQName Class Reference

Implementation of qualified name class.

`#include <edlqname.h>`

Public Member Functions

- **EDLQName** ()
  Empty constructor of EDLQName.
- **EDLQName** (const EDLQName &another)
  Copy constructor of EDLQName.
- EDLQName (IEDLNamespacePtr ptrNamespace, const EDLString &name)
  constructor of EDLQName
- EDLQName (IEDLNamespacePtr ptrNamespace, const EDLSysString &name)
  constructor of EDLQName
- void operator= (const EDLQName &another)
  operator =
- bool operator== (const EDLQName &another) const
  operator ==
- IEDLNamespacePtr getNamespace () const
  Retrieves namespace.
- bool setNamespace (const IEDLNamespacePtr &ns)
  Sets namespace.
- EDLString getName () const
  Retrieves member name of EDLQName.
- bool setName (const EDLString &name)
  Sets member name of EDLQName.
- bool setName (const EDLSysString &sysname)
  Sets member name of EDLQName.
- EDLString getNameWithPrefix () const
  Returns EDLString that is combination of prefix and name (prefix:name)
- bool isEmpty ()
  Returns a boolean that says where this QName was empty.

7.138.1 Detailed Description

Implementation of qualified name class.

7.138.2 Constructor & Destructor Documentation

EDLQName() [1/2]

```
EDLQName::EDLQName (
            IEDLNamespacePtr ptrNamespace,
            const EDLString & name )  [inline]
```

constructor of EDLQName

Parameters

ptrNamespace	Smart pointer to the namespace interface
name	The name value

EDLQName() [2/2]

```
EDLQName::EDLQName (
            IEDLNamespacePtr ptrNamespace,
            const EDLSysString & name )  [inline]
```

constructor of EDLQName

Parameters

ptrNamespace	Smart pointer to the namespace interface
name	The name value

7.138.3 Member Function Documentation

getName()

```
EDLString EDLQName::getName ( ) const  [inline]
```

Retrieves member name of EDLQName.

Returns

EDLString The name value as EDLString

getNamespace()

```
IEDLNamespacePtr EDLQName::getNamespace ( ) const  [inline]
```

Retrieves namespace.

Returns

IEDLNamespacePtr The smart pointer to the namespace interface

getNameWithPrefix()

```
EDLString EDLQName::getNameWithPrefix ( ) const  [inline]
```

Returns EDLString that is combination of prefix and name (prefix:name)

Returns

EDLString The fullname

isEmpty()

```
bool EDLQName::isEmpty ( )  [inline]
```

Returns a boolean that says where this QName was empty.

Returns

bool True if the name is empty

operator=()

```
void EDLQName::operator= (
            const EDLQName & another )  [inline]
```

operator =

Parameters

another	EDLQName

operator==()

```
bool EDLQName::operator== (
            const EDLQName & another ) const  [inline]
```

operator ==

Parameters

another	EDLQName

Returns

bool True if equal, false if not

setName() [1/2]

```
bool EDLQName::setName (
            const EDLString & name )  [inline]
```

Sets member name of EDLQName.

Parameters

name	The name value

Returns

bool True

setName() [2/2]

```
bool EDLQName::setName (
            const EDLSysString & sysname )  [inline]
```

Sets member name of EDLQName.

Parameters

sysname	The name value

Returns

bool True

setNamespace()

```
bool EDLQName::setNamespace (
            const IEDLNamespacePtr & ns )  [inline]
```

Sets namespace.

Parameters

ns	Smart pointer to the namespace interface

Returns

bool True

The documentation for this class was generated from the following file:

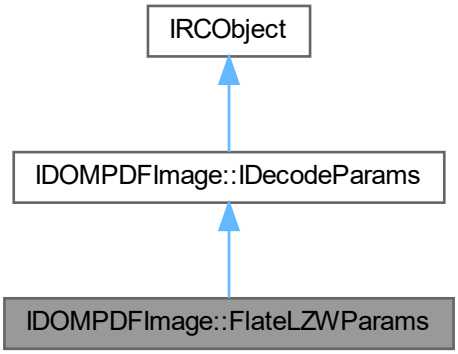
- edlqname.h

7.139 IDOMPDFImage::FlateLZWParams Class Reference

Class to hold filter parameters for Flate or LZW-compressed image data. Please see the PDF specification for the meaning of these parameters.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMPDFImage::FlateLZWParams:



Additional Inherited Members

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()
  Virtual destructor.

7.139.1 Detailed Description

Class to hold filter parameters for Flate or LZW-compressed image data. Please see the PDF specification for the meaning of these parameters.

The documentation for this class was generated from the following file:

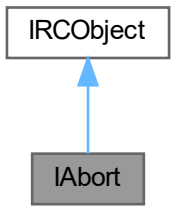
- idomimageresource.h

7.140 IAbort Class Reference

A simple class to signal an abort in processing.

`#include <iabort.h>`

Inheritance diagram for IAbort:



Public Member Functions

- virtual void **resetSignal** ()=0
  Reset the abort object to an unsignalled state.
- virtual void **signalAbort** ()=0
  Signal that the output operation should be aborted.
- virtual bool **abortSignalled** ()=0
  Determine if the abort has been signalled.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0
  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0
  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0
  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static EDL_API IAbortPtr **create** ()

  Create an abort signal.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.140.1 Detailed Description

A simple class to signal an abort in processing.

The documentation for this class was generated from the following file:

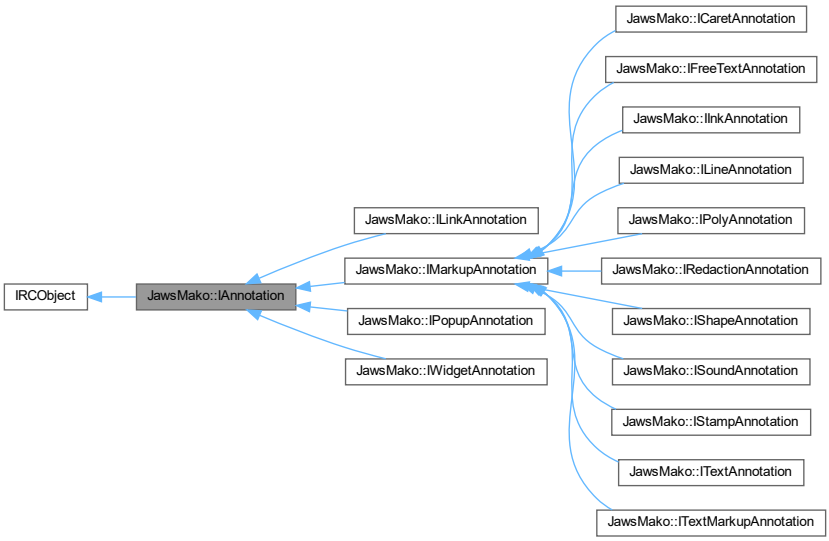
- iabort.h

7.141 JawsMako::IAnnotation Class Reference

An interface class for an annotation.

`#include <interactive.h>`

Inheritance diagram for JawsMako::IAnnotation:



Public Types

- enum eAnnotationType {
  eAT3D , eATCaret , eATCircle , eATFileAttachment ,
  eATFreeText , eATHighlight , eATInk , eATLine ,
  eATLink , eATMovie , eATPolygon , eATPolyLine ,
  eATPopup , eATPrinterMark , eATProjection , eATRedact ,
  eATRichMedia , eATScreen , eATSound , eATSquare ,
  eATSquiggly , eATStamp , eATStrikeOut , eATText ,
  eATTrapNet , eATUnderline , eATWatermark , eATWidget ,
  eATOther }

  Types of annotations, listed with the earliest version of PDF that supported them.

Public Member Functions

- virtual eAnnotationType getType () const =0

  Get the type of the annotation.
- virtual const FRect & getRect () const =0

  Get the rect in which the appearances should be displayed.
- virtual void setRect (const FRect &rect)=0

  Set the rect in which the appearances should be displayed.
- virtual U8String getContents () const =0

  Get the Contents entry in UTF-8.
- virtual void setContents (const U8String &contents)=0

  Set the Contents entry in UTF-8.
- virtual IDOMColorPtr getColor () const =0

  Get the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.
- virtual void setColor (const IDOMColorPtr &color)=0

  Set the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.
- virtual IEDLTimePtr getModificationTime () const =0

  Get the Modification date and time of the annotation, if present.
- virtual void setModificationTime (const IEDLTimePtr &modificationTime)=0

  Set the Modification date and time of the annotation.
- virtual CAnnotationBorder getBorder () const =0

  Get the annotation's border. See CAnnotationBorder for details.
- virtual void setBorder (const CAnnotationBorder &border)=0

  Set the annotation's border.
- virtual uint32 getFlags () const =0

  Get the annotation flags. To interpret these flags please see section 8.4.2 "Annotation Flags" in the PDF 1.7 specification.
- virtual void setFlags (uint32 flags)=0

  Set the annotation flags. Please see section 8.4.2 "Annotation Flags" for details about these flags.
- virtual void rotate (uint16 rotate, double pageWidth, double pageHeight)=0

  Rotate the annotation clockwise as if the page was rotated by the same amount.
- virtual CAnnotationAppearanceVect getAppearances () const =0

  Return all the annotation appearances in a vector.
- virtual void **removeAppearances** ()=0

  Remove all annotation appearances.
- virtual U8String getState () const =0

  Get the current annotation state.
- virtual void setState (const U8String &state)=0

  Set the current annotation state.

- virtual IAnnotationAppearancePtr getAppearance (eAppearanceUsage usage, const U8String &state=U8String()) const =0

  Fetch the annotation appearance that should be used for the given annotation usage and state according to the following rules:
- virtual void addAppearance (const IAnnotationAppearancePtr &appearance)=0

  Add or replace an appearance.
- virtual bool hasNormalAppearance () const =0

  Does the annotation have a normal appearance? Convenience utility function.
- virtual IAnnotationPtr clone () const =0

  Clone the annotation. This is a deep clone. The annotation reference will remain the same.
- virtual bool matchesReference (const IAnnotationReferencePtr &reference) const =0

  Does this annotation match the given IAnnotationReference?
- virtual IAnnotationReferencePtr getReference () const =0

  Get a reference that can be used to refer to this annotation.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.141.1 Detailed Description

An interface class for an annotation.

7.141.2 Member Enumeration Documentation

eAnnotationType

`enum JawsMako::IAnnotation::eAnnotationType`

Types of annotations, listed with the earliest version of PDF that supported them.

Enumerator

eAT3D	(PDF 1.6) 3D annotation
eATCaret	(PDF 1.5) Caret annotation

Enumerator

eATCircle	(PDF 1.3) Circle annotation
eATFileAttachment	(PDF 1.3) File attachment annotation
eATFreeText	(PDF 1.3) Free text annotation
eATHighlight	(PDF 1.3) Highlight annotation
eATInk	(PDF 1.3) Ink annotation
eATLine	(PDF 1.3) Line annotation
eATLink	(PDF 1.2) Link annotation
eATMovie	(PDF 1.2) Movie annotation
eATPolygon	(PDF 1.5) Polygon annotation
eATPolyLine	(PDF 1.5) PolyLine annotation
eATPopup	(PDF 1.3) Popup annotation
eATPrinterMark	(PDF 1.4) Printer's mark annotation
eATProjection	(PDF 1.7) Projection annotation
eATRedact	(PDF 1.7) Redact annotation
eATRichMedia	(PDF 1.7) RichMedia annotation
eATScreen	(PDF 1.5) Screen annotation
eATSound	(PDF 1.2) Sound annotation
eATSquare	(PDF 1.3) Square annotation
eATSquiggly	(PDF 1.4) Squiggly underline annotation
eATStamp	(PDF 1.3) Stamp annotation
eATStrikeOut	(PDF 1.3) Strikeout annotation
eATText	(PDF 1.2) Text annotation
eATTrapNet	(PDF 1.3) Trap network annotation
eATUnderline	(PDF 1.3) Underline annotation
eATWatermark	(PDF 1.6) Watermark annotation
eATWidget	(PDF 1.2) Widget annotation
eATOther	Other annotation.

7.141.3 Member Function Documentation

addAppearance()

```
virtual void JawsMako::IAnnotation::addAppearance (
            const IAnnotationAppearancePtr & appearance )  [pure virtual]
```

Add or replace an appearance.

```
If the insert would result in two or more appearances with the same usage
but different states, then all appearances with the same usage must have
a defined state (an invalid scenario). If not, an exception will be thrown.
```

Parameters

appearance	The annotation appearance

clone()

```
virtual IAnnotationPtr JawsMako::IAnnotation::clone ( ) const  [pure virtual]
```

Clone the annotation. This is a deep clone. The annotation reference will remain the same.

Returns

IAnnotationPtr A smart pointer to the cloned annotation object

getAppearance()

```
virtual IAnnotationAppearancePtr JawsMako::IAnnotation::getAppearance (
            eAppearanceUsage usage,
            const U8String & state = U8String() ) const  [pure virtual]
```

Fetch the annotation appearance that should be used for the given annotation usage and state according to the following rules:

- If an exact match is found, it will be returned.
- If an exact match is not found, but there exists a Normal appearance with the same state, it will be returned.
- If none of the above yield an appearance, an exception results.

Parameters

usage	The annotation usage
state	The annotation state

Returns

IAnnotationAppearancePtr A smart pointer to the annotation appearance object

getAppearances()

```
virtual CAnnotationAppearanceVect JawsMako::IAnnotation::getAppearances ( ) const  [pure virtual]
```

Return all the annotation appearances in a vector.

Returns

CAnnotationAppearanceVect The vector of annotation appearances

getBorder()

```
virtual CAnnotationBorder JawsMako::IAnnotation::getBorder ( ) const  [pure virtual]
```

Get the annotation's border. See CAnnotationBorder for details.

Returns

CAnnotationBorder The annotation's border

getColor()

```
virtual IDOMColorPtr JawsMako::IAnnotation::getColor ( ) const  [pure virtual]
```

Get the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.

Returns

IDOMColorPtr The annotation color

getContents()

```
virtual U8String JawsMako::IAnnotation::getContents ( ) const  [pure virtual]
```

Get the Contents entry in UTF-8.

Returns

U8String The Contents entry, or an empty string if there is no Contents entry.

getFlags()

```
virtual uint32 JawsMako::IAnnotation::getFlags ( ) const  [pure virtual]
```

Get the annotation flags. To interpret these flags please see section 8.4.2 "Annotation Flags" in the PDF 1.7 specification.

Returns

uint32 The annotation flags

getModificationTime()

```
virtual IEDLTimePtr JawsMako::IAnnotation::getModificationTime ( ) const  [pure virtual]
```

Get the Modification date and time of the annotation, if present.

Returns

IEDLTimePtr The modification date. If the date is not of the expected format, or is missing, a NULL object will be returned

getRect()

```
virtual const FRect & JawsMako::IAnnotation::getRect ( ) const  [pure virtual]
```

Get the rect in which the appearances should be displayed.

Returns

FRect The display rect

getReference()

```
virtual IAnnotationReferencePtr JawsMako::IAnnotation::getReference ( ) const  [pure virtual]
```

Get a reference that can be used to refer to this annotation.

Returns

IAnnotationReferencePtr The annotation reference

getState()

```
virtual U8String JawsMako::IAnnotation::getState ( ) const  [pure virtual]
```

Get the current annotation state.

Returns

U8String The current annotation state, or an empty string if there is no state applicable

getType()

```
virtual eAnnotationType JawsMako::IAnnotation::getType ( ) const  [pure virtual]
```

Get the type of the annotation.

Returns

eAnnotationType The annotation type

hasNormalAppearance()

```
virtual bool JawsMako::IAnnotation::hasNormalAppearance ( ) const  [pure virtual]
```

Does the annotation have a normal appearance? Convenience utility function.

Returns

bool True if the annotation has a normal appearance

matchesReference()

```
virtual bool JawsMako::IAnnotation::matchesReference (
            const IAnnotationReferencePtr & reference ) const  [pure virtual]
```

Does this annotation match the given IAnnotationReference?

Parameters

reference	The annotation reference to be matched

Returns

bool True if a match is found, otherwise false

rotate()

```
virtual void JawsMako::IAnnotation::rotate (
             uint16 rotate,
             double pageWidth,
             double pageHeight )  [pure virtual]
```

Rotate the annotation clockwise as if the page was rotated by the same amount.

NB If an annotation has the NoRotate (bit 4) flag set, then the annotation will not be rotated, merely repositioned accordingly.

Parameters

rotate	The angle to rotate. Must be 0, 90, 180 or 270 degrees.
pageWidth	The width of the parent page.
pageHeight	The height of the parent page.

setBorder()

```
virtual void JawsMako::IAnnotation::setBorder (
             const CAnnotationBorder & border )  [pure virtual]
```

Set the annotation's border.

Parameters

border	The border. See CAnnotationBorder for details as not all attributes will be supported for all annotation types and all versions of PDF.

setColor()

```
virtual void JawsMako::IAnnotation::setColor (
             const IDOMColorPtr & color )  [pure virtual]
```

Set the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.

Parameters

color	The required color. Setting a NULL value will remove any existing color. If a color is provided it must use a DeviceRGB, DeviceGray or DeviceCMYK color space.

setContents()

```
virtual void JawsMako::IAnnotation::setContents (
            const U8String & contents )  [pure virtual]
```

Set the Contents entry in UTF-8.

Parameters

contents	The Contents entry

setFlags()

```
virtual void JawsMako::IAnnotation::setFlags (
            uint32 flags )  [pure virtual]
```

Set the annotation flags. Please see section 8.4.2 "Annotation Flags" for details about these flags.

Parameters

flags	The annotation flags

setModificationTime()

```
virtual void JawsMako::IAnnotation::setModificationTime (
            const IEDLTimePtr & modificationTime )  [pure virtual]
```

Set the Modification date and time of the annotation.

Parameters

modificationTime	the Modification date and time. Must not be NULL.

setRect()

```
virtual void JawsMako::IAnnotation::setRect (
            const FRect & rect )  [pure virtual]
```

Set the rect in which the appearances should be displayed.

Parameters

rect	The display rect, which must be non-empty

setState()

```
virtual void JawsMako::IAnnotation::setState (
            const U8String & state )  [pure virtual]
```

Set the current annotation state.

Parameters

state	The annotation state, or an empty string if the state should be removed

The documentation for this class was generated from the following file:

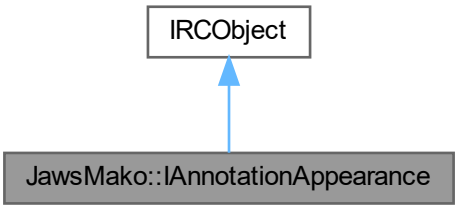
- interactive.h

7.142 JawsMako::IAnnotationAppearance Class Reference

An interface class for an annotation appearance, describing the graphical content of an annotation in a given usage and state. Annotation appearances are immutable.

```
#include <interactive.h>
```

Inheritance diagram for JawsMako::IAnnotationAppearance:



Public Member Functions

- virtual IAnnotationAppearancePtr clone () const =0

  Clone the appearance. This is a deep clone.
- virtual eAppearanceUsage getUsage () const =0

  Get the usage for this appearance.
- virtual U8String getState () const =0

Get the appearance state for this appearance. Will return an empty string if there is no state associated with the appearance.

- virtual IDOMFormPtr getForm () const =0

  Get the graphical contents as an IDOMForm. Do not edit this form.

- virtual IDOMNodePtr getScaledAppearance (const FRect &annotRect) const =0

  Get DOM for the annotation scaled appropriately to the given rect. Useful for preparing an annotation appearance for display or rendering. A clone of the contents will be returned.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.

- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.

- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IAnnotationAppearancePtr create (const IJawsMakoPtr &jawsMako, const IDOMFormPtr &form, eAppearanceUsage usage, const U8String &state=U8String())

  Create a new annotation appearance.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.142.1 Detailed Description

An interface class for an annotation appearance, describing the graphical content of an annotation in a given usage and state. Annotation appearances are immutable.

7.142.2 Member Function Documentation

clone()

```
virtual IAnnotationAppearancePtr JawsMako::IAnnotationAppearance::clone ( ) const  [pure virtual]
```

Clone the appearance. This is a deep clone.

Returns

IAnnotationAppearancePtr A smart pointer to the cloned annotation appearance

create()

```
static JAWSMAKO_API IAnnotationAppearancePtr JawsMako::IAnnotationAppearance::create (
            const IJawsMakoPtr & jawsMako,
            const IDOMFormPtr & form,
            eAppearanceUsage usage,
            const U8String & state = U8String() )  [static]
```

Create a new annotation appearance.

Parameters

jawsMako	The JawsMako instance
form	The graphical contents as an IDOMForm
usage	The usage for this appearance
state	The appearance state for this appearance

Returns

IAnnotationAppearancePtr A smart pointer to the new annotation appearance

getForm()

```
virtual IDOMFormPtr JawsMako::IAnnotationAppearance::getForm ( ) const  [pure virtual]
```

Get the graphical contents as an IDOMForm. Do not edit this form.

Returns

IDOMFormPtr The graphical contents as an IDOMForm

getScaledAppearance()

```
virtual IDOMNodePtr JawsMako::IAnnotationAppearance::getScaledAppearance (
            const FRect & annotRect ) const  [pure virtual]
```

Get DOM for the annotation scaled appropriately to the given rect. Useful for preparing an annotation appearance for display or rendering. A clone of the contents will be returned.

Parameters

annotRect	An FRect that defines the scale of the returned DOM node

Returns

IDOMNodePtr The DOM for the annotation

getState()

```
virtual U8String JawsMako::IAnnotationAppearance::getState ( ) const  [pure virtual]
```

Get the appearance state for this appearance. Will return an empty string if there is no state associated with the appearance.

Returns

U8String The appearance state for this appearance

getUsage()

```
virtual eAppearanceUsage JawsMako::IAnnotationAppearance::getUsage ( ) const  [pure virtual]
```

Get the usage for this appearance.

Returns

eAppearanceUsage The usage for this appearance

The documentation for this class was generated from the following file:

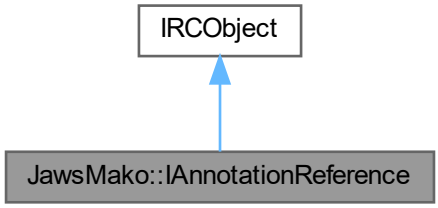
- interactive.h

7.143 JawsMako::IAnnotationReference Class Reference

A generic reference to an annotation. The target annotation might not be loaded. Chiefly used to refer to annotations from a Form.

```
#include <interactive.h>
```

Inheritance diagram for JawsMako::IAnnotationReference:



Public Member Functions

- virtual bool equals (const IAnnotationReferencePtr &other) const =0

  Determine if another annotation reference refers to the same annotation.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()
  Virtual destructor.

7.143.1 Detailed Description

A generic reference to an annotation. The target annotation might not be loaded. Chiefly used to refer to annotations from a Form.

7.143.2 Member Function Documentation

equals()

```
virtual bool JawsMako::IAnnotationReference::equals (
            const IAnnotationReferencePtr & other ) const  [pure virtual]
```

Determine if another annotation reference refers to the same annotation.

Parameters

other	The other annotation reference

Returns

bool True if *other* refers to the same annotation

The documentation for this class was generated from the following file:

- interactive.h

7.144 JawsMako::IAnnotationUtils Class Reference

```
#include <interactive.h>
```

Classes

- class CXMLResource
  Simple class for tracking streams associated with XML generated by generateXMLForDocument()

Static Public Member Functions

- static JAWSMAKO_API IRAInputStreamPtr generateXMLForDocument (const IJawsMakoPtr &jawsMako, const IDocumentPtr &document, CXMLResourceVect &resources)
  Generate an XML version of the Annotations and Forms content in the given IDocument.
- static JAWSMAKO_API IAnnotationReferencePtr createAnnotationReferenceFromTag (const U8String &tag)
  Generate an IAnnotationReference from an Annotation Tag attribute found in the XML.

7.144.1 Detailed Description

Utility class for dealing with XML representations of annotations and forms.

7.144.2 Member Function Documentation

createAnnotationReferenceFromTag()

```
static JAWSMAKO_API IAnnotationReferencePtr JawsMako::IAnnotationUtils::createAnnotation↩
ReferenceFromTag (
            const U8String & tag )  [static]
```

Generate an IAnnotationReference from an Annotation Tag attribute found in the XML.

Parameters

tag	Annotation Tag

Returns

IAnnotationReferencePtr A smart pointer to the new annotation reference

generateXMLForDocument()

```
static JAWSMAKO_API IRAInputStreamPtr JawsMako::IAnnotationUtils::generateXMLForDocument (
            const IJawsMakoPtr & jawsMako,
            const IDocumentPtr & document,
            CXMLResourceVect & resources )  [static]
```

Generate an XML version of the Annotations and Forms content in the given IDocument.

Parameters

jawsMako	The JawsMako instance
document	The document to generate the XML stream for
resources	A reference to a vector to receive any streams that hold sounds, embedded files, or other embedded streams referenced from annotations

Returns

IRAInputStreamPtr A smart pointer to a stream containing the XML representation of the Annotations and Form content

The documentation for this class was generated from the following file:

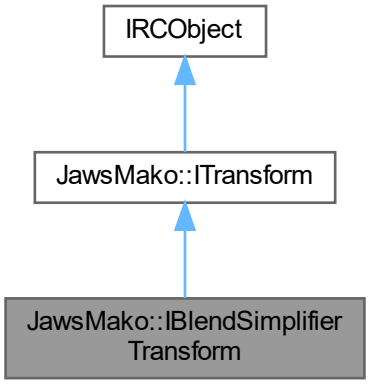
- interactive.h

7.145 JawsMako::IBlendSimplifierTransform Class Reference

A transform to transform linear or radial gradients with repeat or reflect padding to simpler types. For linear gradients this means a simple linear gradient inside a tiling visual brush. For radial gradients, the stops must be repeated as required.

`#include <transforms.h>`

Inheritance diagram for JawsMako::IBlendSimplifierTransform:



Static Public Member Functions

- static JAWSMAKO_API IBlendSimplifierTransformPtr create (const IJawsMakoPtr &jawsMako, const IAbort↩ Ptr &abort=IAbortPtr())

  Create the transform.

Additional Inherited Members

Public Member Functions inherited from JawsMako::ITransform

- virtual IDOMBrushPtr transform (const IDOMBrushPtr &brush, eBrushUsage usage=eBUGeneral, const CTransformState &state=CTransformState())=0

  Apply the transform to the given brush, if applicable. These transforms are thread safe.
- virtual IDOMImagePtr transform (const IDOMImagePtr &image, const CTransformState &state=CTransformState())=0

  Apply the transform to the given image, if applicable. These transforms are thread safe.
- virtual IDOMColorPtr transform (const IDOMColorPtr &color, const CTransformState &state=CTransformState())=0

  Apply the transform to the given color, if applicable. These transforms are thread safe.
- virtual IDOMColorSpacePtr transform (const IDOMColorSpacePtr &colorSpace, const CTransformState &state=CTransformState())=0

  Apply the transform to the given colorspace, if applicable. These transforms are thread safe.
- virtual IDOMNodePtr transform (const IDOMNodePtr &node, bool &changed, bool transformChildren=true, const CTransformState &state=CTransformState())=0

Apply the transform to the given node, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time.

- virtual void transformPage (const IPagePtr &page, bool transformContent=true, bool transform↩ Annotations=true)=0

  Apply the transform to the given page, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time. The transform will also apply to the annotations appearances.

- virtual void **flushCaches** ()=0

  Flush the caches used by the transform. Most transforms cache recently transformed results to improve the performance of repeated transformations of equivalent results. However, it is possible that some cached results may point to entities that no longer exist, such as content inside an XPS file that no longer exists. If you are deleting or replacing files where transforms have been used, it is advisable to invoke this routine to clear the caches.

- virtual void setProgressMonitor (const IProgressMonitorPtr &progressMonitor)=0

  Set the IProgressMonitor object for this transform to allow for monitoring the progress of the transform.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.

- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.

- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.145.1 Detailed Description

A transform to transform linear or radial gradients with repeat or reflect padding to simpler types. For linear gradients this means a simple linear gradient inside a tiling visual brush. For radial gradients, the stops must be repeated as required.

7.145.2 Member Function Documentation

create()

```
static JAWSMAKO_API IBlendSimplifierTransformPtr JawsMako::IBlendSimplifierTransform::create (
            const IJawsMakoPtr & jawsMako,
            const IAbortPtr & abort = IAbortPtr() )  [static]
```

Create the transform.

Parameters

jawsMako	The JawsMako instance.
abort	An abort callback handler.

Returns

The new instance.

The documentation for this class was generated from the following file:

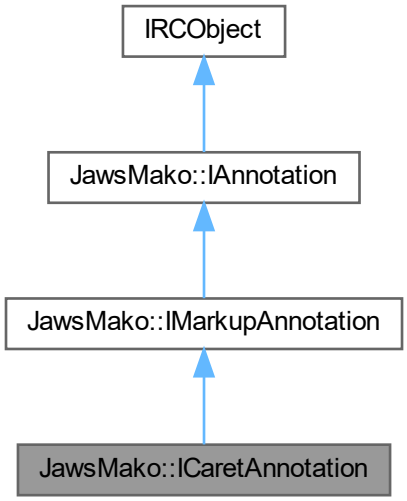
- transforms.h

7.146 JawsMako::ICaretAnnotation Class Reference

A generic interface class for a caret annotation It is intended that future releases of JawsMako will extend this interface.

```
#include <interactive.h>
```

Inheritance diagram for JawsMako::ICaretAnnotation:



Public Member Functions

- virtual CRectInset getRectInset () const =0

  Get the rect inset describing, relative to the annotation rect, the caret area within the annotation.

Public Member Functions inherited from JawsMako::IMarkupAnnotation

- virtual U8String getAuthor () const =0

  Get the Author of the markup annotation. This is the "T" entry in the annotation, which by convention is the author.
- virtual void setAuthor (const U8String &author)=0

  Set the Author of the markup annotation. This is the "T" entry in the annotation, which by convention is the author.
- virtual IEDLTimePtr getCreationTime () const =0

  Get the creation date and time of the annotation, if present.
- virtual void setCreationTime (const IEDLTimePtr &creationTime)=0

  Set the creation date and time of the annotation.
- virtual float getOpacity () const =0

  Get the opacity of the markup annotation.
- virtual void setOpacity (float opacity)=0

  Set the opacity of the markup annotation.
- virtual IAnnotationReferencePtr getPopupReference () const =0

  Get a reference to the popup, if present.
- virtual void setPopup (const IAnnotationReferencePtr &popupReference)=0

  Set or clear the popup, by reference.
- virtual void setPopup (const IPopupAnnotationPtr &popup)=0

  Set a reference to a popup, if present.
- virtual IAnnotationAppearancePtr createNormalAppearance () const =0

  Create a basic appearance given the current annotation's parameters. This can then be installed by using IAnnotation::addAppearance().

Public Member Functions inherited from JawsMako::IAnnotation

- virtual eAnnotationType getType () const =0

  Get the type of the annotation.
- virtual const FRect & getRect () const =0

  Get the rect in which the appearances should be displayed.
- virtual void setRect (const FRect &rect)=0

  Set the rect in which the appearances should be displayed.
- virtual U8String getContents () const =0

  Get the Contents entry in UTF-8.
- virtual void setContents (const U8String &contents)=0

  Set the Contents entry in UTF-8.
- virtual IDOMColorPtr getColor () const =0

  Get the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.
- virtual void setColor (const IDOMColorPtr &color)=0

  Set the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.
- virtual IEDLTimePtr getModificationTime () const =0

  Get the Modification date and time of the annotation, if present.
- virtual void setModificationTime (const IEDLTimePtr &modificationTime)=0

  Set the Modification date and time of the annotation.
- virtual CAnnotationBorder getBorder () const =0

  Get the annotation's border. See CAnnotationBorder for details.
- virtual void setBorder (const CAnnotationBorder &border)=0

  Set the annotation's border.
- virtual uint32 getFlags () const =0

  Get the annotation flags. To interpret these flags please see section 8.4.2 "Annotation Flags" in the PDF 1.7 specification.

- virtual void setFlags (uint32 flags)=0

  Set the annotation flags. Please see section 8.4.2 "Annotation Flags" for details about these flags.
- virtual void rotate (uint16 rotate, double pageWidth, double pageHeight)=0

  Rotate the annotation clockwise as if the page was rotated by the same amount.
- virtual CAnnotationAppearanceVect getAppearances () const =0

  Return all the annotation appearances in a vector.
- virtual void **removeAppearances** ()=0

  Remove all annotation appearances.
- virtual U8String getState () const =0

  Get the current annotation state.
- virtual void setState (const U8String &state)=0

  Set the current annotation state.
- virtual IAnnotationAppearancePtr getAppearance (eAppearanceUsage usage, const U8String &state=U8String()) const =0

  Fetch the annotation appearance that should be used for the given annotation usage and state according to the following rules:
- virtual void addAppearance (const IAnnotationAppearancePtr &appearance)=0

  Add or replace an appearance.
- virtual bool hasNormalAppearance () const =0

  Does the annotation have a normal appearance? Convenience utility function.
- virtual IAnnotationPtr clone () const =0

  Clone the annotation. This is a deep clone. The annotation reference will remain the same.
- virtual bool matchesReference (const IAnnotationReferencePtr &reference) const =0

  Does this annotation match the given IAnnotationReference?
- virtual IAnnotationReferencePtr getReference () const =0

  Get a reference that can be used to refer to this annotation.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Public Types inherited from JawsMako::IAnnotation

- enum eAnnotationType {
  eAT3D , eATCaret , eATCircle , eATFileAttachment ,
  eATFreeText , eATHighlight , eATInk , eATLine ,
  eATLink , eATMovie , eATPolygon , eATPolyLine ,
  eATPopup , eATPrinterMark , eATProjection , eATRedact ,
  eATRichMedia , eATScreen , eATSound , eATSquare ,
  eATSquiggly , eATStamp , eATStrikeOut , eATText ,
  eATTrapNet , eATUnderline , eATWatermark , eATWidget ,
  eATOther }

  Types of annotations, listed with the earliest version of PDF that supported them.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.146.1 Detailed Description

A generic interface class for a caret annotation It is intended that future releases of JawsMako will extend this interface.

7.146.2 Member Function Documentation

getRectInset()

```
virtual CRectInset JawsMako::ICaretAnnotation::getRectInset ( ) const  [pure virtual]
```

Get the rect inset describing, relative to the annotation rect, the caret area within the annotation.

Returns

CRectInset The rect inset

The documentation for this class was generated from the following file:

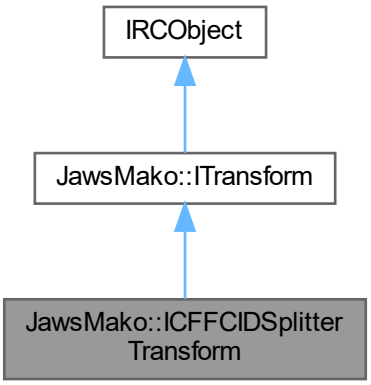
- interactive.h

7.147 JawsMako::ICFFCIDSplitterTransform Class Reference

A simple transform that looks for CID CFF Fonts containing multiple SubFonts. Some viewers do not support these fonts, or do so poorly. If found, this transform will split out the sub fonts into individual font streams, and adjust the Glyphs nodes where they are used accordingly.

```
#include <transforms.h>
```

Inheritance diagram for JawsMako::ICFFCIDSplitterTransform:



Static Public Member Functions

- static JAWSMAKO_API ICFFCIDSplitterTransformPtr create (const IJawsMakoPtr &jawsMako, const IAbort↩ Ptr &abort=IAbortPtr())

  Create the transform.

Additional Inherited Members

Public Member Functions inherited from JawsMako::ITransform

- virtual IDOMBrushPtr transform (const IDOMBrushPtr &brush, eBrushUsage usage=eBUGeneral, const CTransformState &state=CTransformState())=0

  Apply the transform to the given brush, if applicable. These transforms are thread safe.
- virtual IDOMImagePtr transform (const IDOMImagePtr &image, const CTransformState &state=CTransformState())=0

  Apply the transform to the given image, if applicable. These transforms are thread safe.
- virtual IDOMColorPtr transform (const IDOMColorPtr &color, const CTransformState &state=CTransformState())=0

  Apply the transform to the given color, if applicable. These transforms are thread safe.
- virtual IDOMColorSpacePtr transform (const IDOMColorSpacePtr &colorSpace, const CTransformState &state=CTransformState())=0

  Apply the transform to the given colorspace, if applicable. These transforms are thread safe.
- virtual IDOMNodePtr transform (const IDOMNodePtr &node, bool &changed, bool transformChildren=true, const CTransformState &state=CTransformState())=0

  Apply the transform to the given node, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time.
- virtual void transformPage (const IPagePtr &page, bool transformContent=true, bool transform↩ Annotations=true)=0

  Apply the transform to the given page, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time. The transform will also apply to the annotations appearances.
- virtual void **flushCaches** ()=0

  Flush the caches used by the transform. Most transforms cache recently transformed results to improve the performance of repeated transformations of equivalent results. However, it is possible that some cached results may point to entities that no longer exist, such as content inside an XPS file that no longer exists. If you are deleting or replacing files where transforms have been used, it is advisable to invoke this routine to clear the caches.
- virtual void setProgressMonitor (const IProgressMonitorPtr &progressMonitor)=0

  Set the IProgressMonitor object for this transform to allow for monitoring the progress of the transform.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.147.1 Detailed Description

A simple transform that looks for CID CFF Fonts containing multiple SubFonts. Some viewers do not support these fonts, or do so poorly. If found, this transform will split out the sub fonts into individual font streams, and adjust the Glyphs nodes where they are used accordingly.

Operates on nodes. Note that when operating on a glyphs node, the operation may result in multiple nodes, in which case they will be returned encapsulated in an IDOMGroup or IDOMTransparencyGroup depending on its attributes.

Useful whenever a CID CFF with multiple subfonts would not be allowed.

7.147.2 Member Function Documentation

create()

```
static JAWSMAKO_API ICFFCIDSplitterTransformPtr JawsMako::ICFFCIDSplitterTransform::create (
            const IJawsMakoPtr & jawsMako,
            const IAbortPtr & abort = IAbortPtr() )  [static]
```

Create the transform.

Parameters

jawsMako	The JawsMako instance.
abort	An abort callback handler.

Returns

The new instance.

The documentation for this class was generated from the following file:

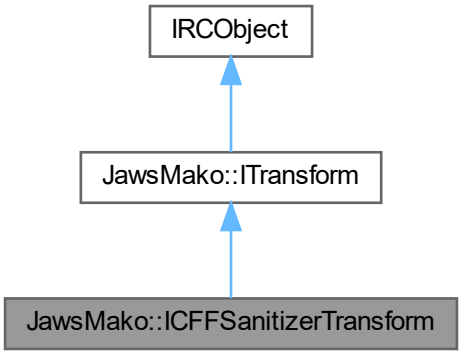
- transforms.h

7.148 JawsMako::ICFFSanitizerTransform Class Reference

A simple transform that scans and sanitises CFF Fonts for wider interoperability.

```
#include <transforms.h>
```

Inheritance diagram for JawsMako::ICFFSanitizerTransform:



Static Public Member Functions

- static JAWSMAKO_API ICFFSanitizerTransformPtr create (const IJawsMakoPtr &jawsMako, const IAbortPtr &abort=IAbortPtr())

  Create the transform.

Additional Inherited Members

Public Member Functions inherited from JawsMako::ITransform

- virtual IDOMBrushPtr transform (const IDOMBrushPtr &brush, eBrushUsage usage=eBUGeneral, const CTransformState &state=CTransformState())=0

  Apply the transform to the given brush, if applicable. These transforms are thread safe.
- virtual IDOMImagePtr transform (const IDOMImagePtr &image, const CTransformState &state=CTransformState())=0

  Apply the transform to the given image, if applicable. These transforms are thread safe.
- virtual IDOMColorPtr transform (const IDOMColorPtr &color, const CTransformState &state=CTransformState())=0

  Apply the transform to the given color, if applicable. These transforms are thread safe.
- virtual IDOMColorSpacePtr transform (const IDOMColorSpacePtr &colorSpace, const CTransformState &state=CTransformState())=0

  Apply the transform to the given colorspace, if applicable. These transforms are thread safe.
- virtual IDOMNodePtr transform (const IDOMNodePtr &node, bool &changed, bool transformChildren=true, const CTransformState &state=CTransformState())=0

  Apply the transform to the given node, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time.
- virtual void transformPage (const IPagePtr &page, bool transformContent=true, bool transform↩Annotations=true)=0

  Apply the transform to the given page, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time. The transform will also apply to the annotations appearances.
- virtual void **flushCaches** ()=0

  Flush the caches used by the transform. Most transforms cache recently transformed results to improve the performance of repeated transformations of equivalent results. However, it is possible that some cached results may point to entities that no longer exist, such as content inside an XPS file that no longer exists. If you are deleting or replacing files where transforms have been used, it is advisable to invoke this routine to clear the caches.
- virtual void setProgressMonitor (const IProgressMonitorPtr &progressMonitor)=0

  Set the IProgressMonitor object for this transform to allow for monitoring the progress of the transform.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.148.1 Detailed Description

A simple transform that scans and sanitises CFF Fonts for wider interoperability.

Some consumers and operating systems may have problems with certain CFF font structures, which this transform seeks to normalise.

Useful for cases where a CFF based OpenType font may need to be used with unknown downstream viewiers in XPS, SVG, etc.

7.148.2 Member Function Documentation

create()

```
static JAWSMAKO_API ICFFSanitizerTransformPtr JawsMako::ICFFSanitizerTransform::create (
            const IJawsMakoPtr & jawsMako,
            const IAbortPtr & abort = IAbortPtr() )  [static]
```

Create the transform.

Parameters

jawsMako	The JawsMako instance.
abort	An abort callback handler.

Returns

The new instance.

The documentation for this class was generated from the following file:

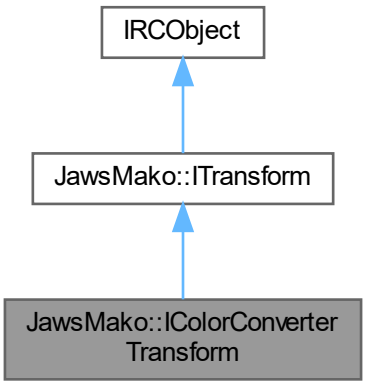
- transforms.h

7.149 JawsMako::IColorConverterTransform Class Reference

A transform for color conversion, converting all appropriate DOM contents to a desired target color space.

`#include <transforms.h>`

Inheritance diagram for JawsMako::IColorConverterTransform:



Public Member Functions

- virtual void setTargetSpace (const IDOMColorSpacePtr &targetSpace)=0

  Set the desired target space for conversion. The default is DeviceRGB. Indexed or DeviceN color spaces cannot be used as a target space.
- virtual IDOMColorSpacePtr **getTargetSpace** ()=0

  Get the target space for conversion.
- virtual void setTargetProfile (const IDOMICCProfilePtr &profile)=0

  Set the desired target space using an ICC profile.
- virtual void **setSWOPTargetSpace** ()=0

  Set the desired target space to the default SWOP color profile.
- virtual void **setOverrideRenderingIntent** (eRenderingIntent intent)=0

  Sets an override rendering intent. By default this is not set. If set, this intent is used instead of any DOM resident information.
- virtual void **useDOMRenderingIntents** (bool use)=0

  Set whether or not to use explicit rendering intents in the DOM when performing conversion. The PDF input for example can set explicit rendering intents for individual nodes in the DOM, and these will be ignored if this property is set to false. The default is true; that is to honour the explicit rendering intent information in the DOM. This is ignored if an overriding rendering intent has been set by setOverrideRenderingIntent();.
- virtual void setBlackPreservation (bool preserveForText, bool preserveForOther)=0

  Enable/control 100% black preservation.
- virtual void **setConvertOnlyDeviceIndependentColors** (bool convert)=0

  Sets whether or not only device-independent colors should be converted. Useful for cases where device independent color spaces are not allowed, or will not produce reliable results. The default is false.
- virtual void **setConvertIndependentToSimilarDeviceSpace** (bool convert)=0

If setConvertOnlyDeviceIndependentColors() has been set to true, this controls the conversion behaviour for affected spaces. If set to true, the converter will convert to the most "similar" Device space; for example 3 channel colors will be converted to DeviceRGB and so forth. If false, the target space will be used. The default is false.

- virtual void **setInspectDeviceNAlternateSpace** (bool inspect)=0

  If setShouldConvertForSpaceType() sets IDOMColorSpace::eDeviceN to false then normally it will not do color conversion. However if setInspectDeviceNAlternateSpace(true) was invoked then the alternate colorspace is also inspected to see if it will trigger a color conversion. The default is false.

- virtual void **setInspectDeviceNColorantsAlternateSpace** (bool inspect)=0

  If setInspectDeviceNAlternateSpace() is true and applies, and this parameter is set to true, then the alternate colorspace(s) in any colorant(s) (if present) are all inspected to see if they would trigger a color conversion. If false, the colorant(s) colorspace(s) are not inspected. The default is true.

- virtual void **setKeepDeviceN** (bool keep)=0

  If a DeviceN color results in a color conversion into the target space, setting this to true results in reencapsulating the transformed color into a DeviceN color. The original colorant names are reused but the alternate colorspace is now the target colorspace. This is not possible for multitone DeviceN color spaces in which case this setting will be ignored. The default is false.

- virtual void **setIgnoreImagesWithSimilarColorSpaces** (bool ignore)=0

  Sets whether or not images can be ignored if their color space is deemed to be similar to the target space (see IDOMColorSpace::similar() for details about the similarity test). This is useful to avoiding needing to be re-encode images in some situations. The default is false.

- virtual void **setIgnoreRenderedImages** (bool ignore)=0

  Sets whether or not images can be ignored if they are the result of the IRendererTransform. Typically such images are already converted to an eventual target space, and as such should not be transformed, particularly to any page group color space. The default is false.

- virtual void **setForceImageSampleConversion** (bool force)=0

  Sets whether or not color conversion of image samples should be forced, even for cases where the source image color space is considered similar to the target space. The default is false; that is, if an image's color space is similar to the target then the individual color samples will not be color converted. In both cases the target color space will be applied to the resulting image.

- virtual void **setShouldConvertForSpaceType** (IDOMColorSpace::eColorSpaceType type, bool convert=true)=0

  Set whether or not objects using a certain color space type should be converted. The default for all is true, although use of setConvertOnlyDeviceIndependentColors() and setInspectDeviceNAlternateSpace() may affect this.

- virtual bool **wouldConvertForSpaceType** (IDOMColorSpace::eColorSpaceType type)=0

  Get whether or not objects using a certain color space type would be converted.

- virtual void **setConvertICCColorsWithNumComponents** (uint8 numComponents, bool convert=true)=0

  Set whether objects with ICC color spaces with the given number of components should be converted. The default is true for all. numComponents is from 1 to EDL_CMM_MAX_COMPONENTS Calling setShouldConvertForSpace↩ Type(IDOMColorSpace::eICCBased, true) however will cause all possibilities to be true. Likewise calling setShould↩ ConvertForSpaceType(IDOMColorSpace::eICCBased, false) will cause all possibilities to be false.

- virtual void **setConvertICCSpacesWithProfileVersionGreaterThan** (uint8 majorVersion, uint8 minor↩ Version)=0

  Set the maximum allowed version of ICC Profiles allowed. Anything with a higher version will be converted to the target space. The default is 0.0, which disables this feature.

- virtual void setConvertColorsInsideLuminositySoftMasks (bool convert=true)=0

  Set whether the contents of Luminosity Soft Masks should be converted or not.

- virtual void setConvertThroughTransparencyGroupColorSpaces (bool useGroups=true)=0

  Set whether or not the page group and isolated transparency group color spaces should be used when converting.

- virtual void setKeepOverprintMode (bool keep=true)=0

  Set whether or not to drop the overprint mode when converting color spaces.

- virtual void setDeviceNHandling (bool convertProcess=true, bool mergeSpots=true, const IDOMColor↩ SpaceDeviceN::CColorantInfoVect &spotsToMerge=IDOMColorSpaceDeviceN::CColorantInfoVect(), const CSpotColorNames &spotsToRetain=CSpotColorNames(), const CSpotColorNames &spotsToDrop=CSpot↩ ColorNames(), bool knockoutDroppedSpots=true)=0

  Control handling of DeviceN color spaces.

Public Member Functions inherited from JawsMako::ITransform

- virtual IDOMBrushPtr transform (const IDOMBrushPtr &brush, eBrushUsage usage=eBUGeneral, const CTransformState &state=CTransformState())=0

  Apply the transform to the given brush, if applicable. These transforms are thread safe.
- virtual IDOMImagePtr transform (const IDOMImagePtr &image, const CTransformState &state=CTransformState())=0

  Apply the transform to the given image, if applicable. These transforms are thread safe.
- virtual IDOMColorPtr transform (const IDOMColorPtr &color, const CTransformState &state=CTransformState())=0

  Apply the transform to the given color, if applicable. These transforms are thread safe.
- virtual IDOMColorSpacePtr transform (const IDOMColorSpacePtr &colorSpace, const CTransformState &state=CTransformState())=0

  Apply the transform to the given colorspace, if applicable. These transforms are thread safe.
- virtual IDOMNodePtr transform (const IDOMNodePtr &node, bool &changed, bool transformChildren=true, const CTransformState &state=CTransformState())=0

  Apply the transform to the given node, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time.
- virtual void transformPage (const IPagePtr &page, bool transformContent=true, bool transform↩ Annotations=true)=0

  Apply the transform to the given page, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time. The transform will also apply to the annotations appearances.
- virtual void **flushCaches** ()=0

  Flush the caches used by the transform. Most transforms cache recently transformed results to improve the performance of repeated transformations of equivalent results. However, it is possible that some cached results may point to entities that no longer exist, such as content inside an XPS file that no longer exists. If you are deleting or replacing files where transforms have been used, it is advisable to invoke this routine to clear the caches.
- virtual void setProgressMonitor (const IProgressMonitorPtr &progressMonitor)=0

  Set the IProgressMonitor object for this transform to allow for monitoring the progress of the transform.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IColorConverterTransformPtr create (const IJawsMakoPtr &jawsMako, const IAbort↩ Ptr &abort=IAbortPtr())

  Create the transform.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.149.1 Detailed Description

A transform for color conversion, converting all appropriate DOM contents to a desired target color space.

Conversion for individual color space types can be enabled or disabled. This is useful for example for consumers who only support a subset of the JawsMako supported color space set.

When handling objects using indexed color spaces, instances of this transform may opt to use a modified indexed color space rather than actually convert the underlying color. This is usually much faster. An IComplexColorSimplifierTransform can be applied after this transform if required.

7.149.2 Member Function Documentation

create()

```
static JAWSMAKO_API IColorConverterTransformPtr JawsMako::IColorConverterTransform::create (
            const IJawsMakoPtr & jawsMako,
            const IAbortPtr & abort = IAbortPtr() )  [static]
```

Create the transform.

Parameters

jawsMako	The JawsMako instance.

Returns

The new instance.

setBlackPreservation()

```
virtual void JawsMako::IColorConverterTransform::setBlackPreservation (
            bool preserveForText,
            bool preserveForOther )  [pure virtual]
```

Enable/control 100% black preservation.

```
100% black preservation preserves a 100% black output when color
converting when enabled for certain solid colors.

A 100% black object is an object painted with one of the following
colors:

- 0 0 0 1 in any CMYK color space (including ICC)
- 0 0 0 in any RGB color space (including ICC)
- 0 in any Gray colorspace (including ICC)
- 1 in a single channel DeviceN (aka Separation) Black color space
- 1 in a multi-channel DeviceN colorspace where only the Black channel
  is set.

If black preservation applies then the resulting color will be either
(depending on the target color space):

- c m y 1 in any CMYK colorspace (including ICC)
```

```
- 0 0 0 in any RGB colorspace (including ICC)
- 0 in any Gray colorspace (including ICC)

Note that the CMY values in a CMYK result may not be zero if color
managed white conversion would not produce pure white.

Note also that where transparency is involved this conversion only
applies when converting to the blending colorspace, and has no
further effect for further conversions.

May be enabled separately for text and vector art. Applies to only
solid color brushes (IDOMSolidColorBrush) and uncolored tiling
patterns only.

The default is set by #IColorManager::setDefaultBlackPreservation(),
which defaults to false, false. The default is applied at the time
of transform creation.
```

Parameters

preserveForText	Whether or not to apply black preservation for text.
preserveForOther	Whether or not to apply for non-text objects.

setConvertColorsInsideLuminositySoftMasks()

```
virtual void JawsMako::IColorConverterTransform::setConvertColorsInsideLuminositySoftMasks (
            bool convert = true )  [pure virtual]
```

Set whether the contents of Luminosity Soft Masks should be converted or not.

```
The default is true.

Luminosity soft masks use the luminosity of the contents of a soft mask to
generate the opacity that is applied to the objects being masked. If the
colors inside such a soft mask are converted to a different color space,
then this will affect the calculation of the opacity, and potentially
undesirable results in some contexts.
```

setConvertThroughTransparencyGroupColorSpaces()

```
virtual void JawsMako::IColorConverterTransform::setConvertThroughTransparencyGroupColorSpaces
(
            bool useGroups = true )  [pure virtual]
```

Set whether or not the page group and isolated transparency group color spaces should be used when converting.

The default is true. That is, if a color is to be converted, it will be converted through the chain of enclosing transparency group color spaces before final conversion to the target space. If a page group with a colorspace is present however it will only be used if transparency is present (or if transparency was present and has been flattened).

When false, the page group and isolated transparency group color spaces are ignored.

setDeviceNHandling()

```
virtual void JawsMako::IColorConverterTransform::setDeviceNHandling (
            bool convertProcess = true,
            bool mergeSpots = true,
            const IDOMColorSpaceDeviceN::CColorantInfoVect & spotsToMerge = IDOMColorSpaceDeviceN↩
::CColorantInfoVect(),
            const CSpotColorNames & spotsToRetain = CSpotColorNames(),
            const CSpotColorNames & spotsToDrop = CSpotColorNames(),
            bool knockoutDroppedSpots = true )  [pure virtual]
```

Control handling of DeviceN color spaces.

```
    This API allows the following handling for objects in a DeviceN color space
    (or a combination thereof).

    - Merging of one or more spot components into the output process color space.
      If a spot is to be merged, the spot will be merged using a transparency-like
      blending method.
    - The dropping of one or more spot components entirely.
    - Color conversion of process components while retaining a DeviceN process.

    If enabled, any other DeviceN handling directions provided to this transform
    (such as #setKeepDeviceN, #setShouldConvertForSpaceType(IDOMColorSpace::eDeviceN)
    will be ignored, with the exception of DeviceN color spaces that are detected
    as multitone (eg special Duotone color spaces).

    Currently, this feature is only compatible with CMYK target color spaces.
    An error will result should a non-CMYK target color space be used with this
    feature enabled.

    If a DeviceN object requires processing, any /None colorants are dropped,
    and any /All colorants are preserved in order to preserve their special
    behavior.

    The default is off for all features.
```

Parameters

convertProcess	Ignored if mergeSpots is true (where process components will always be converted if present). This parameter exists for the use case where no spots are required to be merged or dropped, but where the process components must be color converted.
mergeSpots	If false, no spots will be merged (and #spotsToMerge is ignored).
spotsToMerge	The spots to merge. Consulted only if #mergeSpots is true. If empty, all spots will be merged unless they are explicitly dropped or retained, and the color of each component will be determined using a best effort method from the DeviceN color space of the content being processed. If non-empty, then the given spots are merged into the process components. The components in the info structures must use the same color space as the target color space.
spotsToRetain	The names of colorants to retain even if they are subject to mergeSpots and/or spotsToMerge.
spotsToDrop	The names of colorants to drop entirely. If not provided, no spot components will be dropped.
knockoutDroppedSpots	Controls behaviour if a non-overprinting object is being dropped due to all its components being dropped. Consider for example, an item being painted with only the spot component "Taupe" and "Taupe" is specified in the spotsToDrop list. If this object were to be dropped entirely, then the objects underneath would not be knocked out as expected. If knockoutDroppedSpots is true, the dropped brush is replaced with white in order to preserve this knockout behaviour.

setKeepOverprintMode()

```
virtual void JawsMako::IColorConverterTransform::setKeepOverprintMode (
            bool keep = true )  [pure virtual]
```

Set whether or not to drop the overprint mode when converting color spaces.

```
    The default is true, which will preserve the overprint mode when converting.
```

setTargetProfile()

```
virtual void JawsMako::IColorConverterTransform::setTargetProfile (
            const IDOMICCProfilePtr & profile )  [pure virtual]
```

Set the desired target space using an ICC profile.

Parameters

profile	The ICC profile to use.

setTargetSpace()

```
virtual void JawsMako::IColorConverterTransform::setTargetSpace (
            const IDOMColorSpacePtr & targetSpace )  [pure virtual]
```

Set the desired target space for conversion. The default is DeviceRGB. Indexed or DeviceN color spaces cannot be used as a target space.

Parameters

targetSpace	The target color space to be used.

The documentation for this class was generated from the following file:

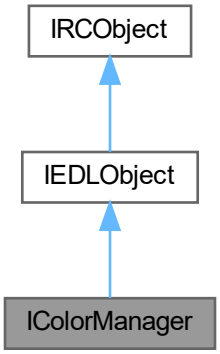
- transforms.h

7.150 IColorManager Class Reference

Public interface to the EDL color manager. There is only one instance of the color manager for each factory. It can be retrieved using the IEDLFactory::getSingleton method, or by using the get() static function.

```
#include <icolormanager.h>
```

Inheritance diagram for IColorManager:



Public Member Functions

- virtual void convertColors (int32 numColors, bool hasAlpha, const IDOMColorSpacePtr &sourceSpace, const IDOMColorSpacePtr &destSpace, eRenderingIntent intent, eBlackPointCompensation bpc, const float ∗in↩ Colors, float ∗outColors)=0

  Convert colors from one color space to another, using floating point samples.
- virtual void convertColors (int32 numColors, bool hasAlpha, const IDOMColorSpacePtr &sourceSpace, const IDOMColorSpacePtr &destSpace, eRenderingIntent intent, eBlackPointCompensation bpc, const uint8 ∗in↩ Colors, uint8 ∗outColors, int32 inBPS, int32 outBPS)=0

  Convert colors from one color space to another, using integer samples. Currently, only base color spaces are supported; Indexed and DeviceN spaces cannot be used. For conversion of multiple samples, samples are interleaved, with any alpha being last. For example, sRGB with alpha must be presented as r, g, b, a, r, g, b, a, etc.
- virtual uint8 getNumComponentsForICCBasedSpace (const IDOMColorSpaceICCBasedPtr &space)=0

  Find the number of components for an ICCBased color space.
- virtual uint8 getNumComponentsForICCProfile (const IDOMICCProfilePtr &profile)=0

  Find the number of components for an ICC profile.
- virtual eRenderingIntent getDefaultIntentForICCBasedSpace (const IDOMColorSpaceICCBasedPtr &space)=0

  Find the default rendering intent for an ICCBased color space.
- virtual uint32 getProfileColorSpaceForICCBasedSpace (const IDOMColorSpaceICCBasedPtr &space)=0

  Get the color space type for an ICCBased color space.
- virtual void getProfileVersionForICCBasedSpace (const IDOMColorSpaceICCBasedPtr &space, uint8 &majorVersion, uint8 &minorVersion)=0

  Get the version number of the given ICC profile.
- virtual void getProfileNameForICCBasedSpace (const IDOMColorSpaceICCBasedPtr &space, EDLSysString &name)=0

  Get the name of the profile for an ICCBased color space.
- virtual CEDLSysStringVect **getColorantNamesForICCBasedSpace** (const IDOMColorSpaceICCBasedPtr &space)=0

  Retrieve the colorant names of an ICC based color space. For simple Gray, RGB or CMYK ICC profiles, the standard process names will be returned. For multichannel profiles, the 'clrt' components will be queried. An IEDLError exception with error code EDL_ERR_NO_COLOR_NAME will be thrown if no name could be determined.
- virtual IDOMICCProfilePtr getCMYKSWOPProfile ()=0

Retrieve the built-in CMYK SWOP profile.

- virtual IDOMICCProfilePtr getsGrayProfile ()=0

  Retrieve the built-in sGray profile.

- virtual IDOMICCProfilePtr getsRGBProfile ()=0

  Retrieve the built-in sRGB profile.

- virtual IDOMICCProfilePtr getscRGBProfile ()=0

  Retrieve the built in scRGB profile.

- virtual IDOMICCProfilePtr createCalibratedGrayProfile (const CEDLVector< double > &whitePoint, double gamma=1.0, const CEDLVector< double > &blackPoint=CEDLVector< double >())=0

  Create a profile for a calibrated gray color space with the given parameters. Such a profile will be analogous to the PDF CalGray color space.

- virtual IDOMICCProfilePtr createGrayProfile (const CEDLVector< double > &whitePoint, CEDLVector< float > &gamma, const CEDLVector< double > &blackPoint=CEDLVector< double >())=0

  Create a profile for a gray color space with the given parameters, including the ability to specify an arbitrary gamma function.

- virtual IDOMICCProfilePtr createCalibratedRGBProfile (const CEDLVector< double > &whitePoint, const CEDLVector< double > &gamma=CEDLVector< double >(), const CEDLVector< double > &matrix=CEDLVector< double >(), const CEDLVector< double > &blackPoint=CEDLVector< double >())=0

  Create a profile for a calibrated RGB color space with the given parameters. Such a profile will be analogous to the PDF CalRGB color space.

- virtual IDOMICCProfilePtr createXyzProfile (const CEDLVector< double > &whitePoint, const CEDLVector< double > &blackPoint=CEDLVector< double >())=0

  Create an XYZ profile with the given parameters.

- virtual void setMapDeviceGrayToCMYKBlack (bool mapGrayDirectly)=0

  Configure how DeviceGray colors are converted to DeviceCMYK.

- virtual bool getMapDeviceGrayToCMYKBlack () const =0

  Setting that determines how the color manager will convert DeviceGray colors to DeviceCMYK.

- virtual void setDefaultBlackPointCompensation (eBlackPointCompensation defaultBpc)=0

  Set the default behaviour for black point compensation when (and only when) eBPCDefault is used by convertColors() routines.

- virtual eBlackPointCompensation getDefaultBlackPointCompensation () const =0

  Obtain the current black point compensation default setting.

- virtual void setDefaultBlackPreservation (bool preserveForText, bool preserveForOther)=0

  Set the default for black point preservation across the Jaws Mako instance.

- virtual bool getDefaultTextBlackPreservation () const =0

  Query the default black point preservation for text objects.

- virtual bool getDefaultOtherBlackPreservation () const =0

  Query the default black point preservation for non-text objects.

- virtual void setDeviceGrayIntercept (const IDOMColorSpacePtr &space)=0

  Set the intercept color space for DeviceGray. This is the color space that will be used for conversion if the color manager is asked to convert from or to DeviceGray.

- virtual IDOMColorSpacePtr getDeviceGrayIntercept ()=0

  Get the intercept color space for DeviceGray. This is the color space that will be used for conversion if the color manager is asked to convert from or to DeviceGray. The default DeviceGray intercept is sGray.

- virtual void setDeviceRGBIntercept (const IDOMColorSpacePtr &space)=0

  Set the intercept color space for DeviceRGB. This is the color space that will be used for conversion if the color manager is asked to convert from or to DeviceRGB.

- virtual IDOMColorSpacePtr getDeviceRGBIntercept ()=0

  Get the intercept color space for DeviceRGB. This is the color space that will be used for conversion if the color manager is asked to convert from or to DeviceRGB. The default DeviceRGB intercept is sRGB.

- virtual void setDeviceCMYKIntercept (const IDOMColorSpacePtr &space)=0

  Set the intercept color space for DeviceCMYK. This is the color space that will be used for conversion if the color manager is asked to convert from or to DeviceCMYK.

- virtual IDOMColorSpacePtr getDeviceCMYKIntercept ()=0

  Get the intercept color space for DeviceCMYK. This is the color space that will be used for conversion if the color manager is asked to convert from or to DeviceCMYK. The default DeviceCMYK intercept is a CMYK SWOP ICCBased color space.
- virtual IDOMColorSpacePtr interceptSpace (const IDOMColorSpacePtr &space)=0

  Gets the intercept associated with the given space if any, or return the original space if not. It is safe to pass any color space to this member.
- virtual IDOMICCProfilePtr getProfileForSpace (const IDOMColorSpacePtr &space)=0

  Get a profile for the given space, if possible. For ICCBased color spaces, this simply returns the profile.
- virtual void getPostScriptCSAForICCBasedSpace (CEDLSimpleBuffer &csaMemory, const IDOMColor↩ SpaceICCBasedPtr &space, eRenderingIntent intent)=0

  Get an equivalent PostScript CSA for an ICC based color space. This will be a CIEBased color space in the form of PostScript code.
- virtual bool getEmbeddedPSCSAForICCBasedSpace (CEDLSimpleBuffer &csaMemory, const IDOMColor↩ SpaceICCBasedPtr &space)=0

  Get any PostScript CSA present in an ICC Based color space. These are found under the 'ps2s' tag in an ICC profile, and are optional.
- virtual void useDeviceLinkForConversionsBetween (const IDOMColorSpaceICCBasedPtr &sourceSpace, const IDOMColorSpaceICCBasedPtr &destSpace, const IDOMICCProfilePtr &deviceLink)=0

  Set the DeviceLink ICC profile with specifying a combination of the source color space and the destination color space. If the same combination of sourceSpace and destSpace was specified with different DeviceLink ICC profile, the old DeviceLink ICC profile is replaced with the new DeviceLink ICC profile.
- virtual void useDeviceLinkForConversionsBetween (const IDOMICCProfilePtr &sourceProfile, const IDOMICCProfilePtr &destProfile, const IDOMICCProfilePtr &deviceLink)=0

  Convenience version of useDeviceLinkForConversionsBetween() that accepts IDOMICCProfiles instead of IDOMColorSpaceICCBased.
- virtual uint32 getNumDeviceLinkICCProfiles ()=0

  Get the number of DeviceLink ICC profiles which have been set.
- virtual void getDeviceLinkICCProfile (const uint8 index, IDOMColorSpaceICCBasedPtr &sourceSpace, IDOMColorSpaceICCBasedPtr &destSpace, IDOMICCProfilePtr &deviceLink)=0

  Get the DeviceLink ICC profile and the associated color spaces.
- virtual void deleteDeviceLinkForConversionsBetween (const IDOMColorSpaceICCBasedPtr &sourceSpace, const IDOMColorSpaceICCBasedPtr &destSpace)=0

  Delete the DeviceLink ICC profile associated with the specified combination of the source color space and the destination color space.
- virtual void deleteDeviceLinkForConversionsBetween (const IDOMICCProfilePtr &sourceProfile, const IDOMICCProfilePtr &destProfile)=0

  Convenience version of deleteDeviceLinkForConversionsBetween() that accepts IDOMICCProfiles instead of IDOMColorSpaceICCBased.
- virtual bool testGamut (uint32 numColors, const IDOMColorSpacePtr &sourceSpace, const IDOMColor↩ SpacePtr &destSpace, eRenderingIntent intent, eBlackPointCompensation bpc, const float ∗inColors)=0

  Tests whether the result of gamut checking is within the gamut of the destination color space.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams ∗pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory ∗pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static EDL_API IColorManagerPtr get (IEDLClassFactory *pFactory)

  Get the color manager singleton. Throws an IEDLError on failure.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.150.1 Detailed Description

Public interface to the EDL color manager. There is only one instance of the color manager for each factory. It can be retrieved using the IEDLFactory::getSingleton method, or by using the get() static function.

Exceptions of type IEDLError are thrown on errors.

7.150.2 Member Function Documentation

convertColors() [1/2]

```
virtual void IColorManager::convertColors (
            int32 numColors,
            bool hasAlpha,
            const IDOMColorSpacePtr & sourceSpace,
            const IDOMColorSpacePtr & destSpace,
            eRenderingIntent intent,
            eBlackPointCompensation bpc,
            const float * inColors,
            float * outColors )  [pure virtual]
```

Convert colors from one color space to another, using floating point samples.

Currently, only base color spaces are supported; Indexed and DeviceN spaces cannot be used. For conversion of multiple samples, samples are interleaved, with any alpha being last. For example, sRGB with alpha must be presented as r, g, b, a, r, g, b, a, etc.

Device Link profiles are supported. To use a Device Link profile, pass an IDOMColorSpaceICCBased colorspace containing that profile as the sourceSpace, and pass a NULL as the destSpace.

Parameters

numColors	The number of colors to convert.
hasAlpha	Set to true if the color data contains alpha.
sourceSpace	The color space describing the input samples.
destSpace	The desired color space for the output samples. May be NULL if destSpace is a Device Link profile.
intent	The desired rendering intent for the conversion.
bpc	The desired handling of black point compensation. If in doubt, use eBPCDefault.
inColors	Pointer to the input sample buffer.
outColors	Pointer to the output sample buffer.

convertColors() [2/2]

```
virtual void IColorManager::convertColors (
            int32 numColors,
            bool hasAlpha,
            const IDOMColorSpacePtr & sourceSpace,
            const IDOMColorSpacePtr & destSpace,
            eRenderingIntent intent,
            eBlackPointCompensation bpc,
            const uint8 * inColors,
            uint8 * outColors,
            int32 inBPS,
            int32 outBPS )  [pure virtual]
```

Convert colors from one color space to another, using integer samples. Currently, only base color spaces are supported; Indexed and DeviceN spaces cannot be used. For conversion of multiple samples, samples are interleaved, with any alpha being last. For example, sRGB with alpha must be presented as r, g, b, a, r, g, b, a, etc.

Device Link profiles are supported. To use a Device Link profile, pass an IDOMColorSpaceICCBased colorspace containing that profile as the sourceSpace, and pass a NULL as the destSpace.

Parameters

numColors	The number of colors to convert.
hasAlpha	Set to true if the color data contains alpha.
sourceSpace	The color space describing the input samples.
destSpace	The desired color space for the output samples. May be NULL if destSpace is a Device Link profile.
intent	The desired rendering intent for the conversion.
bpc	The desired handling of black point compensation. If in doubt, use eBPCDefault.
inColors	Pointer to the input sample buffer.
outColors	Pointer to the output sample buffer.
inBPS	The bits per sample of the input samples. Currently 1, 2, 4, 8, 12 and 16 bits are supported.
outBPS	The desired bits per sample of the output samples. Currently 1, 2, 4, 8, 12 and 16 bits are supported.

createCalibratedGrayProfile()

```
virtual IDOMICCProfilePtr IColorManager::createCalibratedGrayProfile (
            const CEDLVector< double > & whitePoint,
            double gamma = 1.0,
            const CEDLVector< double > & blackPoint = CEDLVector< double >() )  [pure virtual]
```

Create a profile for a calibrated gray color space with the given parameters. Such a profile will be analogous to the PDF CalGray color space.

Parameters

whitePoint	A three entry vector describing the white point of the color space, in the CIE XYZ color space.
gamma	Optional. The gamma for the color space, 1.0 if not provided
blackPoint	Optional. A three entry vector describing the black point of the color space, in the CIE XYZ color space. If not supplied, the values 0,0,0 will be used.

Returns

IDOMICCProfilePtr A smart pointer to the resulting profile.

createCalibratedRGBProfile()

```
virtual IDOMICCProfilePtr IColorManager::createCalibratedRGBProfile (
            const CEDLVector< double > & whitePoint,
            const CEDLVector< double > & gamma = CEDLVector< double >(),
            const CEDLVector< double > & matrix = CEDLVector< double >(),
            const CEDLVector< double > & blackPoint = CEDLVector< double >() )  [pure virtual]
```

Create a profile for a calibrated RGB color space with the given parameters. Such a profile will be analogous to the PDF CalRGB color space.

Parameters

whitePoint	A three entry vector describing the white point of the color space, in the CIE XYZ color space.
gamma	Optional. A vector containing the desired gamma for the color space, one value for R, G and B restpectively. 1 1 1 is used if not provided.
matrix	Optional. A nine entry vector representing a the linear transformation of the gamma corrected RGB values into CIE XYZ space. If not provided, then an identit matrix (1, 0, 0, 0, 1, 0, 0, 0, 1) will be used.
blackPoint	Optional. A three entry vector describing the black point of the color space, in the CIE XYZ color space. If not supplied, the values 0,0,0 will be used.

Returns

IDOMICCProfilePtr A smart pointer to the resulting profile.

createGrayProfile()

```
virtual IDOMICCProfilePtr IColorManager::createGrayProfile (
            const CEDLVector< double > & whitePoint,
```

```
            CEDLVector< float > & gamma,
            const CEDLVector< double > & blackPoint = CEDLVector< double >() )  [pure virtual]
```

Create a profile for a gray color space with the given parameters, including the ability to specify an arbitrary gamma function.

Parameters

whitePoint	A three entry vector describing the white point of the color space, in the CIE XYZ color space.
gamma	A gamma function consisting of a table of evenly- spaced, output values. For example, a 10 entry vector would describe the output of the gamma function for 0.1 increments between 0.0 and 1.0 inclusive. There must be at least two entries (which would describe a linear gamma function).
blackPoint	Optional. A three entry vector describing the black point of the color space, in the CIE XYZ color space. If not supplied, the values 0,0,0 will be used.

Returns

IDOMICCProfilePtr A smart pointer to the resulting profile.

createXyzProfile()

```
virtual IDOMICCProfilePtr IColorManager::createXyzProfile (
            const CEDLVector< double > & whitePoint,
            const CEDLVector< double > & blackPoint = CEDLVector< double >() )  [pure virtual]
```

Create an XYZ profile with the given parameters.

Parameters

whitePoint	A three entry vector describing the white point of the color space, in the CIE XYZ color space.
blackPoint	Optional. A three entry vector describing the black point of the color space, in the CIE XYZ color space. If not supplied, the values 0,0,0 will be used.

Returns

IDOMICCProfilePtr A smart pointer to the resulting profile.

deleteDeviceLinkForConversionsBetween() [1/2]

```
virtual void IColorManager::deleteDeviceLinkForConversionsBetween (
            const IDOMColorSpaceICCBasedPtr & sourceSpace,
            const IDOMColorSpaceICCBasedPtr & destSpace )  [pure virtual]
```

Delete the DeviceLink ICC profile associated with the specified combination of the source color space and the destination color space.

Parameters

sourceSpace	The desired input color space.
destSpace	The desired output color space.

deleteDeviceLinkForConversionsBetween() [2/2]

```
virtual void IColorManager::deleteDeviceLinkForConversionsBetween (
            const IDOMICCProfilePtr & sourceProfile,
            const IDOMICCProfilePtr & destProfile )  [pure virtual]
```

Convenience version of deleteDeviceLinkForConversionsBetween() that accepts IDOMICCProfiles instead of IDOMColorSpaceICCBased.

Parameters

sourceProfile	The desired input color profile.
destProfile	The desired output color profile.

get()

```
static EDL_API IColorManagerPtr IColorManager::get (
            IEDLClassFactory * pFactory )  [static]
```

Get the color manager singleton. Throws an IEDLError on failure.

Parameters

pFactory	The EDL Class factory to use

Returns

IColorManagerPtr Pointer to the color manager

getCMYKSWOPProfile()

```
virtual IDOMICCProfilePtr IColorManager::getCMYKSWOPProfile ( )  [pure virtual]
```

Retrieve the built-in CMYK SWOP profile.

Returns

IDOMICCProfilePtr A smart pointer to the resulting profile.

getDefaultBlackPointCompensation()

```
virtual eBlackPointCompensation IColorManager::getDefaultBlackPointCompensation ( ) const
[pure virtual]
```

Obtain the current black point compensation default setting.

```
Please see the description for setDefaultBlackPointCompensation() for details.
```

Returns

eBlackPointComponsation The default black point compensation setting.

getDefaultIntentForICCBasedSpace()

```
virtual eRenderingIntent IColorManager::getDefaultIntentForICCBasedSpace (
            const IDOMColorSpaceICCBasedPtr & space )  [pure virtual]
```

Find the default rendering intent for an ICCBased color space.

Parameters

space	The ICC space to interrogate

Returns

eRenderingIntent The default intent

getDefaultOtherBlackPreservation()

```
virtual bool IColorManager::getDefaultOtherBlackPreservation ( ) const  [pure virtual]
```

Query the default black point preservation for non-text objects.

Returns

bool The default black point preservation for non-text objects.

getDefaultTextBlackPreservation()

```
virtual bool IColorManager::getDefaultTextBlackPreservation ( ) const  [pure virtual]
```

Query the default black point preservation for text objects.

Returns

bool The default black point preservation for text objects.

getDeviceCMYKIntercept()

```
virtual IDOMColorSpacePtr IColorManager::getDeviceCMYKIntercept ( )  [pure virtual]
```

Get the intercept color space for DeviceCMYK. This is the color space that will be used for conversion if the color manager is asked to convert from or to DeviceCMYK. The default DeviceCMYK intercept is a CMYK SWOP ICCBased color space.

Returns

IDOMColorSpacePtr A smart pointer to the intercept color space.

getDeviceGrayIntercept()

```
virtual IDOMColorSpacePtr IColorManager::getDeviceGrayIntercept ( )  [pure virtual]
```

Get the intercept color space for DeviceGray. This is the color space that will be used for conversion if the color manager is asked to convert from or to DeviceGray. The default DeviceGray intercept is sGray.

Returns

IDOMColorSpacePtr A smart pointer to the intercept color space

getDeviceLinkICCProfile()

```
virtual void IColorManager::getDeviceLinkICCProfile (
            const uint8 index,
            IDOMColorSpaceICCBasedPtr & sourceSpace,
            IDOMColorSpaceICCBasedPtr & destSpace,
            IDOMICCProfilePtr & deviceLink )  [pure virtual]
```

Get the DeviceLink ICC profile and the associated color spaces.

Parameters

index	The index of DeviceLink ICC profile which has already been set.
sourceSpace	The input color space associated with the DeviceLink ICC profile.
destSpace	The output color space associated with the DeviceLink ICC profile.
deviceLink	The desired DeviceLink ICC profile.

getDeviceRGBIntercept()

```
virtual IDOMColorSpacePtr IColorManager::getDeviceRGBIntercept ( )  [pure virtual]
```

Get the intercept color space for DeviceRGB. This is the color space that will be used for conversion if the color manager is asked to convert from or to DeviceRGB. The default DeviceRGB intercept is sRGB.

Returns

IDOMColorSpacePtr A smart pointer to the intercept color space.

getEmbeddedPSCSAForICCBasedSpace()

```
virtual bool IColorManager::getEmbeddedPSCSAForICCBasedSpace (
            CEDLSimpleBuffer & csaMemory,
            const IDOMColorSpaceICCBasedPtr & space )  [pure virtual]
```

Get any PostScript CSA present in an ICC Based color space. These are found under the 'ps2s' tag in an ICC profile, and are optional.

Parameters

csaMemory	A reference to a simple buffer to store the given csa.
space	The ICC Based color space to process.

Returns

bool True if an embedded CSA was found

getMapDeviceGrayToCMYKBlack()

```
virtual bool IColorManager::getMapDeviceGrayToCMYKBlack ( ) const  [pure virtual]
```

Setting that determines how the color manager will convert DeviceGray colors to DeviceCMYK.

See also

setMapDeviceGrayToCMYKBlack() for details.

Returns

bool The current setting

getNumComponentsForICCBasedSpace()

```
virtual uint8 IColorManager::getNumComponentsForICCBasedSpace (
            const IDOMColorSpaceICCBasedPtr & space )  [pure virtual]
```

Find the number of components for an ICCBased color space.

Parameters

space	The ICC space to interrogate

Returns

uint8 The number of components in the ICC space

getNumComponentsForICCProfile()

```
virtual uint8 IColorManager::getNumComponentsForICCProfile (
            const IDOMICCProfilePtr & profile )  [pure virtual]
```

Find the number of components for an ICC profile.

Parameters

profile	The ICC profile to interrogate

Returns

uint8 The number of components in the ICC space

getNumDeviceLinkICCProfiles()

```
virtual uint32 IColorManager::getNumDeviceLinkICCProfiles ( )  [pure virtual]
```

Get the number of DeviceLink ICC profiles which have been set.

Returns

uint32 The number of DeviceLink ICC profiles.

getPostScriptCSAForICCBasedSpace()

```
virtual void IColorManager::getPostScriptCSAForICCBasedSpace (
            CEDLSimpleBuffer & csaMemory,
            const IDOMColorSpaceICCBasedPtr & space,
            eRenderingIntent intent )  [pure virtual]
```

Get an equivalent PostScript CSA for an ICC based color space. This will be a CIEBased color space in the form of PostScript code.

Parameters

csaMemory	A reference to a simple buffer to store the generated csa.
space	The ICC Based color space to process.
intent	The intent for which the CSA should be generated.

getProfileColorSpaceForICCBasedSpace()

```
virtual uint32 IColorManager::getProfileColorSpaceForICCBasedSpace (
            const IDOMColorSpaceICCBasedPtr & space )  [pure virtual]
```

Get the color space type for an ICCBased color space.

Parameters

space	the ICC space to interrogate

Returns

uint32 The ICC color space signature, using the enumeration as per the ICC specification

getProfileForSpace()

```
virtual IDOMICCProfilePtr IColorManager::getProfileForSpace (
            const IDOMColorSpacePtr & space )  [pure virtual]
```

Get a profile for the given space, if possible. For ICCBased color spaces, this simply returns the profile.

- For DeviceGray, DeviceRGB and DeviceCMYK color spaces, this gets the profile for the intercept color space associated with the respective intercept.
- For sRGB, scRGB and sGray, this returns the built-in profiles.

For all other color space types, NULL will be returned.

Parameters

space	The color space to get the profile for.

Returns

IDOMICCProfilePtr A smart pointer to the profile, if applicable, otherwise NULL.

getProfileNameForICCBasedSpace()

```
virtual void IColorManager::getProfileNameForICCBasedSpace (
            const IDOMColorSpaceICCBasedPtr & space,
            EDLSysString & name )  [pure virtual]
```

Get the name of the profile for an ICCBased color space.

Parameters

space	The ICC color space to interrogate.
name	Reference to receive the profile name

getProfileVersionForICCBasedSpace()

```
virtual void IColorManager::getProfileVersionForICCBasedSpace (
            const IDOMColorSpaceICCBasedPtr & space,
            uint8 & majorVersion,
            uint8 & minorVersion )  [pure virtual]
```

Get the version number of the given ICC profile.

Parameters

space	The ICC color space to interrogate.
majorVersion	Reference to receive the major version
minorVersion	Reference to receive the minor version

getscRGBProfile()

```
virtual IDOMICCProfilePtr IColorManager::getscRGBProfile ( )  [pure virtual]
```

Retrieve the built in scRGB profile.

Returns

IDOMICCProfilePtr A smart pointer to the resulting profile.

getsGrayProfile()

```
virtual IDOMICCProfilePtr IColorManager::getsGrayProfile ( )  [pure virtual]
```

Retrieve the built-in sGray profile.

Returns

IDOMICCProfilePtr A smart pointer to the resulting profile.

getsRGBProfile()

```
virtual IDOMICCProfilePtr IColorManager::getsRGBProfile ( )  [pure virtual]
```

Retrieve the built-in sRGB profile.

Returns

IDOMICCProfilePtr A smart pointer to the resulting profile.

interceptSpace()

```
virtual IDOMColorSpacePtr IColorManager::interceptSpace (
            const IDOMColorSpacePtr & space )  [pure virtual]
```

Gets the intercept associated with the given space if any, or return the original space if not. It is safe to pass any color space to this member.

Parameters

space	The color space to find the intercept for.

Returns

IDOMColorSpacePtr A smart pointer to the intercepted space, if applicable. Otherwise the original space is returned.

setDefaultBlackPointCompensation()

```
virtual void IColorManager::setDefaultBlackPointCompensation (
            eBlackPointCompensation defaultBpc )  [pure virtual]
```

Set the default behaviour for black point compensation when (and only when) eBPCDefault is used by convertColors() routines.

Note that black point compensation will not apply to cases where an absolute rendering intent is used.

The choices are:

- *eBPCDefault* (default): Black point compensation will be applied when ICC Version 4 or greater profiles are used and the intent is perceptual or saturation.
- *eBPCOn:* Black point compensation will be applied for all but absolute rendering intents.
- *eBPCOff:* Black point compensation wll not be used.

Parameters

defaultBpc	The default black point compensation to use.

setDefaultBlackPreservation()

```
virtual void IColorManager::setDefaultBlackPreservation (
            bool preserveForText,
            bool preserveForOther )  [pure virtual]
```

Set the default for black point preservation across the Jaws Mako instance.

```
Please note that this has no direct effect in IColorManager, but instead provides:

- The default setting for IColorConverterTransform::setBlackPreservation()
- The default setting for IRendererTransform::setBlackPreservation()
- The default setting for IJawsRenderer::setBlackPreservation()

The default is false for both text and non-text (other).
```

setDeviceCMYKIntercept()

```
virtual void IColorManager::setDeviceCMYKIntercept (
            const IDOMColorSpacePtr & space )  [pure virtual]
```

Set the intercept color space for DeviceCMYK. This is the color space that will be used for conversion if the color manager is asked to convert from or to DeviceCMYK.

Parameters

space	The desired DeviceCMYK intercept color space.

setDeviceGrayIntercept()

```
virtual void IColorManager::setDeviceGrayIntercept (
            const IDOMColorSpacePtr & space )  [pure virtual]
```

Set the intercept color space for DeviceGray. This is the color space that will be used for conversion if the color manager is asked to convert from or to DeviceGray.

Parameters

space	The desired DeviceGray intercept color space.

setDeviceRGBIntercept()

```
virtual void IColorManager::setDeviceRGBIntercept (
            const IDOMColorSpacePtr & space )  [pure virtual]
```

Set the intercept color space for DeviceRGB. This is the color space that will be used for conversion if the color manager is asked to convert from or to DeviceRGB.

Parameters

space	The desired DeviceRGB intercept colorspace.

setMapDeviceGrayToCMYKBlack()

```
virtual void IColorManager::setMapDeviceGrayToCMYKBlack (
            bool mapGrayDirectly )  [pure virtual]
```

Configure how DeviceGray colors are converted to DeviceCMYK.

Normally, all color conversions in JawsMako are achieved using color management. In particular, this includes the DeviceGray, DeviceCMYK and DeviceRGB color spaces, which are converted using the intercept color spaces configured in the functions below.

However this means that conversions between DeviceGray and DeviceCMYK generally pass through ICC conversion. Gray values may not map exclusively to DeviceCMYK K values, resulting in non-zero values in the Cyan, Magenta and Yellow components.

This may be unsuitable in some environments, in particular when dealing with overprint simulation.

This behavior can be changed with this member. If the argument is

- *true:* All color conversions from DeviceGray to DeviceCMYK will simply map and invert the gray value to the K channel. This will also apply to any rendering operations performed by JawsMako.
- *false* (default): The intercept color spaces will be used to convert from DeviceGray to DeviceCMYK.

Parameters

mapGrayDirectly	Determine how DeviceGray colors are converted to DeviceCMYK

testGamut()

```
virtual bool IColorManager::testGamut (
            uint32 numColors,
            const IDOMColorSpacePtr & sourceSpace,
            const IDOMColorSpacePtr & destSpace,
            eRenderingIntent intent,
            eBlackPointCompensation bpc,
            const float * inColors )  [pure virtual]
```

Tests whether the result of gamut checking is within the gamut of the destination color space.

Currently, only base color spaces are supported; Indexed and DeviceN spaces cannot be used.

Device Link profiles are not supported.

Parameters

numColors	The number of colors to test.
sourceSpace	The color space describing the input samples.
destSpace	The desired destination color space.
intent	The desired rendering intent for the test.
bpc	The desired handling of black point compensation. If in doubt, use eBPCDefault.
inColors	Pointer to the input sample buffer.

Returns

bool false, if any colors are Out-Of-Gamut, true otherwise.

useDeviceLinkForConversionsBetween() [1/2]

```
virtual void IColorManager::useDeviceLinkForConversionsBetween (
            const IDOMColorSpaceICCBasedPtr & sourceSpace,
```

```
            const IDOMColorSpaceICCBasedPtr & destSpace,
            const IDOMICCProfilePtr & deviceLink )  [pure virtual]
```

Set the DeviceLink ICC profile with specifying a combination of the source color space and the destination color space. If the same combination of sourceSpace and destSpace was specified with different DeviceLink ICC profile, the old DeviceLink ICC profile is replaced with the new DeviceLink ICC profile.

Parameters

sourceSpace	The desired input color space.
destSpace	The desired output color space.
deviceLink	The desired DeviceLink ICC profile.

useDeviceLinkForConversionsBetween() [2/2]

```
virtual void IColorManager::useDeviceLinkForConversionsBetween (
            const IDOMICCProfilePtr & sourceProfile,
            const IDOMICCProfilePtr & destProfile,
            const IDOMICCProfilePtr & deviceLink )  [pure virtual]
```

Convenience version of useDeviceLinkForConversionsBetween() that accepts IDOMICCProfiles instead of IDOMColorSpaceICCBased.

Parameters

sourceProfile	The desired input color profile.
destProfile	The desired output color profile.
deviceLink	The desired DeviceLink ICC profile.

The documentation for this class was generated from the following file:

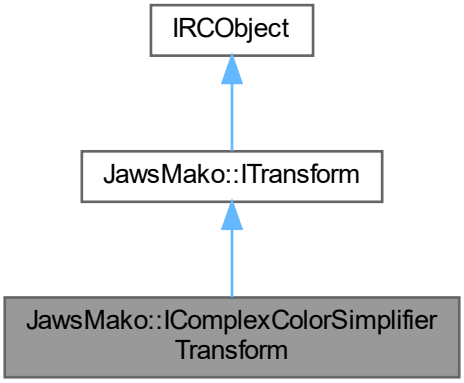
- icolormanager.h

7.151 JawsMako::IComplexColorSimplifierTransform Class Reference

A simple transform that looks for DeviceN or Indexed color spaces, and where found, simplifies the hosting objects to use the underlying color space (for Indexed cases) or the alternate color space (for DeviceN cases). Useful in particular for consumers that do not support such color spaces. DOM brushes using only the /None colorant in a DeviceN colorspace may be dropped entirely.

```
#include <transforms.h>
```

Inheritance diagram for JawsMako::IComplexColorSimplifierTransform:



Static Public Member Functions

- static JAWSMAKO_API IComplexColorSimplifierTransformPtr create (const IJawsMakoPtr &jawsMako, const IAbortPtr &abort=IAbortPtr())

  Create the transform.

Additional Inherited Members

Public Member Functions inherited from JawsMako::ITransform

- virtual IDOMBrushPtr transform (const IDOMBrushPtr &brush, eBrushUsage usage=eBUGeneral, const CTransformState &state=CTransformState())=0

  Apply the transform to the given brush, if applicable. These transforms are thread safe.
- virtual IDOMImagePtr transform (const IDOMImagePtr &image, const CTransformState &state=CTransformState())=0

  Apply the transform to the given image, if applicable. These transforms are thread safe.
- virtual IDOMColorPtr transform (const IDOMColorPtr &color, const CTransformState &state=CTransformState())=0

  Apply the transform to the given color, if applicable. These transforms are thread safe.
- virtual IDOMColorSpacePtr transform (const IDOMColorSpacePtr &colorSpace, const CTransformState &state=CTransformState())=0

  Apply the transform to the given colorspace, if applicable. These transforms are thread safe.
- virtual IDOMNodePtr transform (const IDOMNodePtr &node, bool &changed, bool transformChildren=true, const CTransformState &state=CTransformState())=0

  Apply the transform to the given node, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time.
- virtual void transformPage (const IPagePtr &page, bool transformContent=true, bool transform↩ Annotations=true)=0

  Apply the transform to the given page, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time. The transform will also apply to the annotations appearances.
- virtual void **flushCaches** ()=0

  Flush the caches used by the transform. Most transforms cache recently transformed results to improve the performance of repeated transformations of equivalent results. However, it is possible that some cached results may point to entities that no longer exist, such as content inside an XPS file that no longer exists. If you are deleting or replacing files where transforms have been used, it is advisable to invoke this routine to clear the caches.
- virtual void setProgressMonitor (const IProgressMonitorPtr &progressMonitor)=0

  Set the IProgressMonitor object for this transform to allow for monitoring the progress of the transform.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.151.1 Detailed Description

A simple transform that looks for DeviceN or Indexed color spaces, and where found, simplifies the hosting objects to use the underlying color space (for Indexed cases) or the alternate color space (for DeviceN cases). Useful in particular for consumers that do not support such color spaces. DOM brushes using only the /None colorant in a DeviceN colorspace may be dropped entirely.

7.151.2 Member Function Documentation

create()

```
static JAWSMAKO_API IComplexColorSimplifierTransformPtr JawsMako::IComplexColorSimplifier↩
Transform::create (
            const IJawsMakoPtr & jawsMako,
            const IAbortPtr & abort = IAbortPtr() )  [static]
```

Create the transform.

Parameters

jawsMako	The JawsMako instance.
abort	An abort callback handler.

Returns

The new instance.

The documentation for this class was generated from the following file:

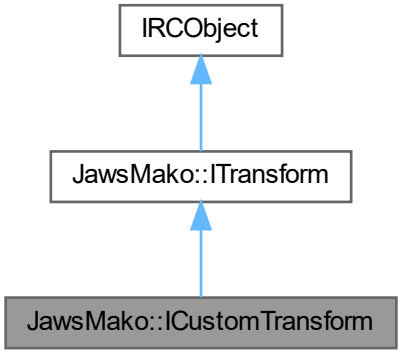
- transforms.h

7.152 JawsMako::ICustomTransform Class Reference

A transform that allows the implementation to be provided externally.

`#include <customtransform.h>`

Inheritance diagram for JawsMako::ICustomTransform:



Classes

- class IImplementation

  Callback interface that provides methods for actually doing the work. Override the cases for the objects you wish to edit or are otherwise interested in.

Static Public Member Functions

- static JAWSMAKO_API ICustomTransformPtr create (const IJawsMakoPtr &jawsMako, IImplementation *implementation, const IAbortPtr &abort=IAbortPtr(), bool dependsOnClipBounds=true, bool dependsOn↩GroupSpace=true, bool dependsOnRenderingIntent=true, bool dependsOnTransform=true, bool depends↩OnBrushUsage=true, bool dependsOnEdgeMode=true, bool dependsOnUncoloredTilingBrush=true)

  Create an ICustomTransform object whose underlying methods are supported by the implementation object. Note - this creation routine will be removed in a future version of Mako and replaced with a version that is equivalent to calling this function with default arguments.
- static JAWSMAKO_API ICustomTransformPtr create (const IJawsMakoPtr &jawsMako, IImplementation *implementation, const IAbortPtr &abort, bool dependsOnClipBounds, bool dependsOnGroupSpace, bool dependsOnRenderingIntent, bool dependsOnTransformScale, bool dependsOnTransformOffset, bool dependsOnBrushUsage, bool dependsOnEdgeMode, bool dependsOnUncoloredTilingBrush)

  Create an ICustomTransform object whose underlying methods are supported by the implementation object.

Additional Inherited Members

Public Member Functions inherited from JawsMako::ITransform

- virtual IDOMBrushPtr transform (const IDOMBrushPtr &brush, eBrushUsage usage=eBUGeneral, const CTransformState &state=CTransformState())=0

  Apply the transform to the given brush, if applicable. These transforms are thread safe.
- virtual IDOMImagePtr transform (const IDOMImagePtr &image, const CTransformState &state=CTransformState())=0

  Apply the transform to the given image, if applicable. These transforms are thread safe.
- virtual IDOMColorPtr transform (const IDOMColorPtr &color, const CTransformState &state=CTransformState())=0

  Apply the transform to the given color, if applicable. These transforms are thread safe.
- virtual IDOMColorSpacePtr transform (const IDOMColorSpacePtr &colorSpace, const CTransformState &state=CTransformState())=0

  Apply the transform to the given colorspace, if applicable. These transforms are thread safe.
- virtual IDOMNodePtr transform (const IDOMNodePtr &node, bool &changed, bool transformChildren=true, const CTransformState &state=CTransformState())=0

  Apply the transform to the given node, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time.
- virtual void transformPage (const IPagePtr &page, bool transformContent=true, bool transform↩ Annotations=true)=0

  Apply the transform to the given page, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time. The transform will also apply to the annotations appearances.
- virtual void **flushCaches** ()=0

  Flush the caches used by the transform. Most transforms cache recently transformed results to improve the performance of repeated transformations of equivalent results. However, it is possible that some cached results may point to entities that no longer exist, such as content inside an XPS file that no longer exists. If you are deleting or replacing files where transforms have been used, it is advisable to invoke this routine to clear the caches.
- virtual void setProgressMonitor (const IProgressMonitorPtr &progressMonitor)=0

  Set the IProgressMonitor object for this transform to allow for monitoring the progress of the transform.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.152.1 Detailed Description

A transform that allows the implementation to be provided externally.

This class allows Mako users to create their own transforms. In Mako, transforms are a convenient method of performing operations on items in the DOM tree with a minimum of fuss.

To create a transform, provide your own implementation of the ICustomTransform::IImplementation. Here you can override the default handlers for a given graphical node, color, colorspace, font etc, making the edits you wish to that object, while the custom transform mechanism will update the DOM with your result, taking care of its use in shared objects like forms and patterns automatically. Once you have an IImplementation, use ICustomTransform::create() and use the resulting transform as you would any other. These transforms can also be inserted into ITransform↩ Chains.

For most objects, a CTransformState will be provided, providing some context for edits that rely on scaling, the group color space, clip, etc.

This can also be used for scanning purposes. For example, let's say you wanted to find out information for every image in a document. You could write a custom transform to override transformImageBrush(). Your code will be called every time a unique imagebrush (IDOMImageBush) is encountered, and from there get to the image (IDOMImage) and its frame (IDOMFrame) to obtain all the information you need - size, resolution, colorspace etc in conjunction with the CTransformState.

The documentation for this class was generated from the following file:

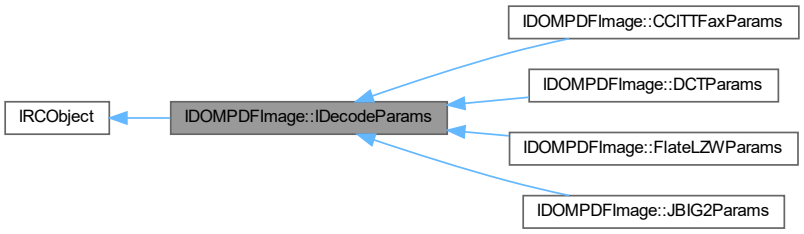
- customtransform.h

7.153 IDOMPDFImage::IDecodeParams Class Reference

Abstract interface for per-image decoding filter parameters.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMPDFImage::IDecodeParams:



Additional Inherited Members

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.153.1 Detailed Description

Abstract interface for per-image decoding filter parameters.

The documentation for this class was generated from the following file:

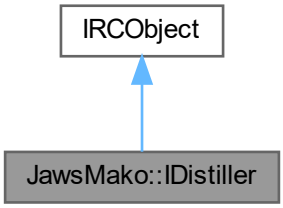
- idomimageresource.h

7.154 JawsMako::IDistiller Class Reference

An instance of the JawsMako Distiller class.

`#include <distiller.h>`

Inheritance diagram for JawsMako::IDistiller:



Public Types

- enum ePDFVersion {
  ePDF1_3 = DIST_MAKE_PDF_VERSION(1, 3) , ePDF1_4 = DIST_MAKE_PDF_VERSION(1, 4) , ePDF1_5 = DIST_MAKE_PDF_VERSION(1, 5) , ePDF1_6 = DIST_MAKE_PDF_VERSION(1, 6) ,
  ePDF1_7 = DIST_MAKE_PDF_VERSION(1, 7) }

  Supported PDF versions.
- enum eTransferFunctionMethod { eTRApply , eTRRemove , eTRPreserve }

  Enumeration for transfer function methods.
- enum eImageCompression {
  eICNone , eICAuto , eICFlate , eICFlatePredict ,
  eICDCT , eICLZW , eICCCITT }

  Image compression formats.
- enum eJpegQuality { eJQLow , eJQMedium , eJQHigh , eJQUser }

  JPEG-equivalent quality factors (QFactor)
- enum eImageDownsamplingMethod { eIDNone , eIDSubsample , eIDAverage , eIDBicubic }

  Image downsampling methods.
- enum eDocumentInfo {
  eDIAuthor = 0 , eDISubject , eDITitle , eDICreator ,
  eDIN }

  Document Information types.
- enum eThumbnailType { eTHNone = 0 , eTHColor , eTHMono }

  PDF Thumbnail types.

Public Member Functions

- virtual void setParameter (const U8String &param, const U8String &value)=0

  Apply a key-value pair output parameter with a string value. The parameter name is case insensitive. Please refer to the supplied documentation for the details of the available parameters and their ranges.
- virtual void setPdfVersion (ePDFVersion version)=0

  Set the PDF version to generate.
- virtual void setCompressPages (bool compressPages)=0

  Set whether or not compression should be applied to page content. The default is true.
- virtual void setAutoRotatePages (bool autoRotatePages)=0

  Set whether pages should be automatically rotated. The default is false.
- virtual void setSubsetFonts (bool subset)=0

  Set whether embedded fonts should be subset or not. The default is true.
- virtual void setEmbedFonts (bool embed)=0

  Set whether fonts should be forcibly embedded. The standard 14 fonts are not affected by this. The default is true.
- virtual void setEmbedBase14Fonts (bool embed)=0

  Set whether the base 14 PDF fonts should be forcibly embedded. Only the standard 14 fonts are not affected by this; use setEmbedFonts() to affect other fonts. The default is false.
- virtual void setEmbedDiskBasedCidFonts (bool embed)=0

  Set whether CID fonts obtained from resources (i.e, disk based CID fonts) can be embedded. This setting is only effective if the "EmbedFonts" parameter is set to true.
- virtual void setResolution (float resolution)=0

  Set the target resolution for the output, in dots per inch. The default is 288. Equivalent to calling setParameter with the parameter name "Resolution".
- virtual void setTransfers (eTransferFunctionMethod transfers)=0

  Set the desired method to for controlling transfer functions.
- virtual void setColorImageCompression (eImageCompression compression)=0

  Set the image compression for color images. The default is eICFlate.
  CCITT is not allowed for color images.
- virtual void setGrayImageCompression (eImageCompression compression)=0

  Set the image compression for gray images. The default is eICFlate.
  CCITT is not allowed for gray images.
- virtual void setMonoImageCompression (eImageCompression compression)=0

  Set the image compression for monochrome images. The default is eICCCITT.
  JPEG/DCT and auto are not allowed for mono images.
- virtual void setColorJPEGQuality (eJpegQuality quality)=0

  Set the JPEG quality to use when compressing color images in DCT format. Equivalent to calling setParameter() with the parameter name "ColorJPEGQuality" enumeration value as a string ("Low", "Medium", "High" or "User"). The default is eJQLow.
- virtual void setColorQFactor (float qfactor)=0

  Set the QFactor to use when compressing color images in DCT format. Applies only when setting the JPEG quality to eJQUser with setColorJPEGQuality().
- virtual void setGrayJPEGQuality (eJpegQuality quality)=0

  Set the JPEG quality to use when compressing gray images in DCT format. Equivalent to calling setParameter() with the parameter name "GrayJPEGQuality" enumeration value as a string ("Low", "Medium", "High" or "User"). The default is eJQLow.
- virtual void setGrayQFactor (float qfactor)=0

  Set the QFactor to use when compressing gray images in DCT format. Applies only when setting the JPEG quality to eJQUser with setColorJPEGQuality().
- virtual void setColorImageDownsampling (eImageDownsamplingMethod method)=0

  Set the desired downsampling method for color images. The default method is eIDNone (no downsampling).
- virtual void setGrayImageDownsampling (eImageDownsamplingMethod method)=0

  Set the desired downsampling method for gray images. The default method is eIDNone (no downsampling).
- virtual void setMonoImageDownsampling (eImageDownsamplingMethod method)=0

Set the desired downsampling method for monochrome images. The default method is eIDNone (no downsampling).

- virtual void setColorImageResolution (float resolution)=0

  Set the desired resolution for color images when downsampling. The default is 72.
- virtual void setGrayImageResolution (float resolution)=0

  Set the desired resolution for gray images when downsampling. The default is 72.
- virtual void setMonoImageResolution (float resolution)=0

  Set the desired resolution for monochrome images when downsampling. The default is 72.
- virtual void setConvertCMYKImagesToRGB (bool convert)=0

  Set whether to convert CMYK images to RGB. The default is false.
- virtual void setProlog (const IInputStreamPtr &prolog)=0

  Set a prolog stream to be consumed by the PostScript interpreter before the input stream is processed, or NULL to clear.
- virtual void setEpilog (const IInputStreamPtr &epilog)=0

  Set a epilog stream to be consumed by the PostScript interpreter after the input stream is processed, or NULL to clear.
- virtual void setDefaultPanoseStyle (const U8String &defaultPanoseStyle)=0

  Sets the default Panose style to be used when emitting a CID font that is not in the Panose list, or NULL to clear. The Panose style must be a 12-byte hexadecimal string as described in the PDF reference, for example "010502020300000000000000".
- virtual void setPanose (const IInputStreamPtr &panose)=0

  Sets a stream that refers to a list of CID font names and their associated Panose styles.
- virtual void getFontNames (const U8String &font, CU8StringVect &names)=0

  Gets a list of font names from a given file.
- virtual void addFont (const U8String &font)=0

  Adds a font.
- virtual void addFonts (const CU8StringVect &fonts)=0

  Adds fonts.
- virtual void removeFont (const U8String &fontName)=0

  Removes a named font.
- virtual void setFontDevicePath (const U8String &path)=0

  Sets the font device path.
- virtual void setResourceDevicePath (const U8String &path)=0

  Sets the resource device path.
- virtual void setTrimToBBox (bool trimToBBox)=0

  Set whether to use the BoundingBox DSC comment for the page size. The default is false.
- virtual void setCropToBBox (bool cropToBBox)=0

  Set whether to use the BoundingBox DSC comment for the PDF CropBox. The default is false.
- virtual void setHiResBBox (bool hiResBBox)=0

  Set whether to use the HiResBoundingBox DSC comment if present.
- virtual void setCompressObjects (bool compressObjects)=0

  Set whether or not to compress individual objects.
- virtual void setPageFilterLow (uint64 pageLow)=0

  Set lower page limit.
- virtual void setPageFilterHigh (uint64 pageHigh)=0

  Set upper page limit.
- virtual void setPageFilterRange (uint64 pageLow, uint64 pageHigh)=0

  Set page range.
- virtual void setDocumentInformationString (eDocumentInfo key, const U8String &value)=0

  Set PDF Document Information string.
- virtual void setEncryption (const U8String &ownerPassword, const U8String &userPassword="", uint32 permissions=0)=0

  Set the encryption for the output PDF.

- virtual void setOverprint (bool overprint)=0

  Set whether overprinting should be preserved. The default is false.
- virtual void setOverprintMode (bool overprintMode)=0

  Set the overprint mode. The default is true, which sets non-zero overprint mode (/OPM 1).
- virtual void setHalftone (bool halftone)=0

  Set whether halftone information should be preserved. The default is false.
- virtual void setOPI (bool opi)=0

  Set whether OPI comments should be preserved. The default is false.
- virtual void setAscii85EncodePages (bool ascii85EncodePages)=0

  Set whether to ASCII85-encode streams. The default is false.
- virtual void setLinearize (bool linearize)=0

  Set whether the output should be linearized. The default is false.
- virtual void setJobTicket (bool jobTicket)=0

  Set whether to add a job ticket object to the output. The default is false.
- virtual void setThumbnails (eThumbnailType thumbnails)=0

  Set the PDF thumbnail type.
- virtual void setUseDeviceDependentColor (bool useDeviceDependentColor)=0

  Set whether to convert device independent colors to device dependent colors The default is false.
- virtual void setCommentMonitor (IPSCommentMonitor *commentMonitor)=0

  Set a comment monitor.
- virtual void distill (const IInputStreamPtr &inputStream, const IOutputStreamPtr &outputStream, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())=0

  Convert (distill) an input PostScript stream to PDF.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IDistillerPtr create (const IJawsMakoPtr &jawsMako)

  Create an IDistiller interface.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.154.1 Detailed Description

An instance of the JawsMako Distiller class.

7.154.2 Member Enumeration Documentation

eDocumentInfo

enum JawsMako::IDistiller::eDocumentInfo

Document Information types.

Enumerator

eDIAuthor	The document author.
eDISubject	The document subject.
eDITitle	The document title.
eDICreator	The document creator.
eDIN	The number of Document Information types.

eImageCompression

enum JawsMako::IDistiller::eImageCompression

Image compression formats.

Enumerator

eICNone	Uncompressed.
eICAuto	Automatic selection of appropriate compression method.
eICFlate	Flate (zip), a lossless compression method.
eICFlatePredict	Flate with predictor (which can improve compression at the cost of additional processing)
eICDCT	Discrete Cosine Transform, a lossy form of compression used for color or grayscale images, defined by the Joint Photographic Experts Group (JPEG).
eICLZW	LZW (Lempel-Ziv-Welch), a lossless compression method.
eICCCITT	A method for compressing monochrome images, defined by Comité Consultatif International Téléphonique et Télégraphique (ITU-T),.

eImageDownsamplingMethod

enum JawsMako::IDistiller::eImageDownsamplingMethod

Image downsampling methods.

Enumerator

eIDNone	No downsampling.
eIDSubsample	Subsampling.
eIDAverage	Average downsampling.
eIDBicubic	Bicubic downsampling.

eJpegQuality

enum JawsMako::IDistiller::eJpegQuality

JPEG-equivalent quality factors (QFactor)

Enumerator

eJQLow	eJQLow is equivalent to a QFactor of 0.1
eJQMedium	eJQMedium is equivalent to a QFactor of 0.5
eJQHigh	eJQHigh is equivalent to a QFactor of 1.3
eJQUser	Compress using the QFactor set by setColorQFactor()

ePDFVersion

enum JawsMako::IDistiller::ePDFVersion

Supported PDF versions.

Enumerator

ePDF1↩ _3	PDF 1.3.
ePDF1↩ _4	PDF 1.4.
ePDF1↩ _5	PDF 1.5.
ePDF1↩ _6	PDF 1.6.
ePDF1↩ _7	PDF 1.7.

eThumbnailType

enum JawsMako::IDistiller::eThumbnailType

PDF Thumbnail types.

Enumerator

eTHNone	No thumbnails.
eTHColor	Color thumbnails.
eTHMono	Monochrome thumbnails.

eTransferFunctionMethod

enum JawsMako::IDistiller::eTransferFunctionMethod

Enumeration for transfer function methods.

Enumerator

eTRApply	The effect of a transfer function is applied to the page content before being stored in the PDF file.
eTRRemove	Transfer functions are ignored.
eTRPreserve	Transfer functions are stored in the PDF file.

7.154.3 Member Function Documentation

addFont()

```
virtual void JawsMako::IDistiller::addFont (
            const U8String & font )  [pure virtual]
```

Adds a font.

Adds a font to the list of fonts for use when distilling with in-memory devices. Both the PostScript font name and the full name will be added to the list.

When using font or resource devices on disk, addFont() will add the font to the list of fonts on disk. Any existing font with the same name will be removed.

Only TrueType, TrueType collections or OpenType/CFF fonts are supported.

Equivalent to calling setParameter() with the parameter name "AddFont" the file name of the font.

Parameters

font	The input font file name.

addFonts()

```
virtual void JawsMako::IDistiller::addFonts (
            const CU8StringVect & fonts )  [pure virtual]
```

Adds fonts.

Convenience function to add fonts to the list of fonts for use when distilling with in-memory devices. Equivalent to calling addFont() for each font in the fonts vector.

Parameters

fonts	The fonts to add.

create()

```
static JAWSMAKO_API IDistillerPtr JawsMako::IDistiller::create (
            const IJawsMakoPtr & jawsMako )  [static]
```

Create an IDistiller interface.

The IDistiller interface allows PostScript to be converted (distilled) to PDF.

Parameters

jawsMako	The IJawsMako object.

Returns

IDistillerPtr A smart pointer to the IDistiller object.

distill()

```
virtual void JawsMako::IDistiller::distill (
            const IInputStreamPtr & inputStream,
            const IOutputStreamPtr & outputStream,
            const IProgressMonitorPtr & progressMonitor = IProgressMonitorPtr() )  [pure virtual]
```

Convert (distill) an input PostScript stream to PDF.

The input and output streams will be both opened and closed, and will not be cloned.

Parameters

inputStream	The input PostScript stream.
outputStream	The output stream to write the PDF to.
progressMonitor	A smart pointer to an IProgressMonitor object which can be NULL if no such object was passed in.

getFontNames()

```
virtual void JawsMako::IDistiller::getFontNames (
            const U8String & font,
            CU8StringVect & names )  [pure virtual]
```

Gets a list of font names from a given file.

Both the PostScript font name and the full name will be retrieved from the file. Only TrueType, TrueType collections or OpenType/CFF fonts are supported.

If the font argument is an empty string, names will be populated with the names of all known fonts added with addFont().

Parameters

font	The input font file name.
names	A list of font names

removeFont()

```
virtual void JawsMako::IDistiller::removeFont (
            const U8String & fontName )  [pure virtual]
```

Removes a named font.

Removes a font from the list of fonts to use when distilling.

Only font names added with addFont() will be removed.

Parameters

fontName	The name of the font to remove.

setAscii85EncodePages()

```
virtual void JawsMako::IDistiller::setAscii85EncodePages (
            bool ascii85EncodePages )  [pure virtual]
```

Set whether to ASCII85-encode streams. The default is false.

Equivalent to calling setParameter() with the parameter name "ASCII85Encode" with a boolean string ("true" or "false").

Parameters

ascii85EncodePages	Use ASCII85 encoding?

setAutoRotatePages()

```
virtual void JawsMako::IDistiller::setAutoRotatePages (
            bool autoRotatePages )  [pure virtual]
```

Set whether pages should be automatically rotated. The default is false.

If true, pages are automatically rotated based on the prevalent text direction on the page.

Equivalent to calling setParameter() with the parameter name "AutoRotatePages" with a boolean string ("true" or "false").

```
    @param  autoRotatePages Auto-rotate pages?
```

setColorImageCompression()

```
virtual void JawsMako::IDistiller::setColorImageCompression (
            eImageCompression compression )  [pure virtual]
```

Set the image compression for color images. The default is eICFlate.
CCITT is not allowed for color images.

Note

This is advisory only and may not be honored in all cases.

Equivalent to calling setParameter() with the parameter name "ColorImageCompression" with appropriate values.

Parameters

compression	The desired compression.

setColorImageDownsampling()

```
virtual void JawsMako::IDistiller::setColorImageDownsampling (
            eImageDownsamplingMethod method )  [pure virtual]
```

Set the desired downsampling method for color images. The default method is eIDNone (no downsampling).

Equivalent to calling setParameter() with the parameter name "ColorImageDownsampling".

Parameters

method	The method to use when downsampling.

setColorImageResolution()

```
virtual void JawsMako::IDistiller::setColorImageResolution (
            float resolution )  [pure virtual]
```

Set the desired resolution for color images when downsampling. The default is 72.

Equivalent to calling setParameter() with the parameter name "ColorImageResolution".

Parameters

resolution	The desired output resolution, in dpi. Pass 0 to leave images unchanged.

setColorJPEGQuality()

```
virtual void JawsMako::IDistiller::setColorJPEGQuality (
            eJpegQuality quality )  [pure virtual]
```

Set the JPEG quality to use when compressing color images in DCT format. Equivalent to calling setParameter() with the parameter name "ColorJPEGQuality" enumeration value as a string ("Low", "Medium", "High" or "User"). The default is eJQLow.

When set to eJQUser, color images will be compressed using the QFactor set with setColorQFactor(). eJQLow is equivalent to a QFactor of 0.1, eJQMedium to 0.5 and eJQHigh to 1.3.

Parameters

quality	The desired quality level.

setColorQFactor()

```
virtual void JawsMako::IDistiller::setColorQFactor (
            float qfactor )  [pure virtual]
```

Set the QFactor to use when compressing color images in DCT format. Applies only when setting the JPEG quality to eJQUser with setColorJPEGQuality().

The valid range of this field is from 0 to 1000000. Large numbers give better compression; small numbers give better quality.

Equivalent to calling setParameter() with the parameter name "ColorQFactor" with appropriate values.

Parameters

qfactor	The QFactor to use when compressing with eJQUser.

setCommentMonitor()

```
virtual void JawsMako::IDistiller::setCommentMonitor (
            IPSCommentMonitor * commentMonitor )  [pure virtual]
```

Set a comment monitor.

Sets a IPSCommentMonitor instance that can be used to monitor PostScript comments.

Note that this routine does not take ownership of the passed pointer, which must remain allocated while IDistiller is in use.

Parameters

commentMonitor	The comment monitor.

setCompressObjects()

```
virtual void JawsMako::IDistiller::setCompressObjects (
            bool compressObjects )  [pure virtual]
```

Set whether or not to compress individual objects.

In addition to the compression of individual objects, setting this to true allows the creation of cross reference streams, which allow the generation of PDF files larger than 9999999999 (∼9.3GB).

This setting is advisory only. In particular if writing to PDF 1.4 or earlier, object compression will not be used at all.

Equivalent to calling setParameter() with the parameter name "CompressObjects" with a boolean string ("true" or "false").

The default is false.

Parameters

compressObjects	Whether to compress individual objects.

setCompressPages()

```
virtual void JawsMako::IDistiller::setCompressPages (
            bool compressPages )  [pure virtual]
```

Set whether or not compression should be applied to page content. The default is true.

Equivalent to calling setParameter() with the parameter name "CompressPages" with a boolean string ("true" or "false").

Parameters

compressPages	Compress pages?

setConvertCMYKImagesToRGB()

```
virtual void JawsMako::IDistiller::setConvertCMYKImagesToRGB (
            bool convert )  [pure virtual]
```

Set whether to convert CMYK images to RGB. The default is false.

Equivalent to calling setParameter() with the parameter name "ConvertCMYKImagesToRGB" with a boolean string ("true" or "false").

Parameters

convert	Convert CMYK images to RGB?

setCropToBBox()

```
virtual void JawsMako::IDistiller::setCropToBBox (
            bool cropToBBox )  [pure virtual]
```

Set whether to use the BoundingBox DSC comment for the PDF CropBox. The default is false.

Equivalent to calling setParameter() with the parameter name "CropToBBox" with a boolean string ("true" or "false").

Parameters

cropToBBox	Whether to use the BoundingBox DSC comment for the PDF CropBox.

setDefaultPanoseStyle()

```
virtual void JawsMako::IDistiller::setDefaultPanoseStyle (
            const U8String & defaultPanoseStyle )  [pure virtual]
```

Sets the default Panose style to be used when emitting a CID font that is not in the Panose list, or NULL to clear. The Panose style must be a 12-byte hexadecimal string as described in the PDF reference, for example "010502020300000000000000".

Equivalent to calling setParameter() with the parameter name "DefaultPanoseStyle" with appropriate values.

Parameters

defaultPanoseStyle	The default Panose style to use.

setDocumentInformationString()

```
virtual void JawsMako::IDistiller::setDocumentInformationString (
            eDocumentInfo key,
            const U8String & value )  [pure virtual]
```

Set PDF Document Information string.

Sets the Document Information string for the given key.

setParameter() may also be used to set documentation information types, where the names are equivalent to eDocumentInfo keys. For example:

setParameter("Author", "John Smith");

setEmbedBase14Fonts()

```
virtual void JawsMako::IDistiller::setEmbedBase14Fonts (
            bool embed )  [pure virtual]
```

Set whether the base 14 PDF fonts should be forcibly embedded. Only the standard 14 fonts are not affected by this; use setEmbedFonts() to affect other fonts. The default is false.

Equivalent to calling setParameter() with the parameter name "EmbedBase14Fonts" with a boolean string ("true" or "false").

Parameters

embed	Embed 'Base 14' fonts?

setEmbedDiskBasedCidFonts()

```
virtual void JawsMako::IDistiller::setEmbedDiskBasedCidFonts (
            bool embed )  [pure virtual]
```

Set whether CID fonts obtained from resources (i.e, disk based CID fonts) can be embedded. This setting is only effective if the "EmbedFonts" parameter is set to true.

Equivalent to calling setParameter() with the parameter name "EmbedDiskBasedCidFonts" with a boolean string ("true" or "false").

Parameters

embed	Embed disk-based CID fonts?

setEmbedFonts()

```
virtual void JawsMako::IDistiller::setEmbedFonts (
            bool embed )  [pure virtual]
```

Set whether fonts should be forcibly embedded. The standard 14 fonts are not affected by this. The default is true.

Equivalent to calling setParameter() with the parameter name "EmbedFonts" with a boolean string ("true" or "false").

Parameters

embed	Embed fonts?

setEncryption()

```
virtual void JawsMako::IDistiller::setEncryption (
            const U8String & ownerPassword,
            const U8String & userPassword = "",
            uint32 permissions = 0 )  [pure virtual]
```

Set the encryption for the output PDF.

The default is no encryption.

Please note:

- Setting both a zero length user and owner password disables encryption.
- If encryption is enabled, the encryption algorithm will be chosen based on the output PDF version. If only a user password is supplied, it will also be used for the owner password.
- Permissions are only used if encryption is enabled. That is, permissions information is only stored if the PDF is to be encrypted. For 40 bit output, permissionseFillFormAllowed, eExtractionAllowed, eAssemblyAllowed and eHighQualityPrintAllowed are ignored.

Equivalent to calling setParameter with the parameter names "OwnerPassword", "UserPassword" and "↩ Permissions".

Parameters

ownerPassword	The owner password to use.
userPassword	The user password to use.
permissions	The permissions bit mask for the output PDF. (see IDOMStandardPDFSecurityInfo::ePermissionsFlags).

setEpilog()

```
virtual void JawsMako::IDistiller::setEpilog (
            const IInputStreamPtr & epilog )  [pure virtual]
```

Set a epilog stream to be consumed by the PostScript interpreter after the input stream is processed, or NULL to clear.

This stream will be both opened and closed, and will not be cloned; take care when opening multiple PostScript inputs with the same epilog simultaneously.

A epilog stream may also be set with setParameter() using either the "EpilogFile" for an epilog file name value, or "EpilogCommand" for epilog PostScript data.

Parameters

epilog	The epilog stream to use.

setFontDevicePath()

```
virtual void JawsMako::IDistiller::setFontDevicePath (
            const U8String & path )  [pure virtual]
```

Sets the font device path.

Provides a method of using a custom PostScript font device in place of the usual in-memory device used by IDistiller. A fully qualified directory path must be provided, and the directory must already exist, IDistiller will not create it automatically. The font path must contain all the fonts required for distilling, and additional fonts may be added when using addFont().

Equivalent to calling setParameter() with the parameter name "FontDevice" with the name of the path.

Parameters

path	The path name to use for the font device.

setGrayImageCompression()

```
virtual void JawsMako::IDistiller::setGrayImageCompression (
            eImageCompression compression )  [pure virtual]
```

Set the image compression for gray images. The default is eICFlate.
CCITT is not allowed for gray images.

Note

This is advisory only and may not be honored in all cases.

Equivalent to calling setParameter() with the parameter name "GrayImageCompression" with appropriate values.

Parameters

compression	The desired compression.

setGrayImageDownsampling()

```
virtual void JawsMako::IDistiller::setGrayImageDownsampling (
            eImageDownsamplingMethod method )  [pure virtual]
```

Set the desired downsampling method for gray images. The default method is eIDNone (no downsampling).

Equivalent to calling setParameter() with the parameter name "GrayImageDownsampling".

Parameters

method	The method to use when downsampling.

setGrayImageResolution()

```
virtual void JawsMako::IDistiller::setGrayImageResolution (
            float resolution )  [pure virtual]
```

Set the desired resolution for gray images when downsampling. The default is 72.

Equivalent to calling setParameter() with the parameter name "GrayImageResolution".

Parameters

resolution	The desired output resolution, in dpi. Pass 0 to leave images unchanged.

setGrayJPEGQuality()

```
virtual void JawsMako::IDistiller::setGrayJPEGQuality (
            eJpegQuality quality )  [pure virtual]
```

Set the JPEG quality to use when compressing gray images in DCT format. Equivalent to calling setParameter() with the parameter name "GrayJPEGQuality" enumeration value as a string ("Low", "Medium", "High" or "User"). The default is eJQLow.

When set to eJQUser, gray images will be compressed using the QFactor set with setGrayQFactor(). eJQLow is equivalent to a QFactor of 0.1, eJQMedium to 0.5 and eJQHigh to 1.3.

Parameters

quality	The desired quality level.

setGrayQFactor()

```
virtual void JawsMako::IDistiller::setGrayQFactor (
            float qfactor )  [pure virtual]
```

Set the QFactor to use when compressing gray images in DCT format. Applies only when setting the JPEG quality to eJQUser with setColorJPEGQuality().

The valid range of this field is from 0 to 1000000. Large numbers give better compression; small numbers give better quality.

Equivalent to calling setParameter() with the parameter name "GrayQFactor" with appropriate values.

Parameters

qfactor	The QFactor to use when compressing with eJQUser.

setHalftone()

```
virtual void JawsMako::IDistiller::setHalftone (
            bool halftone )  [pure virtual]
```

Set whether halftone information should be preserved. The default is false.

Equivalent to calling setParameter() with the parameter name "Halftone" with a boolean string ("true" or "false").

Parameters

halftone	Preserve halftones?

setHiResBBox()

```
virtual void JawsMako::IDistiller::setHiResBBox (
            bool hiResBBox )  [pure virtual]
```

Set whether to use the HiResBoundingBox DSC comment if present.

```
When set to true, and "TrimToBBox" or "CropToBBox" are set to true
the HiResBoundingBox DSC comment will be used for the page size or
CropBox instead of the BoundingBox DSC comment if it is defined in the
input job.
If the job defines no HiResBoundingBox DSC comment the BoundingBox is
used.

The default is false.

Equivalent to calling setParameter() with the parameter name
"HiResBBox" with a boolean string ("true" or "false").
```

Parameters

hiResBBox	Whether to use the HiResBoundingBox DSC comment.

setJobTicket()

```
virtual void JawsMako::IDistiller::setJobTicket (
            bool jobTicket )  [pure virtual]
```

Set whether to add a job ticket object to the output. The default is false.

If this flag is set, Portable Job Ticket information is added to the generated PDF file.

Equivalent to calling setParameter() with the parameter name "JobTicket" with a boolean string ("true" or "false").

Parameters

jobTicket	Add job ticket?

setLinearize()

```
virtual void JawsMako::IDistiller::setLinearize (
            bool linearize )  [pure virtual]
```

Set whether the output should be linearized. The default is false.

When true, a PDF optimized for byte serving (for example from a web server) will be produced. This will generally be a slow operation.

Equivalent to calling setParameter() with the parameter name "Linearize" with a boolean string ("true" or "false").

Parameters

linearize	Linearize?

setMonoImageCompression()

```
virtual void JawsMako::IDistiller::setMonoImageCompression (
            eImageCompression compression )  [pure virtual]
```

Set the image compression for monochrome images. The default is eICCCITT.
JPEG/DCT and auto are not allowed for mono images.

Note

This is advisory only and may not be honored in all cases.

Equivalent to calling setParameter() with the parameter name "MonoImageCompression" with appropriate values.

Parameters

compression	The desired compression.

setMonoImageDownsampling()

```
virtual void JawsMako::IDistiller::setMonoImageDownsampling (
            eImageDownsamplingMethod method )  [pure virtual]
```

Set the desired downsampling method for monochrome images. The default method is eIDNone (no downsampling).

Note

Bicubic downsampling is not supported for monochrome images and average downsampling will be used when setting the method to eIDBicubic.

Equivalent to calling setParameter() with the parameter name "MonoImageDownsampling".

Parameters

method	The method to use when downsampling.

setMonoImageResolution()

```
virtual void JawsMako::IDistiller::setMonoImageResolution (
            float resolution )  [pure virtual]
```

Set the desired resolution for monochrome images when downsampling. The default is 72.

Equivalent to calling setParameter() with the parameter name "MonoImageResolution".

Parameters

resolution	The desired output resolution, in dpi. Pass 0 to leave images unchanged.

setOPI()

```
virtual void JawsMako::IDistiller::setOPI (
            bool opi )  [pure virtual]
```

Set whether OPI comments should be preserved. The default is false.

Equivalent to calling setParameter() with the parameter name "OPI" with a boolean string ("true" or "false").

Parameters

opi	Preserve OPI comments?

setOverprint()

```
virtual void JawsMako::IDistiller::setOverprint (
            bool overprint )  [pure virtual]
```

Set whether overprinting should be preserved. The default is false.

Equivalent to calling setParameter() with the parameter name "Overprint" with a boolean string ("true" or "false").

Parameters

overprint	Preserve overprinting?

setOverprintMode()

```
virtual void JawsMako::IDistiller::setOverprintMode (
            bool overprintMode )  [pure virtual]
```

Set the overprint mode. The default is true, which sets non-zero overprint mode (/OPM 1).

When set to false /OPM 0 is used when overprinting.

Equivalent to calling setParameter() with the parameter name "OverprintMode" with a boolean string ("true" or "false").

Parameters

overprintMode	Use non-zero overprint mode?

setPageFilterHigh()

```
virtual void JawsMako::IDistiller::setPageFilterHigh (
            uint64 pageHigh )  [pure virtual]
```

Set upper page limit.

Sets the upper limit of pages to be included in the output. Pages are numbered starting with page 1. If set to a non-zero value, all input data following that page will be silently discarded. If set to 0, no upper page limit is set.

Equivalent to calling setParameter() with the parameter name "PageFilterHigh" with the page number as a string.

The default is 0.

Parameters

pageHigh	The upper page limit.

setPageFilterLow()

```
virtual void JawsMako::IDistiller::setPageFilterLow (
            uint64 pageLow )  [pure virtual]
```

Set lower page limit.

Sets the lower limit of pages to be included in the output. Pages are numbered starting with page 1. If set to 0, no lower page limit is set.

Equivalent to calling setParameter() with the parameter name "PageFilterLow" with the page number as a string.

The default is 0.

Parameters

pageLow	The lower page limit.

setPageFilterRange()

```
virtual void JawsMako::IDistiller::setPageFilterRange (
            uint64 pageLow,
            uint64 pageHigh )  [pure virtual]
```

Set page range.

Convenience function to set the pages to be included in the output. Equivalent to calling setPageFilterLow() and setPageFilterHigh() with lower and upper page limits.

The default for each parameter is 0.

Parameters

pageLow	The lower page limit.
pageHigh	The upper page limit.

setPanose()

```
virtual void JawsMako::IDistiller::setPanose (
            const IInputStreamPtr & panose )  [pure virtual]
```

Sets a stream that refers to a list of CID font names and their associated Panose styles.

Each line of the list must consist of two fields that are tab delimited. The first field is the font name. The second field is the complete string representing the Panose Style entry. For example:

GothicBBBPr6-Medium<tab>0801020b0500000000000000 GothicBBB-Medium<tab>0801020b0500000000000000

If set, the default Panose style will be used when Jaws emits a CID font that is not in the list.

Equivalent to calling setParameter() with the parameter name "Panose" with appropriate values.

Parameters

panose	The Panose stream.

setParameter()

```
virtual void JawsMako::IDistiller::setParameter (
            const U8String & param,
            const U8String & value )  [pure virtual]
```

Apply a key-value pair output parameter with a string value. The parameter name is case insensitive. Please refer to the supplied documentation for the details of the available parameters and their ranges.

Parameters

param	The parameter to change
value	The value to set

setPdfVersion()

```
virtual void JawsMako::IDistiller::setPdfVersion (
            ePDFVersion version )  [pure virtual]
```

Set the PDF version to generate.

Valid versions supported are:

- 1.3
- 1.4
- 1.5
- 1.6
- 1.7

Equivalent to calling setParameter with the parameter name "PDFVersion" with the value of the version as a string (e.g. "1.6"). The default is 1.3.

Parameters

version	The PDF version to generate.

setProlog()

```
virtual void JawsMako::IDistiller::setProlog (
            const IInputStreamPtr & prolog )  [pure virtual]
```

Set a prolog stream to be consumed by the PostScript interpreter before the input stream is processed, or NULL to clear.

This stream will be both opened and closed, and will not be cloned; take care when opening multiple PostScript inputs with the same prolog simultaneously.

A prolog stream may also be set with setParameter() using either the "PrologFile" for a prolog file name value, or "PrologCommand" for prolog PostScript data.

Parameters

prolog	The prolog stream to use.

setResolution()

```
virtual void JawsMako::IDistiller::setResolution (
            float resolution )  [pure virtual]
```

Set the target resolution for the output, in dots per inch. The default is 288. Equivalent to calling setParameter with the parameter name "Resolution".

Parameters

resolution	The desired resolution.

setResourceDevicePath()

```
virtual void JawsMako::IDistiller::setResourceDevicePath (
            const U8String & path )  [pure virtual]
```

Sets the resource device path.

Provides a method of using a custom PostScript resource device in place of the usual in-memory device used by IDistiller. A fully qualified directory path must be provided, and the directory must already exist, IDistiller will not create it automatically. The resource path must contain all the PostScript required for distilling, and additional fonts may be added when using addFont().

Equivalent to calling setParameter() with the parameter name "ResourceDevice" with the name of the path.

Parameters

path	The path name to use for the resource device.

setSubsetFonts()

```
virtual void JawsMako::IDistiller::setSubsetFonts (
            bool subset )  [pure virtual]
```

Set whether embedded fonts should be subset or not. The default is true.

This is a preference only. Some font types may require subsetting based on context.

Equivalent to calling setParameter() with the parameter name "SubsetFonts" with a boolean string ("true" or "false").

Parameters

subset	Subset fonts?

setThumbnails()

```
virtual void JawsMako::IDistiller::setThumbnails (
            eThumbnailType thumbnails )  [pure virtual]
```

Set the PDF thumbnail type.

```
Equivalent to calling setParameter with the parameter
name "thumbnails" with the enumeration value as a string
("None", "Color" or "Mono").  The default is eTHNone.
```

Parameters

thumbnails	The desired thumbnail type.

setTransfers()

```
virtual void JawsMako::IDistiller::setTransfers (
            eTransferFunctionMethod transfers )  [pure virtual]
```

Set the desired method to for controlling transfer functions.

```
Equivalent to calling setParameter with the parameter
name "Transfers" with the enumeration value as a string
("Apply", "Remove" or "Preserve"). The default is eTRApply.
```

Parameters

transfers	The desired transfer method.

setTrimToBBox()

```
virtual void JawsMako::IDistiller::setTrimToBBox (
            bool trimToBBox )  [pure virtual]
```

Set whether to use the BoundingBox DSC comment for the page size. The default is false.

Equivalent to calling setParameter() with the parameter name "TrimToBBox" with a boolean string ("true" or "false").

Parameters

trimToBBox	Whether to use the BoundingBox DSC comment for the page size.

setUseDeviceDependentColor()

```
virtual void JawsMako::IDistiller::setUseDeviceDependentColor (
            bool useDeviceDependentColor )  [pure virtual]
```

Set whether to convert device independent colors to device dependent colors The default is false.

Equivalent to calling setParameter() with the parameter name "UseDeviceDependentColor" with a boolean string ("true" or "false").

Parameters

useDeviceDependentColor	Use convert?

The documentation for this class was generated from the following file:

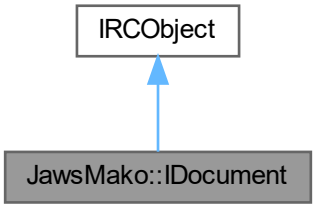
- distiller.h

7.155 JawsMako::IDocument Class Reference

A document from an IDocumentAssembly, allowing for high level document and page mangement, and providing on-demand lazy loading of page markup.

```
#include <jawsmako.h>
```

Inheritance diagram for JawsMako::IDocument:



Public Member Functions

- virtual uint32 getNumPages ()=0

  Return the number of pages in the document, if known.
- virtual IPagePtr getPage (uint32 index=0)=0

  Get the IPage from the document at the given index.
- virtual bool pageExists (uint32 index)=0

  Determine if a page with the given index exists in the document.
- virtual void insertPage (const IPagePtr &page, uint32 index=0, const IDocumentPtr &source↩ Document=IDocumentPtr())=0

  Insert a page into the document at the given index. Note that the page will not be cloned, and so any changes to the added page will affect all users of the page. The interactive forms will be updated with any widget annotations present on the page. Note that if the names of any widgets being added clash with existing widgets or fields they will be forcibly renamed. Provide a sourceDocument if possible for improved form merging.
- virtual void appendPage (const IPagePtr &page, const IDocumentPtr &sourceDocument=IDocumentPtr())=0

Append a page to the document.

- virtual void removePage (uint32 index)=0

  Remove the page at the given index.

- virtual void **removePage** (const IPagePtr &page)=0

  Remove the given page from the document. If the page is not present, ane exception will result.

- virtual void **findTarget** (DOMid targetId, uint32 &pageNum)=0

  Find the page contatining the target with the given DOMid in the document, providing the index of the page within the document. Throws an IError if the target could not be found.

- virtual IAnnotationPtr **findAnnotation** (const IAnnotationReferencePtr &reference)=0

  Find the annotation with the given annotation reference within the document. Throws an IError if the target could not be found.

- virtual IAnnotationPtr **findAnnotation** (const IAnnotationReferencePtr &reference, uint32 &pageNum)=0

  Find the annotation with the given annotation reference within the document, providing the index of the page that contains the annotation. Throws an IError if the target could not be found.

- virtual IDOMJobTkPtr getJobTicket () const =0

  Get the document job ticket, if present.

- virtual void **setJobTicket** (const IDOMJobTkPtr &jobTicket)=0

  Set the document job ticket.

- virtual COutputIntentVect **getOutputIntents** () const =0

  Get the output intents, if present.

- virtual IOptionalContentPtr **getOptionalContent** () const =0

  Get the optional content if present.

- virtual void **setOptionalContent** (const IOptionalContentPtr &optionalContent)=0

  Set the optional content for the document, or NULL if if should be removed.

- virtual IStructurePtr **getStructure** () const =0

  Get the structure information if present.

- virtual void **setStructure** (const IStructurePtr &structure)=0

  Set the structure content for the document, or NULL if if should be removed.

- virtual IDOMOutlinePtr getOutline () const =0

  Get the document outline, if present.

- virtual void setOutline (const IDOMOutlinePtr &outline)=0

  Set the document outline, if present.

- virtual IFormPtr getForm () const =0

  Get the document interactive form, if present.

- virtual void **setForm** (const IFormPtr &form)=0

  Set the document interactive form. May be NULL.

- virtual CFileSpecAsEmbeddedDataVect **getEmbeddedStreams** ()=0

  Get any embedded streams or attachments attached to the document. This is currently a PDF-specific feature.

- virtual void addEmbeddedStream (const IFileSpecAsEmbeddedDataPtr &embeddedData)=0

  Add an embedded file stream to the document. This is currently a PDF-specific feature.

- virtual CNamedDestinationVect **getNamedDestinations** ()=0

  Get any named destinations present in the document.

- virtual void addNamedDestination (const INamedDestinationPtr &namedDestination)=0

  Add a named destination to the document. Note that this will override any named destination with the same name that may be present.

- virtual void setNamedDestinations (const CNamedDestinationVect &namedDestinations)=0

  Replace the named destinations in the document with the given vector. Note that if there are multiple destinations with the same name, the results are undefined.

- virtual IThreadsPtr getThreads () const =0

  Get the document threads, if present.

- virtual IDocumentPtr clone ()=0

  Clone an IDocument. Will also clone all the pages in the document.

- virtual IPDFObjectPtr lookupFarReference (const IPDFFarReferencePtr &farReference, IPDFObjectStorePtr &store)=0

  Attempt to find and resolve an indirect far reference to a PDF object. If found, the object store that contained it will be provided. If not found at the document level, the documents' pages will be searched. Objects which are controlled by formal Mako APIs may not be reachable, and may be overridden at output.
- virtual IPDFObjectStorePtr getObjectStore ()=0

  Obtain access to the document level object store. Objects which are controlled by formal Mako APIs may not be reachable, and may be overridden at output.
- virtual IPDFObjectPtr readPdfObject (const IPDFReferencePtr &reference)=0

  Raw access to the PDF object data base from an original PDF file. For informational purposes only; the objects returned from this interface must not be edited.
- virtual IPDFDictionaryPtr readPdfTrailerDictionary ()=0

  Raw access to the PDF trailer dictionary. For informational purposes only; the object returned from this interface must not be edited.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IDocumentPtr create (const IJawsMakoPtr &jawsMako, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create an empty document.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.155.1 Detailed Description

A document from an IDocumentAssembly, allowing for high level document and page mangement, and providing on-demand lazy loading of page markup.

7.155.2 Member Function Documentation

addEmbeddedStream()

```
virtual void JawsMako::IDocument::addEmbeddedStream (
            const IFileSpecAsEmbeddedDataPtr & embeddedData )  [pure virtual]
```

Add an embedded file stream to the document. This is currently a PDF-specific feature.

Parameters

embeddedData	The embedded file stream to be added.

addNamedDestination()

```
virtual void JawsMako::IDocument::addNamedDestination (
            const INamedDestinationPtr & namedDestination )  [pure virtual]
```

Add a named destination to the document. Note that this will override any named destination with the same name that may be present.

Parameters

namedDestination	The named destination to be added

appendPage()

```
virtual void JawsMako::IDocument::appendPage (
            const IPagePtr & page,
            const IDocumentPtr & sourceDocument = IDocumentPtr() )  [pure virtual]
```

Append a page to the document.

Parameters

page	smart pointer to the page to be inserted.
sourceDocument	(optional) the source document that the page came from. If provided, the target document's outline will be update with any outline entries present on the inserted page. The interactive forms will be updated with any widget annotations present on the page. Note that if the names of any widgets being added clash with existing widgets or fields they will be forcibly renamed. Provide a sourceDocument if possible for improved form merging.

clone()

```
virtual IDocumentPtr JawsMako::IDocument::clone ( )  [pure virtual]
```

Clone an IDocument. Will also clone all the pages in the document.

Returns

IDocumentPtr The clone.

create()

```
static JAWSMAKO_API IDocumentPtr JawsMako::IDocument::create (
            const IJawsMakoPtr & jawsMako,
            const IProgressMonitorPtr & progressMonitor = IProgressMonitorPtr() )  [static]
```

Create an empty document.

Parameters

jawsMako	The JawsMako instance.
progressMonitor	A smart pointer to an IProgressMonitor object or NULL if no object was passed.

Returns

IDocumentPtr the new page.

getForm()

```
virtual IFormPtr JawsMako::IDocument::getForm ( ) const  [pure virtual]
```

Get the document interactive form, if present.

Returns

IFormPtr The document form, or NULL if not present.

getJobTicket()

```
virtual IDOMJobTkPtr JawsMako::IDocument::getJobTicket ( ) const  [pure virtual]
```

Get the document job ticket, if present.

Returns

IDOMJobTkPtr the job ticket, or NULL if not present

getNumPages()

```
virtual uint32 JawsMako::IDocument::getNumPages ( )  [pure virtual]
```

Return the number of pages in the document, if known.

Returns

uint32 The number of pages in the document. Note: For streaming inputs such as PCL/5, this will cause the input to block until the document has been consumed and the number of pages known.

getObjectStore()

```
virtual IPDFObjectStorePtr JawsMako::IDocument::getObjectStore ( )  [pure virtual]
```

Obtain access to the document level object store. Objects which are controlled by formal Mako APIs may not be reachable, and may be overridden at output.

Returns

IPDFObjectStorePtr The object store for the document.

getOutline()

```
virtual IDOMOutlinePtr JawsMako::IDocument::getOutline ( ) const  [pure virtual]
```

Get the document outline, if present.

Returns

IDOMOutlinePtr the document outline, or NULL if not present.

getPage()

```
virtual IPagePtr JawsMako::IDocument::getPage (
            uint32 index = 0 )  [pure virtual]
```

Get the IPage from the document at the given index.

Returns

IPagePtr the requested page, 0 being the first page.

getThreads()

```
virtual IThreadsPtr JawsMako::IDocument::getThreads ( ) const  [pure virtual]
```

Get the document threads, if present.

Returns

IThreadsPtr The document threads, if present.

insertPage()

```
virtual void JawsMako::IDocument::insertPage (
            const IPagePtr & page,
            uint32 index = 0,
            const IDocumentPtr & sourceDocument = IDocumentPtr() )  [pure virtual]
```

Insert a page into the document at the given index. Note that the page will not be cloned, and so any changes to the added page will affect all users of the page. The interactive forms will be updated with any widget annotations present on the page. Note that if the names of any widgets being added clash with existing widgets or fields they will be forcibly renamed. Provide a sourceDocument if possible for improved form merging.

Parameters

page	smart pointer to the page to be inserted.
index	The position in the document to insert the page, 0 being the first position.
sourceDocument	(optional) the source document that the page came from. If provided, the target document's outline will be update with any outline entries present on the inserted page.

lookupFarReference()

```
virtual IPDFObjectPtr JawsMako::IDocument::lookupFarReference (
            const IPDFFarReferencePtr & farReference,
            IPDFObjectStorePtr & store )  [pure virtual]
```

Attempt to find and resolve an indirect far reference to a PDF object. If found, the object store that contained it will be provided. If not found at the document level, the documents' pages will be searched. Objects which are controlled by formal Mako APIs may not be reachable, and may be overridden at output.

Parameters

farReference	The far reference to resolve.
store	A reference to receive the store where the object was located.

Returns

IPDFObject The resolved object, or NULL if the resolved object was not found.

pageExists()

```
virtual bool JawsMako::IDocument::pageExists (
            uint32 index )  [pure virtual]
```

Determine if a page with the given index exists in the document.

Parameters

index	The index of the page to check, 0 being the first page.

Returns

bool True if the document has a page with the given index.

readPdfObject()

```
virtual IPDFObjectPtr JawsMako::IDocument::readPdfObject (
            const IPDFReferencePtr & reference )  [pure virtual]
```

Raw access to the PDF object data base from an original PDF file. For informational purposes only; the objects returned from this interface must not be edited.

Parameters

reference	The indirect reference to be retrieved, or NULL to read the document's Catalog dictionary.

Returns

IPDFObjectPtr The read object, or NULL if there is no such object.

readPdfTrailerDictionary()

```
virtual IPDFDictionaryPtr JawsMako::IDocument::readPdfTrailerDictionary ( )  [pure virtual]
```

Raw access to the PDF trailer dictionary. For informational purposes only; the object returned from this interface must not be edited.

Returns

IPDFDictionaryPtr The trailer dictionary, or null if no trailer dictionary is available.

removePage()

```
virtual void JawsMako::IDocument::removePage (
            uint32 index )  [pure virtual]
```

Remove the page at the given index.

Parameters

index	The index of the page to be removed (0 being the first page)

setNamedDestinations()

```
virtual void JawsMako::IDocument::setNamedDestinations (
            const CNamedDestinationVect & namedDestinations )  [pure virtual]
```

Replace the named destinations in the document with the given vector. Note that if there are multiple destinations with the same name, the results are undefined.

Parameters

namedDestinations	The vector of named destinations

setOutline()

```
virtual void JawsMako::IDocument::setOutline (
            const IDOMOutlinePtr & outline )  [pure virtual]
```

Set the document outline, if present.

Parameters

outline	The desired document outline, or NULL to set no outline.

The documentation for this class was generated from the following file:

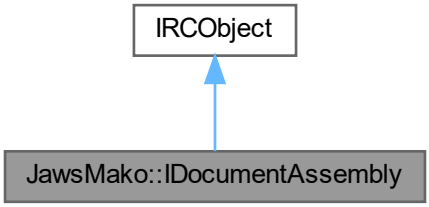
- jawsmako.h

7.156 JawsMako::IDocumentAssembly Class Reference

A self contained collection of IDocuments.

`#include <jawsmako.h>`

Inheritance diagram for JawsMako::IDocumentAssembly:



Public Member Functions

- virtual uint32 getNumDocuments ()=0

  Return the number of documents in the assembly, if known. Note: For streaming inputs such as PCL/5, this will cause the input to block until the stream has been exhausted and the number of documents known.
- virtual IDocumentPtr getDocument (uint32 index=0)=0

  Get the IDocument from the assembly at the given index.
- virtual bool documentExists (uint32 index)=0

  Determine if a document with the given index exists in the assembly.
- virtual void insertDocument (const IDocumentPtr &document, uint32 index=0)=0

  Insert a document into the assembly at the given index.
- virtual void removeDocument (uint32 index)=0

  Remove the document at the given index.
- virtual void **removeDocument** (const IDocumentPtr &document)=0

  Remove the given document from the document. If the document is not present, an exception will be raised.
- virtual void appendDocument (const IDocumentPtr &document)=0

  Append a document to the assembly.
- virtual IDOMJobTkPtr getJobTicket () const =0

  Get the overall job ticket for the entire assembly, if present.
- virtual void **setJobTicket** (const IDOMJobTkPtr &jobTicket)=0

Set the document job ticket.

- virtual void **findTarget** (DOMid targetId, uint32 &docNum, uint32 &pageNum)=0

  Find the page contatining the target with the given DOMid in the assembly, providing the index of the document and page within the assembly. Throws an IError if the target could not be found.
- virtual IAnnotationPtr **findAnnotation** (const IAnnotationReferencePtr &reference)=0

  Find the annotation with the given annotation reference within the document assembly. Throws an IError if the target could not be found.
- virtual IAnnotationPtr **findAnnotation** (const IAnnotationReferencePtr &reference, uint32 &docNum, uint32 &pageNum)=0

  Find the annotation with the given annotation reference within the document assembly, providing the index of the document and page that contains the annotation. Throws an IError if the target could not be found.
- virtual IDocumentAssemblyPtr clone ()=0

  Clone an IDocumentAssembly. Will also clone all the documents and pages in the assembly.
- virtual IDOMMetadataPtr getJobMetadata () const =0

  Get the job metadata, if present (ie the document properties).
- virtual void **setJobMetadata** (const IDOMMetadataPtr &metadata)=0

  Set the job metadata.
- virtual IInputStreamPtr getXmpPacket () const =0

  Get the PDF XMP packet, if present.
- virtual void **setXmpPacket** (const IInputStreamPtr &xmpPacket)=0

  Set the PDF XMP packet, if present. Note that the XMP packet in any PDF output will be modified to match the job metadata for compliance with specification. If NULL, the XMP packet will be reset.
- virtual IDOMSecurityInfoPtr getSecurityInfo () const =0

  Get the security information that applied to the source file/stream. Curently only relevant for PDF.
- virtual IDOMImagePtr getThumbnail () const =0

  Get the thumbnail image for the assembly.
- virtual void setThumbnail (const IDOMImagePtr &thumbnail)=0

  Sets a new thumbnail image for the document assembly. The image must be in either JPEG or PNG format. Setting thumbnail to NULL deletes the document assembly thumbnail.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IDocumentAssemblyPtr create (const IJawsMakoPtr &jawsMako, const IProgress↩ MonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create a new, empty document assembly.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.156.1 Detailed Description

A self contained collection of IDocuments.

7.156.2 Member Function Documentation

appendDocument()

```
virtual void JawsMako::IDocumentAssembly::appendDocument (
            const IDocumentPtr & document )  [pure virtual]
```

Append a document to the assembly.

Parameters

document	smart pointer to the page to be appended.

clone()

```
virtual IDocumentAssemblyPtr JawsMako::IDocumentAssembly::clone ( )  [pure virtual]
```

Clone an IDocumentAssembly. Will also clone all the documents and pages in the assembly.

Returns

IDocumentAssemblyPtr the clone.

create()

```
static JAWSMAKO_API IDocumentAssemblyPtr JawsMako::IDocumentAssembly::create (
            const IJawsMakoPtr & jawsMako,
            const IProgressMonitorPtr & progressMonitor = IProgressMonitorPtr() )  [static]
```

Create a new, empty document assembly.

Parameters

jawsMako	The JawsMako instance.
progressMonitor	The progress monitor which allows aborting an operation or registering a progress callback.

Returns

IDocumentAssemblyPtr the new assembly.

documentExists()

```
virtual bool JawsMako::IDocumentAssembly::documentExists (
```

```
            uint32 index )  [pure virtual]
```

Determine if a document with the given index exists in the assembly.

Parameters

index	The index of the document to check, 0 being the first document.

Returns

bool True if the assembly has a document with the given index.

getDocument()

```
virtual IDocumentPtr JawsMako::IDocumentAssembly::getDocument (
            uint32 index = 0 )  [pure virtual]
```

Get the IDocument from the assembly at the given index.

Returns

IDocumentPtr the requested document, 0 being the first document.

getJobMetadata()

```
virtual IDOMMetadataPtr JawsMako::IDocumentAssembly::getJobMetadata ( ) const  [pure virtual]
```

Get the job metadata, if present (ie the document properties).

Returns

IDOMMetadataPtr. Returns NULL if no such information was provided by the input.

getJobTicket()

```
virtual IDOMJobTkPtr JawsMako::IDocumentAssembly::getJobTicket ( ) const  [pure virtual]
```

Get the overall job ticket for the entire assembly, if present.

Returns

IDOMJobTkPtr the job ticket, or NULL if not present

getNumDocuments()

```
virtual uint32 JawsMako::IDocumentAssembly::getNumDocuments ( )  [pure virtual]
```

Return the number of documents in the assembly, if known. Note: For streaming inputs such as PCL/5, this will cause the input to block until the stream has been exhausted and the number of documents known.

Returns

uint32 The number of pages in the assembly.

getSecurityInfo()

```
virtual IDOMSecurityInfoPtr JawsMako::IDocumentAssembly::getSecurityInfo ( ) const  [pure
virtual]
```

Get the security information that applied to the source file/stream. Curently only relevant for PDF.

Returns

IDOMPDFSecurityInfoPtr. Returns NULL if no security information was provided by the input.

getThumbnail()

```
virtual IDOMImagePtr JawsMako::IDocumentAssembly::getThumbnail ( ) const  [pure virtual]
```

Get the thumbnail image for the assembly.

Returns

IDOMImagePtr. Returns NULL if no thumbnail image is present.

getXmpPacket()

```
virtual IInputStreamPtr JawsMako::IDocumentAssembly::getXmpPacket ( ) const  [pure virtual]
```

Get the PDF XMP packet, if present.

Returns

IInputStreamPtr. Returns NULL if the XMP packet is not present.

insertDocument()

```
virtual void JawsMako::IDocumentAssembly::insertDocument (
            const IDocumentPtr & document,
            uint32 index = 0 )  [pure virtual]
```

Insert a document into the assembly at the given index.

Parameters

document	smart pointer to the document to be inserted.
index	The position in the assembly to insert the document, 0 being the first position.

removeDocument()

```
virtual void JawsMako::IDocumentAssembly::removeDocument (
            uint32 index )  [pure virtual]
```

Remove the document at the given index.

Parameters

index	The index of the document to be removed (0 being the first document).

setThumbnail()

```
virtual void JawsMako::IDocumentAssembly::setThumbnail (
            const IDOMImagePtr & thumbnail )  [pure virtual]
```

Sets a new thumbnail image for the document assembly. The image must be in either JPEG or PNG format. Setting thumbnail to NULL deletes the document assembly thumbnail.

Parameters

thumbnail	Smart pointer to new thumbnail image.

The documentation for this class was generated from the following file:

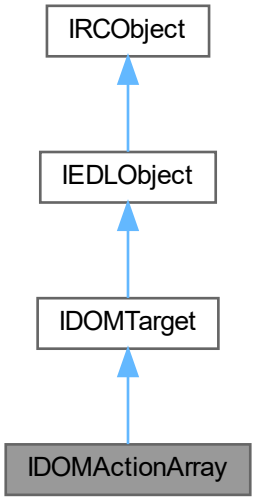
- jawsmako.h

7.157 IDOMActionArray Class Reference

IDOMActionArray interface.

```
#include <idomtarget.h>
```

Inheritance diagram for IDOMActionArray:



Public Member Functions

- virtual IDOMTargetCollectionEnumPtr getActionsEnum ()=0

  Retrieves enumerator of the action collection Action collection is analog of entry "Next" of PDF action dictionary.
- virtual uint32 getActionsCount ()=0

  Retrieves count of the actions.
- virtual void **clearActions** ()=0

  Clears collection of actions.
- virtual bool addAction (const IDOMTargetPtr &ptrAction)=0

  Append a action to the action collection.
- virtual eTargetType getTargetType () const

  Implementation of geTargetType for IDOMActionArray.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Public Types inherited from IDOMTarget

- enum eTargetType {
  eExternal , eInternal , ePage , ePageRect ,
  eActionGoToR , eActionGoToE , eActionLaunch , eActionThread ,
  eActionSound , eActionMovie , eActionHide , eActionNamed ,
  eActionSubmitForm , eActionResetForm , eActionImportData , eActionJavaScript ,
  eActionSetOCGState , eActionRendition , eActionTrans , eActionGoTo3DView ,
  eActionArray }

  An enumeration of target types.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.157.1 Detailed Description

IDOMActionArray interface.

7.157.2 Member Function Documentation

addAction()

```
virtual bool IDOMActionArray::addAction (
            const IDOMTargetPtr & ptrAction )  [pure virtual]
```

Append a action to the action collection.

Parameters

ptrAction	The smart pointer to the target interface

Returns

bool Returns true on success

getActionsCount()

```
virtual uint32 IDOMActionArray::getActionsCount ( )  [pure virtual]
```

Retrieves count of the actions.

Returns

uint32 Size of action collection

getActionsEnum()

```
virtual IDOMTargetCollectionEnumPtr IDOMActionArray::getActionsEnum ( )  [pure virtual]
```

Retrieves enumerator of the action collection Action collection is analog of entry "Next" of PDF action dictionary.

Returns

IDOMActionCollectionEnumPtr Actions enumerator

getTargetType()

```
virtual eTargetType IDOMActionArray::getTargetType ( ) const  [inline], [virtual]
```

Implementation of geTargetType for IDOMActionArray.

Returns

eTargetType Returns eActionArray;

Implements IDOMTarget.

The documentation for this class was generated from the following file:

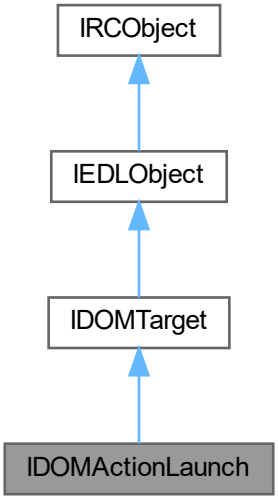
- idomtarget.h

7.158 IDOMActionLaunch Class Reference

IDOMActionLaunch interface.

`#include <idomtarget.h>`

Inheritance diagram for IDOMActionLaunch:



Public Member Functions

- virtual EDLString getFile () const =0
  Retrieves the file specification.
- virtual void setFile (const EDLString &file)=0
  Sets the file specification.
- virtual EDLString getWinParameters () const =0
  Retrieves the Windows-specific launch parameters.
- virtual void setWinParameters (const EDLString &params)=0
  Sets the Windows-specific launch parameters.
- virtual EDLString getMacParameters () const =0
  Retrieves the Mac OS-specific launch parameters.
- virtual void setMacParameters (const EDLString &params)=0
  Sets the Mac OS-specific launch parameters.
- virtual EDLString getUnixParameters () const =0
  Retrieves the UNIX-specific launch parameters.
- virtual void setUnixParameters (const EDLString &params)=0
  Sets the UNIX-specific launch parameters.
- virtual bool getNewWindow () const =0
  Getter for NewWindow flag.
- virtual void setNewWindow (bool bNewWindow)=0
  Setter for NewWindow flag.
- virtual eTargetType getTargetType () const
  Implementation of geTargetType for IDOMActionLaunch.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Public Types inherited from IDOMTarget

- enum eTargetType {
  eExternal , eInternal , ePage , ePageRect ,
  eActionGoToR , eActionGoToE , eActionLaunch , eActionThread ,
  eActionSound , eActionMovie , eActionHide , eActionNamed ,
  eActionSubmitForm , eActionResetForm , eActionImportData , eActionJavaScript ,
  eActionSetOCGState , eActionRendition , eActionTrans , eActionGoTo3DView ,
  eActionArray }

  An enumeration of target types.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.158.1 Detailed Description

IDOMActionLaunch interface.

7.158.2 Member Function Documentation

getFile()

```
virtual EDLString IDOMActionLaunch::getFile ( ) const  [pure virtual]
```

Retrieves the file specification.

Returns

EDLSysString The file specification

getMacParameters()

```
virtual EDLString IDOMActionLaunch::getMacParameters ( ) const  [pure virtual]
```

Retrieves the Mac OS-specific launch parameters.

Returns

EDLSysString The launch parameters

getNewWindow()

```
virtual bool IDOMActionLaunch::getNewWindow ( ) const  [pure virtual]
```

Getter for NewWindow flag.

Returns

bool Returns the value of the NewWindow flag

getTargetType()

```
virtual eTargetType IDOMActionLaunch::getTargetType ( ) const  [inline], [virtual]
```

Implementation of geTargetType for IDOMActionLaunch.

Returns

eTargetType Returns eActionLaunch;

Implements IDOMTarget.

getUnixParameters()

```
virtual EDLString IDOMActionLaunch::getUnixParameters ( ) const  [pure virtual]
```

Retrieves the UNIX-specific launch parameters.

Returns

EDLSysString The launch parameters

getWinParameters()

```
virtual EDLString IDOMActionLaunch::getWinParameters ( ) const  [pure virtual]
```

Retrieves the Windows-specific launch parameters.

Returns

EDLSysString The launch parameters

setFile()

```
virtual void IDOMActionLaunch::setFile (
            const EDLString & file )  [pure virtual]
```

Sets the file specification.

Parameters

file	File specification

setMacParameters()

```
virtual void IDOMActionLaunch::setMacParameters (
            const EDLString & params )  [pure virtual]
```

Sets the Mac OS-specific launch parameters.

Parameters

params	The launch parameters

setNewWindow()

```
virtual void IDOMActionLaunch::setNewWindow (
            bool bNewWindow )  [pure virtual]
```

Setter for NewWindow flag.

Parameters

bNewWindow	New value of the NewWindow flag

setUnixParameters()

```
virtual void IDOMActionLaunch::setUnixParameters (
            const EDLString & params )  [pure virtual]
```

Sets the UNIX-specific launch parameters.

Parameters

params	The launch parameters

setWinParameters()

```
virtual void IDOMActionLaunch::setWinParameters (
            const EDLString & params )  [pure virtual]
```

Sets the Windows-specific launch parameters.

Parameters

params	The launch parameters

The documentation for this class was generated from the following file:

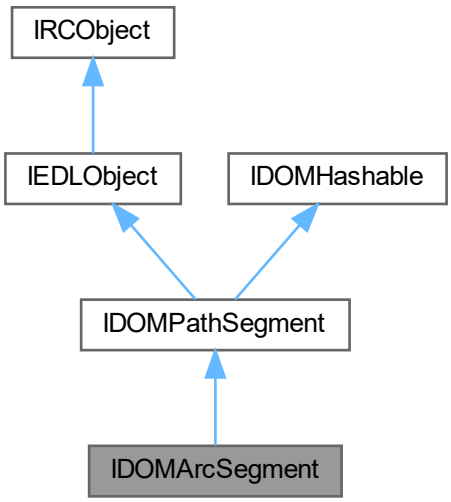
- idomtarget.h

7.159 IDOMArcSegment Class Reference

Interface to Arc Segment element.

`#include <idompathgeometry.h>`

Inheritance diagram for IDOMArcSegment:



Classes

- class Data

  Initialization data.

Public Types

- enum eSweepDirection

  The direction of the sweep-clockwise or counter clockwise-between the start and end points.

Public Member Functions

- virtual const FPoint & getPoint () const =0

  Retrieves the end point of the arc.
- virtual void setPoint (const FPoint &pt)=0

  Sets the end point of the arc. An exception is thrown if this segment is immutable.
- virtual double getRadiusX () const =0

  Retrieves the x-radius of the arc.
- virtual void setRadiusX (double radiusX)=0

  Sets the x-radius of the base ellipses. An exception is thrown if this segment is immutable.
- virtual double getRadiusY () const =0

  Retrieves the y-radius of the base ellipses.
- virtual void setRadiusY (double radiusY)=0

  Sets the y-radius of the base ellipses.
- virtual double getRotationAngle () const =0

  Retrieves the rotation angle of the base ellipses.
- virtual void setRotationAngle (double rA)=0

  Sets rotation angle of the arc. An exception is thrown if this segment is immutable.
- virtual bool getIsLargeArc () const =0

  Retrieves isLargeArc.
- virtual void setIsLargeArc (bool isLA)=0

  Sets isLargeArc. See getIsLargeArc() for a description of how isLargeArc determines the arc to be drawn. An exception is thrown if this segment is immutable.
- virtual eSweepDirection getSweepDirection () const =0

  Retrieves the sweep direction.
- virtual void setSweepDirection (eSweepDirection sd)=0

  Sets sweep direction An exception is thrown if this segment is immutable.
- virtual IDOMPathSegmentPtr convertToSimpleSegment (IEDLClassFactory *factory, const FPoint &start↩ Point) const =0

  Create a segment that represents this arc using a simpler segment type.

Public Member Functions inherited from IDOMPathSegment

- virtual bool getIsStroked () const =0

  Retrieves the value for IsStroked. IsStroked specifies whether the stroke for this segment of the path is drawn. Can be true or false.
- virtual void setIsStroked (bool isStroked)=0

  Sets the value of IsStroked. IsStroked specifies whether the stroke for this segment of the path is drawn. Can be true or false. An exception is thrown if this segment is immutable.
- virtual FRect getBounds (FPoint &startPoint) const =0

  Gets the conservative bounding box of the segment given the start point.
- virtual const FPoint & getEndPoint () const =0

  Gets the end point of the segment.
- virtual bool getIsImmutable () const =0

  Determine if the segment is immutable (non-editable).
- virtual void **setImmutable** ()=0

  Force the segment to be flagged immutable.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves the class id of IDOMArcSegment.
- static EDL_API IDOMArcSegmentPtr create (IEDLClassFactory *factory, bool isStroked=false, const FPoint &point=FPoint(), double radiusX=0.0, double radiusY=0.0, double rotationAngle=0.0, bool isLargeArc=false, eSweepDirection sweepDirection=eSDClockwise)

  Simplified creator for an arc segment.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.159.1 Detailed Description

Interface to Arc Segment element.

An ArcSegment describes an elliptical arc. An arc segment is defined by the intersection of two identical ellipses. The ellipses are defined by their x-radius and y-radius, and their angle of rotation relative to the current coordinate system.
The intersection between the ellipses is defined by the start and end points of the arc. The correct line segment of the four specified by the intersection of two ellipses is determined by a combination of sweepDirection and isLargeArc.

Note: The starting point for the arc is defined by the end point of the previous path segment in the path figure (see IDOMPathFigure), or by the starting point defined in the path figure if this is the first path segment in the figure.

Note: It is possible to specify an ellipse that is incapable of intersecting simultaneously with the start and end points of the desired arc. An example of this would be an ellipse with a total height that is less than the vertical distance between the two points. In this case, no intersection is possible and no arc will be generated.

7.159.2 Member Function Documentation

classID()

```
static const CClassID & IDOMArcSegment::classID ( )  [inline], [static]
```

Retrieves the class id of IDOMArcSegment.

Returns

CClassID. The function returns the class id of the element.

convertToSimpleSegment()

```
virtual IDOMPathSegmentPtr IDOMArcSegment::convertToSimpleSegment (
            IEDLClassFactory * factory,
            const FPoint & startPoint ) const  [pure virtual]
```

Create a segment that represents this arc using a simpler segment type.

This member will convert this arc path segment to a simpler segment type.
If the arc segment is degenerate, the result will be a single IDOMPolyLineSegment representing this. Otherwise, the arc segment will be decomposed to cubic beziers in an IDOMPolyBezierSegment.

Parameters

factory	Pointer to the class factory to be used.
startPoint	The start point for the arc.

Returns

IDOMPathSegmentPtr The resulting simple segment.

create()

```
static EDL_API IDOMArcSegmentPtr IDOMArcSegment::create (
            IEDLClassFactory * factory,
            bool isStroked = false,
            const FPoint & point = FPoint(),
            double radiusX = 0.0,
            double radiusY = 0.0,
            double rotationAngle = 0.0,
            bool isLargeArc = false,
            eSweepDirection sweepDirection = eSDClockwise )  [static]
```

Simplified creator for an arc segment.

Parameters

factory	The factory to use.
point	The end point of the arc.
radiusX	The desired x radius of the arc.
radiusY	The desired y radius of the arc.
rotationAngle	The rotation angle.
isLargeArc	Whether or not the large arc should be used.
isStroked	Should the segment be stroked, if used in a stroking path.
sweepDirection	Which direction the arc should sweep.

Returns

IDOMArcSegmentPtr The new segment.

getIsLargeArc()

```
virtual bool IDOMArcSegment::getIsLargeArc ( ) const  [pure virtual]
```

Retrieves isLargeArc.

```
The member isLargeArc determines whether the specified arc is one of the large
arcs (sweep 180 degrees or greater) or one of the small arcs (sweep less than
180 degrees) produced by the intersection of the two ellipses. In conjunction with
the sweep direction, this determines which of the four arcs produced by intersecting
ellipses is required. See also getSweepDirection().
```

Returns

bool The function returns true if the arc is large.

getPoint()

```
virtual const FPoint & IDOMArcSegment::getPoint ( ) const  [pure virtual]
```

Retrieves the end point of the arc.

```
Note: the start point of the arc is specified by the end point of the previous path segment
in the path figure or by the start point specified in the path figure if this is the first
path segment in the figure (see IDOMPathFigure).
```

Returns

FPoint The function returns the end point of the arc.

getRadiusX()

```
virtual double IDOMArcSegment::getRadiusX ( ) const  [pure virtual]
```

Retrieves the x-radius of the arc.

Returns

double The function returns the x-radius of the arc.

getRadiusY()

```
virtual double IDOMArcSegment::getRadiusY ( ) const  [pure virtual]
```

Retrieves the y-radius of the base ellipses.

Returns

double The function returns the y-radius of the base ellipses.

getRotationAngle()

```
virtual double IDOMArcSegment::getRotationAngle ( ) const  [pure virtual]
```

Retrieves the rotation angle of the base ellipses.

```
The rotation angle indicates how the ellipses are rotated relative to the
current coordinate system. Positive values indicate clockwise rotation and
negative values indicate counter clockwise rotation.
```

Returns

double The function returns the angle of rotation in degrees, where positive values indicate clockwise rotation and negative values indicate counter clockwise rotation.

getSweepDirection()

```
virtual eSweepDirection IDOMArcSegment::getSweepDirection ( ) const  [pure virtual]
```

Retrieves the sweep direction.

```
SweepDirection specifies the direction in which the arc is drawn from the start point.
At the start point, all four line segments converge.  Two of these will be short
arcs and two will be long arcs.  One long arc and one short arc will move away from the start
point following a clockwise path; the other two arcs will follow a counter clockwise path.
In conjunction with isLargeArc, the sweep direction determines which of the four line
segments is the specified arc.
```

Returns

eSweepDirection The return value specifies whether the stroke direction is clockwise or counter clockwise from the start point.

setIsLargeArc()

```
virtual void IDOMArcSegment::setIsLargeArc (
            bool isLA )  [pure virtual]
```

Sets isLargeArc. See getIsLargeArc() for a description of how isLargeArc determines the arc to be drawn. An exception is thrown if this segment is immutable.

Parameters

isLA	New value for isLargeArc.

setPoint()

```
virtual void IDOMArcSegment::setPoint (
            const FPoint & pt )  [pure virtual]
```

Sets the end point of the arc. An exception is thrown if this segment is immutable.

Parameters

pt	The new end point of the arc.

setRadiusX()

```
virtual void IDOMArcSegment::setRadiusX (
            double radiusX )  [pure virtual]
```

Sets the x-radius of the base ellipses. An exception is thrown if this segment is immutable.

Parameters

radiusX	The new x-radius of the base ellipses.

setRadiusY()

```
virtual void IDOMArcSegment::setRadiusY (
            double radiusY )  [pure virtual]
```

Sets the y-radius of the base ellipses.

Parameters

radiusY	The new y-radius of the base ellipses.

setRotationAngle()

```
virtual void IDOMArcSegment::setRotationAngle (
            double rA )  [pure virtual]
```

Sets rotation angle of the arc. An exception is thrown if this segment is immutable.

Parameters

rA	New rotation angle. Positive values indicate clockwise rotation and negative values indicate counter clockwise rotation. The rotated ellipses must still be capable of intersecting with the specified start and end points, as before.

setSweepDirection()

```
virtual void IDOMArcSegment::setSweepDirection (
            eSweepDirection sd )  [pure virtual]
```

Sets sweep direction An exception is thrown if this segment is immutable.

Parameters

sd	eSweepDirection The direction in which the arc is drawn from the start point.

The documentation for this class was generated from the following file:

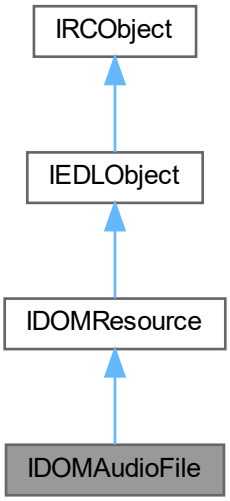
- idompathgeometry.h

7.160 IDOMAudioFile Class Reference

IDOMAudioFile interface.

`#include <idomresources.h>`

Inheritance diagram for IDOMAudioFile:



Classes

- class Data

  Initialization data.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves class id of IDOM.

Additional Inherited Members

Public Member Functions inherited from IDOMResource

- virtual IInputStreamPtr getStream () const =0

  Retrieves the resource stream.
- virtual void setStream (const IInputStreamPtr &stream)=0

  Sets the resource stream for the node.
- virtual uint64 getStreamLength () const =0

  Retrieves the stream length, if it is available.
- virtual const EDLSysString & getUri () const =0

  Retrieves the resource URI.
- virtual void setUri (const EDLSysString &uri)=0

  Sets the resource URI.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.160.1 Detailed Description

IDOMAudioFile interface.

7.160.2 Member Function Documentation

classID()

```
static const CClassID & IDOMAudioFile::classID ( )  [inline], [static]
```

Retrieves class id of IDOM.

Returns

CClassID Class id of the element

The documentation for this class was generated from the following file:

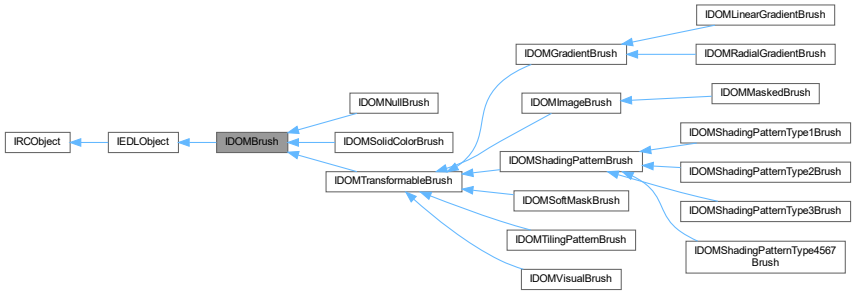
- idomresources.h

7.161 IDOMBrush Class Reference

Interface to the brush element.

`#include <idombrush.h>`

Inheritance diagram for IDOMBrush:



Public Types

- enum eBrushType {
  eSolidColor , eLinearGradient , eRadialGradient , eImage ,
  eMasked , eVisual , eSoftMask , eTilingPattern ,
  eType1ShadingPattern , eType2ShadingPattern , eType3ShadingPattern , eType4567ShadingPattern ,
  eNull }

  Brush type enumeration.

Public Member Functions

- virtual eBrushType getBrushType () const =0

  Retrieves the type of the brush.
- virtual float getOpacity () const =0

  Retrieves the opacity value of the brush element.
- virtual void setOpacity (float opc)=0

  Sets the opacity value of a brush element.
- virtual IDOMBrushPtr getAdjustedForUseInTransformedNode (IEDLClassFactory ∗pFactory, const FMatrix &nodeTransform)

  Get a version of this brush adjusted for use inside a node with the given transform.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams ∗pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory ∗pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.161.1 Detailed Description

Interface to the brush element.

Brushes are used to paint the interior of the geometric shapes defined by a Path and the characters rendered by a Glyphs node. They are also used to define the alpha-transparency mask in the IDOMCanvas::OpacityMask, IDOMPathNode::OpacityMask, and IDOMGlyphs::OpacityMask properties.

All brushes are defined relative to a coordinate space. Most brushes (including image brushes, visual brushes, linear gradient brushes, and radial gradient brushes) may specify a coordinate space transform, in which the transform property is concatenated with the current effective coordinate space to yield an effective coordinate space local to the brush. For image brushes and visual brushes, the viewport is transformed using the local effective render transform. For linear gradient brushes, the start point and end point are transformed. For radial gradient brushes, the center, x-radius, y-radius, and gradient origin are transformed.

If there is an alpha component of the color, it is combined in a muliplicative way with the opacity value.

Note

Overprint information is stored in a property attached to each node that has overprint set - the "Device Params". You can retrieve it with:

```
uint32 devParams = 0;
PValue val;
if (node->getProperty("DeviceParams", val))
{
        devParams = (uint32) val.getInt32();
}
```

It's a set of flags. The most relevant values are:

```
#define OVERPRINT_MODE   1
#define OVERPRINT_FILL   2
#define OVERPRINT_STROKE 4
```

7.161.2 Member Function Documentation

getAdjustedForUseInTransformedNode()

```
virtual IDOMBrushPtr IDOMBrush::getAdjustedForUseInTransformedNode (
            IEDLClassFactory * pFactory,
            const FMatrix & nodeTransform )  [virtual]
```

Get a version of this brush adjusted for use inside a node with the given transform.

That is, it adjusts the transform by post-multiplying the render transform with the inverse of the given transform.

If the brush is not transformable, or if the transform is degenerate (where the matrix of the brush is irrelevant), then the original brush will be returned.

This will also adjust the sub-brush, if this is a masked brush.

Parameters

pFactory	The factory to use.
nodeTransform	The transform of the node in question.

Returns

IDOMBrushPtr The resulting brush.

getBrushType()

```
virtual eBrushType IDOMBrush::getBrushType ( ) const  [pure virtual]
```

Retrieves the type of the brush.

Returns

eBrushType the brush type

getOpacity()

```
virtual float IDOMBrush::getOpacity ( ) const  [pure virtual]
```

Retrieves the opacity value of the brush element.

The opacity value defines the uniform transparency of the canvas. This is a number between 0 (fully transparent) and 1 (fully opaque). Default value 1.0

Returns

float The function returns the brush's opacity value

setOpacity()

```
virtual void IDOMBrush::setOpacity (
            float opc )  [pure virtual]
```

Sets the opacity value of a brush element.

Parameters

opc	The opacity value

The documentation for this class was generated from the following file:

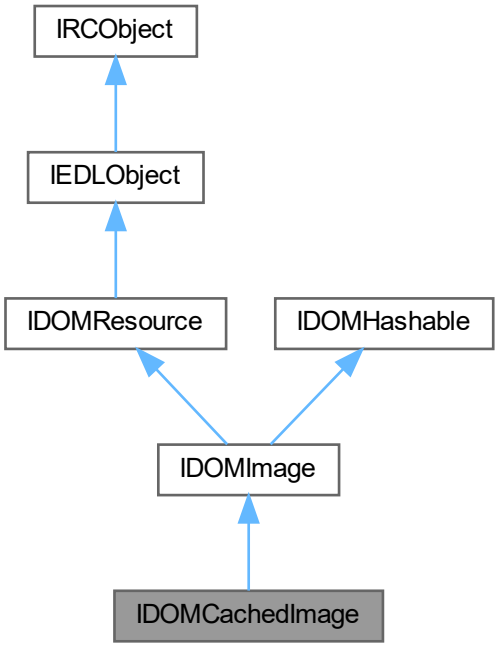
- idombrush.h

7.162 IDOMCachedImage Class Reference

Interface to a class that when overlayed over an image will cache portions of its output. Useful for cases where images are repeatedly and are relatively expensive to process, and where the source image is certain to never change.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMCachedImage:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual IDOMImagePtr getSourceImage () const =0

  Retrieves the source image that is cached by this instance.
- virtual bool getStream (IInputStreamPtr &stream) const

  This image type does not allow direct access to the underlying streams.
- virtual void setStream (const IInputStreamPtr &stream)

  This image type does not allow direct access to the underlying streams.

Public Member Functions inherited from IDOMImage

- virtual IImageDecoderPtr createImageDecoder (IEDLClassFactory *factory, const IDOMImagePropertiesPtr &imageProperties)=0

  Creates a properly initialized image decoder object that reads from an inputstream that is specific to that image format.
- virtual IImageFramePtr getImageFrame (IEDLClassFactory *factory)

  Fetch the image frame; convenience.
- virtual IImageEncoderPtr createImageEncoder (const ISessionPtr &session, const IOutputStreamPtr &imageDest, const IDOMImagePropertiesPtr &imageProperties)=0

  Creates a properly initialized image encoder object that writes to an outputstream that is specific to that image format.
- virtual IDOMImagePropertiesPtr getImageProperties ()=0

  Returns an object that stores the properties for this image object. The properties can then be inspected (or more added) by clients that need to manipulate the image resource.
- virtual eDOMImageType getImageType ()=0

  Retrieves the image type.
- virtual bool getIsRendered ()=0

  Determine if the image is as a result of rendering. This is indicated if the image type is eDITRendered or if the image explicitly notes this is the case (such as for IDOMPDFImage).
- virtual IDOMImagePtr getImageWithSubstitutedColorSpace (IEDLClassFactory *factory, const IDOMColor↩SpacePtr &colorSpace)

  Obtain an image that is the same as this image, but with the colorspace substituted for another.

Public Member Functions inherited from IDOMResource

- virtual IInputStreamPtr getStream () const =0

  Retrieves the resource stream.
- virtual uint64 getStreamLength () const =0

  Retrieves the stream length, if it is available.
- virtual const EDLSysString & getUri () const =0

  Retrieves the resource URI.
- virtual void setUri (const EDLSysString &uri)=0

  Sets the resource URI.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static EDL_API IDOMCachedImagePtr create (IEDLClassFactory *pFactory, const IDOMImagePtr &source↩ Image, bool cacheSamples, bool alwaysCache=false)

  Simplified creator for a IDOMCachedImage.
- static const CClassID & classID ()

  Retrieves class id of IDOMCachedImage.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.162.1 Detailed Description

Interface to a class that when overlayed over an image will cache portions of its output. Useful for cases where images are repeatedly and are relatively expensive to process, and where the source image is certain to never change.

7.162.2 Member Function Documentation

classID()

```
static const CClassID & IDOMCachedImage::classID ( )  [inline], [static]
```

Retrieves class id of IDOMCachedImage.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMCachedImagePtr IDOMCachedImage::create (
            IEDLClassFactory * pFactory,
            const IDOMImagePtr & sourceImage,
            bool cacheSamples,
            bool alwaysCache = false )  [static]
```

Simplified creator for a IDOMCachedImage.

Parameters

pFactory	The EDL Class Factory.
sourceImage	The image to use.
cacheSamples	Set to true if image samples should be cached in addition to metadata. Note that unless forceCache is also set, the samples may not always be actually cached, and may be cached only if the image samples are repeatedly requested. Also note that samples are never cached if the image is too large to be effectively cached.
alwaysCache	Set this (and cacheSamples) to true if samples should always be cached, subject to cache size limits. Consulted only if alwaysCache is true.

Returns

IDOMCachedImage The combined result.

getSourceImage()

```
virtual IDOMImagePtr IDOMCachedImage::getSourceImage ( ) const  [pure virtual]
```

Retrieves the source image that is cached by this instance.

Returns

IDOMImagePtr The source image.

getStream()

```
virtual bool IDOMCachedImage::getStream (
            IInputStreamPtr & stream ) const  [inline], [virtual]
```

This image type does not allow direct access to the underlying streams.

Parameters

stream	A smart pointer to the stream

Returns

bool Always false

setStream()

```
virtual void IDOMCachedImage::setStream (
            const IInputStreamPtr & stream )  [inline], [virtual]
```

This image type does not allow direct access to the underlying streams.

Parameters

stream	A smart pointer to the stream

Implements IDOMResource.

The documentation for this class was generated from the following file:

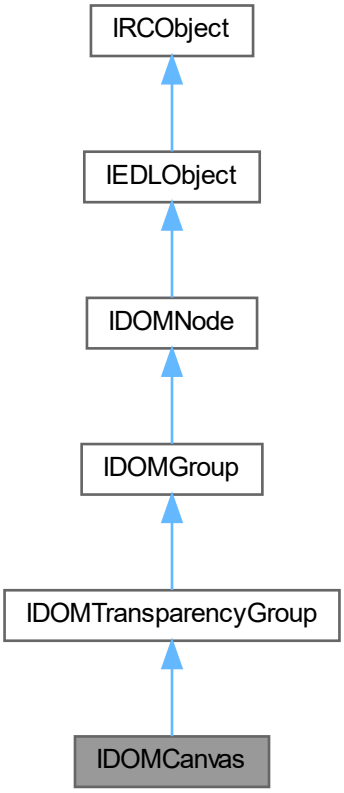
- idomimageresource.h

7.163 IDOMCanvas Class Reference

A canvas is a special form of an isolated, non-knockout, normal blended transparency group.

`#include <idomcanvas.h>`

Inheritance diagram for IDOMCanvas:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual EDLString getLanguage () const =0

  Retrieves the default language of the <Canvas> element and any of its children.
- virtual void setLanguage (const EDLString &lang)=0

  Sets the default language of the <Canvas> element and any of its children.
- virtual EDLString getAutomationPropertiesName () const =0

  Retrieves the automation properties name of the <Canvas> element.
- virtual void setAutomationPropertiesName (const EDLString &propname)=0

  Sets the automation properties name of the <Canvas> element.
- virtual EDLString getAutomationPropertiesHelpText () const =0

  Retrieves the automation properties help text of the <Canvas> element.
- virtual void setAutomationPropertiesHelpText (const EDLString &helptext)=0

  Sets automation properties help text of the <Canvas> element.
- virtual eEdgeMode getEdgeMode () const =0

  Retrieves render options edge mode of the <Canvas> element.
- virtual void setEdgeMode (eEdgeMode em)=0

  Sets render the options edge mode of the <Canvas> element.
- virtual IDOMTargetPtr getNavigateLink () const =0

  Retrieves the target of a hyperlink.
- virtual void setNavigateLink (const IDOMTargetPtr &target)=0

  Sets the target of a hyperlink.
- virtual IDOMResourceDictionaryPtr getResourceDictionary () const =0

  Retrieves a smart pointer to the resource dictionary.
- virtual void setResourceDictionary (const IDOMResourceDictionaryPtr &ptrResourceDictionary)=0

  Sets the resource dictionary.

Public Member Functions inherited from IDOMTransparencyGroup

- virtual float getOpacity () const =0

  Get the group alpha/opacity.
- virtual void setOpacity (float opacity)=0

  Set the group opacity.
- virtual eBlendMode getBlendMode () const =0

  Get the blend mode to be used for compositing this group with the backdrop.
- virtual void setBlendMode (eBlendMode blendMode)=0

  Set the blend mode to be used for compositing this group with the backdrop.
- virtual IDOMColorSpacePtr getColorSpace () const =0

  Get the group colorspace.
- virtual void setColorSpace (const IDOMColorSpacePtr &colorSpace)=0

  Set the group colorspace.
- virtual bool getIsIsolated () const =0

  Is the group an isolated group? See section 7.5.5 of the PDF 1.7 spec for details.
- virtual void setIsIsolated (bool isolated)=0

  Set whether the group is isolated. See section 7.5.5 of the PDF 1.7 spec for details.
- virtual bool getIsKnockout () const =0

  Is the group a knockout group? See section 7.5.5 of the PDF 1.7 spec for details.
- virtual void setIsKnockout (bool knockout)=0

  Set whether the group is a knockout group. See section 7.5.5 of the PDF 1.7 spec for details.
- virtual IDOMBrushPtr getOpacityMask () const =0

  Retrieves smart pointer to opacity mask.
- virtual void setOpacityMask (const IDOMBrushPtr &ptrOpacityMask)=0

  Sets opacity mask.

Public Member Functions inherited from IDOMGroup

- virtual const FMatrix & getRenderTransform () const =0

  Retrieves render transform matrix of the Group and its children.
- virtual void setRenderTransform (const FMatrix &matrix)=0

  Sets render transform matrix of the Group and its children.
- virtual IDOMPathGeometryPtr getClip () const =0

  Retrieves smart pointer to the clip.
- virtual void setClip (const IDOMPathGeometryPtr &ptrClip)=0

  Sets clip.
- virtual JawsMako::IOptionalContentDetailsPtr getOptionalContentDetails () const =0

  Get the JawsMako Optional Content details, or NULL if the group is not subject to optional content.
- virtual void setOptionalContentDetails (const JawsMako::IOptionalContentDetailsPtr &details)=0

  Set the JawsMako Optional Content details for the group, or NULL to remove. Note that this is only allowed for bare IDOMGroup objects and not for subclasses. Any attempt to set optional content details on any object that is not an IDOMGroup (getNodeType() is eDOMGroupNode) will result in an exception. Note that an IDOMGroup may have optional content details or marked content details, but not both.
- virtual JawsMako::IMarkedContentDetailsPtr getMarkedContentDetails () const =0

  Get the JawsMako Marked Content details for this group, or NULL if the group is not marked.
- virtual void setMarkedContentDetails (const JawsMako::IMarkedContentDetailsPtr &details)=0

  Set the JawsMako Marked Content details for this group, or NULL to remove. Note that this is only allowed for bare IDOMGroup objects and not for subclasses. Any attempt to set marked content details on any object that is not an IDOMGroup (getNodeType() is eDOMGroupNode) will result in an exception. Note that an IDOMGroup may have optional content details or marked content details, but not both.

Public Member Functions inherited from IDOMNode

- virtual ~**IDOMNode** ()

  virtual destructor
- virtual DOMid getDOMid () const =0

  Retrieves the node ID.
- virtual void setDOMid (DOMid id)=0

  Sets the node ID.
- virtual eDOMNodeType getNodeType () const =0

  Retrieves the DOM node type.
- virtual bool getProperty (const EDLSysString &propertyName, PValue &propertyValue) const =0

  Retrieves the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.
- virtual void setProperty (const EDLSysString &propertyName, const PValue &propertyValue)=0

  Sets the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.
- virtual void removeProperty (const EDLSysString &propertyName)=0

  Removes property.
- virtual IEDLSysStringCollectionEnumPtr getPropertyCollectionEnum ()=0

  Retrieves a navigable list of the property names stored on this node.
- virtual bool hasChildNodes () const =0

  Function that indicates whether this node is a parent to other nodes.
- virtual IDOMNodePtr getParentNode () const =0

  Gets the parent node of this node.
- virtual IDOMNodePtr getFirstChild () const =0

Gets the first child node of this node.

- virtual IDOMNodePtr getLastChild () const =0

  Gets the last child node of this node.

- virtual IDOMNodePtr getNextChild (const IDOMNodePtr &child) const =0

  Gets the child node which follows the node passed in.

- virtual IDOMNodePtr getPreviousChild (const IDOMNodePtr &child) const =0

  Gets the child node which precedes the node passed in.

- virtual IDOMNodePtr getPreviousSibling () const =0

  Retrieves the node's previous sibling node.

- virtual IDOMNodePtr getNextSibling () const =0

  Retrieves node's next sibling node.

- virtual void appendChild (const IDOMNodePtr &child)=0

  Appends a node to the end of the node's child list.

- virtual void insertChild (const IDOMNodePtr &ptrPreviousSibling, const IDOMNodePtr &child, bool bCheck↩ Complete=true)=0

  Insert a child node after ptrPreviousSibling.

- virtual IDOMNodePtr extractChild (const IDOMNodePtr &child)=0

  Extracts (that is, finds and removes) a child node from the node children. After extraction the child node is no longer a part of the DOM. If no node is specified, the first available node will be extracted from the node's children.

- virtual void replaceChild (const IDOMNodePtr &oldChild, const IDOMNodePtr &newChild)=0

  Replaces the child node with another.

- virtual bool isComplete () const =0

  Signals the completeness of the node.
  A complete node is one that has no more children to be added to it.

- virtual void **setComplete** ()=0

  Sets the node's completeness status to "true".

- virtual IDOMNodeFlags * getFlags ()=0

  Retrieves the node's flags property.

- virtual void setParentNode (const IDOMNodePtr &ptrParent)=0

  Sets the parent node.

- virtual void setPreviousSibling (const IDOMNodePtr &ptrPreviousSibling)=0

  Sets the previous sibling node.

- virtual void setNextSibling (const IDOMNodePtr &ptrNextSibling)=0

  Sets the next sibling node.

- virtual bool isAncestor (const IDOMNodePtr &ptrCandidate)=0

  Function tests whether a candidate node is a descendant of the node.

- virtual FRect getBounds (bool applyTransform=true, bool applyClip=true)

  Find the conservative bounding box of the marking content of the node.

- virtual bool copyNodeData (IDOMNode *pSourceNode)=0

  Copy the properties collection, the flags and the DOM ID from the given source node to this one.

- virtual IDOMNodePtr cloneNode (IEDLClassFactory *pFactory) const =0

  Simpified node cloning. An exception of type IEDLError will be thrown on failure.

- virtual IDOMNodePtr cloneTree (IEDLClassFactory *pFactory) const =0

  Clone the tree of nodes beginning at this node. An exception of type IEDLError will be thrown on failure.

- virtual void cloneTreeAndAppend (IEDLClassFactory *pFactory, const IDOMNodePtr &dest) const =0

  Clone the tree of nodes beginning at this node, and append the result to the destination tree.

- virtual void **completeTree** ()=0

  Mark the entire tree from this point as complete. You should not ordinarily need to call this function.

- virtual void **removeCompleteFlagFromTree** ()=0

  Mark the entire tree from this point as complete.

- virtual void findChildrenOfType (eDOMNodeType type, CDOMNodeVect &nodes, bool searchForms=false)=0

Find all children of this node with the given type, appending to the given vector. Does not descend into brushes.

- virtual void walkTree (WalkTreeFunc func, void *priv, bool descendIntoBrushes=false, bool descendInto↩ Forms=false)=0

  Walk through the DOM calling a given function on each node. The function is allowed to:
- virtual void notifyOnDestruct (NodeDeleteFunc func, void *priv)=0

  Register interest in being told when this node is about to be destroyed.
- virtual void unregisterNotify (NodeDeleteFunc func, void *priv)=0

  Unregister interest in being told when this node is about to be destroyed.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves the class id of IDOMCanvas.

Static Public Member Functions inherited from IDOMTransparencyGroup

- static const CClassID & classID ()

  Retrieves class id of IDOMTransparencyGroup.
- static EDL_API IDOMTransparencyGroupPtr create (IEDLClassFactory *pFactory, const FMatrix &transform=FMatrix(), const IDOMPathGeometryPtr &clip=IDOMPathGeometryPtr(), float opacity=1.0f, const IDOMBrushPtr &opacityMask=IDOMBrushPtr(), eBlendMode blendMode=eBlendModeNormal, const IDOMColorSpacePtr &colorSpace=IDOMColorSpacePtr(), bool isolated=false, bool knockout=false)

  Simplified creation function for IDOMGroup. Throws an IEDLError exception on failure.

Static Public Member Functions inherited from IDOMGroup

- static EDL_API IDOMGroupPtr create (IEDLClassFactory *pFactory, const FMatrix &transform=FMatrix(), const IDOMPathGeometryPtr &clip=IDOMPathGeometryPtr())

  Simplified creation function for IDOMGroup. Throws an IEDLError exception on failure.
- static const CClassID & classID ()

  Retrieves class id of IDOMGroup.

Static Public Member Functions inherited from IDOMNode

- static EDL_API FMatrix effectiveTransformationOfNode (const IDOMNodePtr &node)

  Attempt to find the effective transformation matrix external to the specified node relative to either a containing page or ultimate parent.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.163.1 Detailed Description

A canvas is a special form of an isolated, non-knockout, normal blended transparency group.

A Canvas groups other elements of a page together. For example, Glyphs and Paths can be grouped in a Canvas in order to be identified as a unit or to apply a composed property value to each child and its ancestor.

Some properties of the Canvas element, including the coordinate space of the canvas, are composable and affect the rendering of child elements.

The RenderOptionsEdgeMode property can be set in the Canvas node to instruct anti-aliasing consumers to render the contents of the Canvas node and all child and descendant nodes without performing anti-aliasing.

7.163.2 Member Function Documentation

classID()

```
static const CClassID & IDOMCanvas::classID ( )  [inline], [static]
```

Retrieves the class id of IDOMCanvas.

Returns

CClassID Class id of the element

getAutomationPropertiesHelpText()

```
virtual EDLString IDOMCanvas::getAutomationPropertiesHelpText ( ) const  [pure virtual]
```

Retrieves the automation properties help text of the <Canvas> element.

The automation properties help text of a canvas is a more detailed description of the canvas content for accessibility purposes. This is particularly useful if the canvas is filled with a set of vector graphics and text elements intended to comprise a single vector graphic. For example, "This is a picture of the Earth."

Returns

EDLString. Returns the automation properties help text.

getAutomationPropertiesName()

```
virtual EDLString IDOMCanvas::getAutomationPropertiesName ( ) const  [pure virtual]
```

Retrieves the automation properties name of the <Canvas> element.

The automation properties name is a brief description of the <Canvas> content for accessibility purposes, particularly if the canvas is filled with a set of vector graphics and text elements intended to comprise a single vector graphic.

Returns

EDLString. Returns the automation properties name.

getEdgeMode()

```
virtual eEdgeMode IDOMCanvas::getEdgeMode ( ) const  [pure virtual]
```

Retrieves render options edge mode of the <Canvas> element.

Render options edge mode controls how edges of paths within the canvas are rendered. The only valid value for render options edge mode is Aliased. Omitting this attribute causes the edges to be rendered in the consumer's default manner

Returns

eEdgeMode. The function returns the edge mode of the canvas.

getLanguage()

```
virtual EDLString IDOMCanvas::getLanguage ( ) const  [pure virtual]
```

Retrieves the default language of the <Canvas> element and any of its children.

English is defined as en_GB and American English as en_US. There is no default setting. If the language is not known it is set to und (undetermined). For further information see http://www.w3.org/International/articles/language-tags/.

The language is specified according to RFC 3066.

Returns

EDLString. Returns the language setting of the <Canvas> element

getNavigateLink()

```
virtual IDOMTargetPtr IDOMCanvas::getNavigateLink ( ) const  [pure virtual]
```

Retrieves the target of a hyperlink.

Returns

IDOMTargetPtr. Returns the target of the hyperlink

getResourceDictionary()

```
virtual IDOMResourceDictionaryPtr IDOMCanvas::getResourceDictionary ( ) const  [pure virtual]
```

Retrieves a smart pointer to the resource dictionary.

The resource dictionary contains the IDs of resources stored at this level of the document.

Returns

IIDOMResourceDictionaryPtr. Returns a smart pointer to the resource dictionary.

setAutomationPropertiesHelpText()

```
virtual void IDOMCanvas::setAutomationPropertiesHelpText (
            const EDLString & helptext )  [pure virtual]
```

Sets automation properties help text of the <Canvas> element.

The automation properties help text of a canvas is a more detailed description of the canvas content for accessibility purposes. This is particularly useful if the canvas is filled with a set of vector graphics and text elements intended to comprise a single vector graphic. For example, "This is a picture of the Earth."

Parameters

helptext	The new automation properties help text.

setAutomationPropertiesName()

```
virtual void IDOMCanvas::setAutomationPropertiesName (
            const EDLString & propname )  [pure virtual]
```

Sets the automation properties name of the <Canvas> element.

The automation properties name is a brief description of the <Canvas> content for accessibility purposes, particularly if the canvas is filled with a set of vector graphics and text elements intended to comprise a single vector graphic.

Parameters

propname	The new automation properties name.

setEdgeMode()

```
virtual void IDOMCanvas::setEdgeMode (
            eEdgeMode em )  [pure virtual]
```

Sets render the options edge mode of the <Canvas> element.

The EdgeMode property controls how the edges of paths within the canvas are rendered. The only valid value is Aliased. Omitting this attribute causes the edges to be rendered in the consumers default manner. The EdgeMode property can be set in a Canvas to instruct anti-aliasing consumers to render the contents of the Canvas and all child and descendant nodes without performing antialiasing.

Parameters

em	The new edge mode for the canvas.

setLanguage()

```
virtual void IDOMCanvas::setLanguage (
            const EDLString & lang )  [pure virtual]
```

Sets the default language of the <Canvas> element and any of its children.

Parameters

lang	The new language setting for the canvas.

English is defined as en_GB and American English as en_US. There is no default setting. If the language is not known it is set to und (undetermined). For further information see http://www.w3.org/↩International/articles/language-tags/.

The language is specified according to RFC 3066.

setNavigateLink()

```
virtual void IDOMCanvas::setNavigateLink (
            const IDOMTargetPtr & target )  [pure virtual]
```

Sets the target of a hyperlink.

Parameters

target	The new target for the hyperlink.

setResourceDictionary()

```
virtual void IDOMCanvas::setResourceDictionary (
            const IDOMResourceDictionaryPtr & ptrResourceDictionary )  [pure virtual]
```

Sets the resource dictionary.

The resource dictionary contains the IDs of resources stored at this level of the document.

Parameters

ptrResourceDictionary	Smart pointer to the resource dictionary

The documentation for this class was generated from the following file:

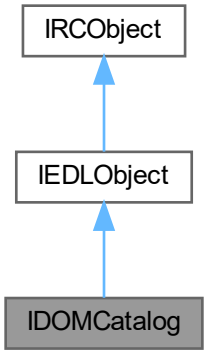
- idomcanvas.h

7.164 IDOMCatalog Class Reference

IDOMCatalog interface The IDOMCatalog serves as a catalog for addressable DOM nodes, where a DOM node ID is used as the address of the node.

`#include <idomcatalog.h>`

Inheritance diagram for IDOMCatalog:



Public Member Functions

- virtual DOMid createNewDOMid (const EDLSysString &uri)=0

  Creates a new DOM id from a URI string.
- virtual DOMid createNewDOMid (uint32 id1, uint32 id2=0)=0

  Creates a new DOM id based on an integer identifier.
- virtual bool registerObject (DOMid id, const IEDLObjectPtr &ptrObject)=0

  Registers the IEDLObject specified by ptrObject in the catalog, under the ID specified by id.
- virtual bool registerNumbers (DOMid id, uint32 id1, uint32 id2)=0

  Registers the number pait in the catalog, under the ID specified by id.
- virtual bool unregisterObject (DOMid id)=0

  Unregisters the IEDLObject currently registered under the specified ID in the catalog.
- virtual bool getObject (DOMid id, IEDLObjectPtr &ptrObject)=0

  Retrieves an IEDLObject by using its DOM id.
- virtual bool getURI (DOMid id, EDLSysString &uri)=0

  Retrieves the URI of a registered object by using its DOM ID.
- virtual DOMid getIdByURI (const EDLSysString &uri, bool createIfNotExist=false)=0

  Retrieve the DOM ID of a registered resource by using its URI, optionally creating it.
- virtual DOMid getIdByNumbers (uint32 id1, uint32 id2=0, bool createIfNotExist=false)=0

  Retrieve the DOM ID of a registered resource by using the integer identifier(s) used to create the DOM ID, optionally creating it.
- virtual DOMid getIdByIndex (uint32 index) const =0

  Retrieve the DOM ID of a registered resource by its index in the internal vector.
- virtual uint32 getCount () const =0

  Retrieve count of registered resource objects.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves the class id of IDOMCatalog.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.164.1 Detailed Description

IDOMCatalog interface The IDOMCatalog serves as a catalog for addressable DOM nodes, where a DOM node ID is used as the address of the node.

7.164.2 Member Function Documentation

classID()

```
static const CClassID & IDOMCatalog::classID ( )  [inline], [static]
```

Retrieves the class id of IDOMCatalog.

Returns

CClassID Class id of the catalog.

createNewDOMid() [1/2]

```
virtual DOMid IDOMCatalog::createNewDOMid (
            const EDLSysString & uri )  [pure virtual]
```

Creates a new DOM id from a URI string.

Parameters

uri	A string uniquely addressing a resource.

Returns

DOMid. The newly created DOM id. The newly allocated ID will be unique across the the current process.

createNewDOMid() [2/2]

```
virtual DOMid IDOMCatalog::createNewDOMid (
            uint32 id1,
            uint32 id2 = 0 )  [pure virtual]
```

Creates a new DOM id based on an integer identifier.

Parameters

id1	A numeric identifier.
id2	A second numeric identifier (optional. Presently not used, and set to zero.)

Returns

DOMid. The newly created DOM id. The newly allocated ID will be unique across the the current process.

getCount()

```
virtual uint32 IDOMCatalog::getCount ( ) const  [pure virtual]
```

Retrieve count of registered resource objects.

Returns

uint32 Returns the count of catalog entries.

getIdByIndex()

```
virtual DOMid IDOMCatalog::getIdByIndex (
            uint32 index ) const  [pure virtual]
```

Retrieve the DOM ID of a registered resource by its index in the internal vector.

Parameters

index	The registered resource index

Returns

DOMid Returns the DOM ID of the resource.

getIdByNumbers()

```
virtual DOMid IDOMCatalog::getIdByNumbers (
            uint32 id1,
            uint32 id2 = 0,
            bool createIfNotExist = false )  [pure virtual]
```

Retrieve the DOM ID of a registered resource by using the integer identifier(s) used to create the DOM ID, optionally creating it.

Parameters

id1	A numeric identifier.
id2	A second numeric identifier (optional. Presently not used, and set to zero.)
createIfNotExist	If set to true, create a new DOM ID if it doesn't already exist

Returns

DOMid The DOM id of the identified resource. If a new id has been created it will be unique across the process.

getIdByURI()

```
virtual DOMid IDOMCatalog::getIdByURI (
            const EDLSysString & uri,
            bool createIfNotExist = false )  [pure virtual]
```

Retrieve the DOM ID of a registered resource by using its URI, optionally creating it.

Parameters

uri	URI of the resource whose DOM ID is required.
createIfNotExist	If set to true, create a new DOM ID if it doesn't already exist

Returns

DOMid The DOM ID of the resource.

getObject()

```
virtual bool IDOMCatalog::getObject (
            DOMid id,
            IEDLObjectPtr & ptrObject )  [pure virtual]
```

Retrieves an IEDLObject by using its DOM id.

Parameters

id	The DOM ID of the IEDLObject to retrieve.
ptrObject	A Smart pointer to receive a reference to the retrieved IEDLObject.

Returns

bool True on success, false if the call fails.

getURI()

```
virtual bool IDOMCatalog::getURI (
            DOMid id,
            EDLSysString & uri )  [pure virtual]
```

Retrieves the URI of a registered object by using its DOM ID.

Parameters

id	DOM ID of the resource
uri	Pointer to receive the URI for the resource

Returns

bool True on success, false if the call fails.

registerNumbers()

```
virtual bool IDOMCatalog::registerNumbers (
            DOMid id,
            uint32 id1,
            uint32 id2 )  [pure virtual]
```

Registers the number pait in the catalog, under the ID specified by id.

Parameters

id	The ID to register the object under.
id1	A numeric identifier.
id2	A second numeric identifier.

Returns

bool True on success, false if the call fails.

registerObject()

```
virtual bool IDOMCatalog::registerObject (
```

```
            DOMid id,
            const IEDLObjectPtr & ptrObject )  [pure virtual]
```

Registers the IEDLObject specified by ptrObject in the catalog, under the ID specified by id.

Parameters

id	The ID to register the object under.
ptrObject	A smart pointer to the IEDLObject to register.

Returns

bool True on success, false if the call fails.

unregisterObject()

```
virtual bool IDOMCatalog::unregisterObject (
            DOMid id )  [pure virtual]
```

Unregisters the IEDLObject currently registered under the specified ID in the catalog.

Parameters

id	The DOM id of the object to unregister.

Returns

bool True on success, false if the call fails.

The documentation for this class was generated from the following file:

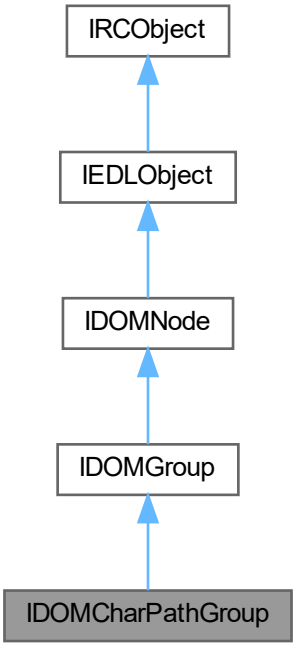
- idomcatalog.h

7.165 IDOMCharPathGroup Class Reference

IDOMCharPathGroup interface Interface to DOM node representing Group Elements that consist of stroked text.

```
#include <idomcharpathgroup.h>
```

Inheritance diagram for IDOMCharPathGroup:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual eCharPathType getCharPathType () const =0

  Retrieves the type of charpath rendering needed which can be Stroke or Clip.
- virtual void setCharPathType (eCharPathType charPathType)=0

  Sets the type of charpath rendering needed which can be Stroke or Clip.
- virtual IDOMPathNodePtr getStrokePath () const =0

  Gets the equivalent path that the charpath makes.
- virtual void setStrokePath (const IDOMPathNodePtr &strokePath)=0

  Sets the equivalent path that the charpath makes.
- virtual IDOMGroupPtr getClippedGroup () const =0

  Gets the objects that need to be drawn that must be clipped by the charpath. Only used when the charpath type is Clip.
- virtual void setClippedGroup (const IDOMGroupPtr &clippedGroup)=0

  Sets the objects that need to be drawn that must be clipped by the charpath Only used when the charpath type is Clip.

Public Member Functions inherited from IDOMGroup

- virtual const FMatrix & getRenderTransform () const =0

  Retrieves render transform matrix of the Group and its children.
- virtual void setRenderTransform (const FMatrix &matrix)=0

  Sets render transform matrix of the Group and its children.
- virtual IDOMPathGeometryPtr getClip () const =0

  Retrieves smart pointer to the clip.
- virtual void setClip (const IDOMPathGeometryPtr &ptrClip)=0

  Sets clip.
- virtual JawsMako::IOptionalContentDetailsPtr getOptionalContentDetails () const =0

  Get the JawsMako Optional Content details, or NULL if the group is not subject to optional content.
- virtual void setOptionalContentDetails (const JawsMako::IOptionalContentDetailsPtr &details)=0

  Set the JawsMako Optional Content details for the group, or NULL to remove. Note that this is only allowed for bare IDOMGroup objects and not for subclasses. Any attempt to set optional content details on any object that is not an IDOMGroup (getNodeType() is eDOMGroupNode) will result in an exception. Note that an IDOMGroup may have optional content details or marked content details, but not both.
- virtual JawsMako::IMarkedContentDetailsPtr getMarkedContentDetails () const =0

  Get the JawsMako Marked Content details for this group, or NULL if the group is not marked.
- virtual void setMarkedContentDetails (const JawsMako::IMarkedContentDetailsPtr &details)=0

  Set the JawsMako Marked Content details for this group, or NULL to remove. Note that this is only allowed for bare IDOMGroup objects and not for subclasses. Any attempt to set marked content details on any object that is not an IDOMGroup (getNodeType() is eDOMGroupNode) will result in an exception. Note that an IDOMGroup may have optional content details or marked content details, but not both.

Public Member Functions inherited from IDOMNode

- virtual ~**IDOMNode** ()

  virtual destructor
- virtual DOMid getDOMid () const =0

  Retrieves the node ID.
- virtual void setDOMid (DOMid id)=0

  Sets the node ID.
- virtual eDOMNodeType getNodeType () const =0

  Retrieves the DOM node type.
- virtual bool getProperty (const EDLSysString &propertyName, PValue &propertyValue) const =0

  Retrieves the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.
- virtual void setProperty (const EDLSysString &propertyName, const PValue &propertyValue)=0

  Sets the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.
- virtual void removeProperty (const EDLSysString &propertyName)=0

  Removes property.
- virtual IEDLSysStringCollectionEnumPtr getPropertyCollectionEnum ()=0

  Retrieves a navigable list of the property names stored on this node.
- virtual bool hasChildNodes () const =0

  Function that indicates whether this node is a parent to other nodes.
- virtual IDOMNodePtr getParentNode () const =0

  Gets the parent node of this node.
- virtual IDOMNodePtr getFirstChild () const =0

Gets the first child node of this node.

- virtual IDOMNodePtr getLastChild () const =0

  Gets the last child node of this node.

- virtual IDOMNodePtr getNextChild (const IDOMNodePtr &child) const =0

  Gets the child node which follows the node passed in.

- virtual IDOMNodePtr getPreviousChild (const IDOMNodePtr &child) const =0

  Gets the child node which precedes the node passed in.

- virtual IDOMNodePtr getPreviousSibling () const =0

  Retrieves the node's previous sibling node.

- virtual IDOMNodePtr getNextSibling () const =0

  Retrieves node's next sibling node.

- virtual void appendChild (const IDOMNodePtr &child)=0

  Appends a node to the end of the node's child list.

- virtual void insertChild (const IDOMNodePtr &ptrPreviousSibling, const IDOMNodePtr &child, bool bCheck↩ Complete=true)=0

  Insert a child node after ptrPreviousSibling.

- virtual IDOMNodePtr extractChild (const IDOMNodePtr &child)=0

  Extracts (that is, finds and removes) a child node from the node children. After extraction the child node is no longer a part of the DOM. If no node is specified, the first available node will be extracted from the node's children.

- virtual void replaceChild (const IDOMNodePtr &oldChild, const IDOMNodePtr &newChild)=0

  Replaces the child node with another.

- virtual bool isComplete () const =0

  Signals the completeness of the node.
  A complete node is one that has no more children to be added to it.

- virtual void **setComplete** ()=0

  Sets the node's completeness status to "true".

- virtual IDOMNodeFlags * getFlags ()=0

  Retrieves the node's flags property.

- virtual void setParentNode (const IDOMNodePtr &ptrParent)=0

  Sets the parent node.

- virtual void setPreviousSibling (const IDOMNodePtr &ptrPreviousSibling)=0

  Sets the previous sibling node.

- virtual void setNextSibling (const IDOMNodePtr &ptrNextSibling)=0

  Sets the next sibling node.

- virtual bool isAncestor (const IDOMNodePtr &ptrCandidate)=0

  Function tests whether a candidate node is a descendant of the node.

- virtual FRect getBounds (bool applyTransform=true, bool applyClip=true)

  Find the conservative bounding box of the marking content of the node.

- virtual bool copyNodeData (IDOMNode *pSourceNode)=0

  Copy the properties collection, the flags and the DOM ID from the given source node to this one.

- virtual IDOMNodePtr cloneNode (IEDLClassFactory *pFactory) const =0

  Simpified node cloning. An exception of type IEDLError will be thrown on failure.

- virtual IDOMNodePtr cloneTree (IEDLClassFactory *pFactory) const =0

  Clone the tree of nodes beginning at this node. An exception of type IEDLError will be thrown on failure.

- virtual void cloneTreeAndAppend (IEDLClassFactory *pFactory, const IDOMNodePtr &dest) const =0

  Clone the tree of nodes beginning at this node, and append the result to the destination tree.

- virtual void **completeTree** ()=0

  Mark the entire tree from this point as complete. You should not ordinarily need to call this function.

- virtual void **removeCompleteFlagFromTree** ()=0

  Mark the entire tree from this point as complete.

- virtual void findChildrenOfType (eDOMNodeType type, CDOMNodeVect &nodes, bool searchForms=false)=0

Find all children of this node with the given type, appending to the given vector. Does not descend into brushes.

- virtual void walkTree (WalkTreeFunc func, void *priv, bool descendIntoBrushes=false, bool descendInto↩ Forms=false)=0

  Walk through the DOM calling a given function on each node. The function is allowed to:
- virtual void notifyOnDestruct (NodeDeleteFunc func, void *priv)=0

  Register interest in being told when this node is about to be destroyed.
- virtual void unregisterNotify (NodeDeleteFunc func, void *priv)=0

  Unregister interest in being told when this node is about to be destroyed.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves class id of IDOMGroup.

Static Public Member Functions inherited from IDOMGroup

- static EDL_API IDOMGroupPtr create (IEDLClassFactory *pFactory, const FMatrix &transform=FMatrix(), const IDOMPathGeometryPtr &clip=IDOMPathGeometryPtr())

  Simplified creation function for IDOMGroup. Throws an IEDLError exception on failure.
- static const CClassID & classID ()

  Retrieves class id of IDOMGroup.

Static Public Member Functions inherited from IDOMNode

- static EDL_API FMatrix effectiveTransformationOfNode (const IDOMNodePtr &node)

  Attempt to find the effective transformation matrix external to the specified node relative to either a containing page or ultimate parent.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()
  Virtual destructor.

7.165.1 Detailed Description

IDOMCharPathGroup interface Interface to DOM node representing Group Elements that consist of stroked text.

7.165.2 Member Function Documentation

classID()

```
static const CClassID & IDOMCharPathGroup::classID ( )  [inline], [static]
```

Retrieves class id of IDOMGroup.

Returns

CClassID class id of the element

getCharPathType()

```
virtual eCharPathType IDOMCharPathGroup::getCharPathType ( ) const  [pure virtual]
```

Retrieves the type of charpath rendering needed which can be Stroke or Clip.

Returns

eCharPathType Returns the type of charpath rendering

getClippedGroup()

```
virtual IDOMGroupPtr IDOMCharPathGroup::getClippedGroup ( ) const  [pure virtual]
```

Gets the objects that need to be drawn that must be clipped by the charpath. Only used when the charpath type is Clip.

Returns

IDOMGroupPtr Returns a smart pointer to group of objects that should be clipped or NULL if there is no such group.

getStrokePath()

```
virtual IDOMPathNodePtr IDOMCharPathGroup::getStrokePath ( ) const  [pure virtual]
```

Gets the equivalent path that the charpath makes.

Returns

IDOMPathNodePtr Returns a smart pointer to stroke path of the charpath

setCharPathType()

```
virtual void IDOMCharPathGroup::setCharPathType (
            eCharPathType charPathType )  [pure virtual]
```

Sets the type of charpath rendering needed which can be Stroke or Clip.

Parameters

charPathType	The type of charpath

setClippedGroup()

```
virtual void IDOMCharPathGroup::setClippedGroup (
            const IDOMGroupPtr & clippedGroup )  [pure virtual]
```

Sets the objects that need to be drawn that must be clipped by the charpath Only used when the charpath type is Clip.

Parameters

clippedGroup	Smart pointer to group of objects that should be clipped or NULL if there is no such group.

setStrokePath()

```
virtual void IDOMCharPathGroup::setStrokePath (
            const IDOMPathNodePtr & strokePath )  [pure virtual]
```

Sets the equivalent path that the charpath makes.

Parameters

strokePath	Smart pointer to stroke path of the charpath

The documentation for this class was generated from the following file:

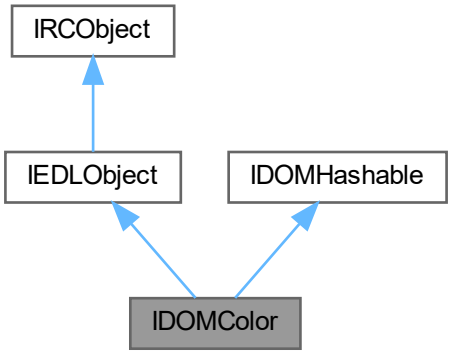
- idomcharpathgroup.h

7.166 IDOMColor Class Reference

Holds a single color value. The color values themselves are held as floating point values for all color spaces. For some spaces (such as indexed color spaces) the values will be integral, but still stored as floats.

```
#include <idomcolor.h>
```

Inheritance diagram for IDOMColor:



Public Member Functions

- virtual float getAlpha () const =0

  Retrieves the alpha channel value. The alpha channel value specifies the transparency of the color.
- virtual void setAlpha (float a)=0

  Sets the alpha channel value.
- virtual IDOMColorSpacePtr getColorSpace ()=0

  Retrieves the color space.
- virtual void setColorSpace (const IDOMColorSpacePtr &colorSpace, bool setDefaultColor=true)=0

  Set the color space.
- virtual void setColorSpace (const IDOMColorSpacePtr &colorSpace, eRenderingIntent intent, eBlackPointCompensation bpc, IEDLClassFactory ∗pFactory)=0

  Set the color space, converting color values from the previous color space to the new.
- virtual float getComponentValue (uint32 component)=0

  Retrieves the value of a component.
- virtual void setComponentValue (uint32 component, float value)=0

  Sets a component value. The value will be clipped to the range of the color space, if known.
- virtual bool testGamut (IEDLClassFactory ∗pFactory, const IDOMColorSpacePtr &colorSpace, eRenderingIntent intent, eBlackPointCompensation bpc)=0

  Tests the color is within the color gamut for the given color space.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams ∗pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory ∗pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static IDOMColorPtr createSolidGray (IEDLClassFactory *pFactory, float gray=0.0f)

  Simplified color creation for DeviceGray colours. Default parameters will yield an opaque DeviceGray black.
- static IDOMColorPtr createSolidRgb (IEDLClassFactory *pFactory, float r=0.0f, float g=0.0f, float b=0.0f)

  Simplified color creation for DeviceRGB colours. Default parameters will yield an opaque DeviceRGB black.
- static IDOMColorPtr createSolidCmyk (IEDLClassFactory *pFactory, float c=0.0f, float m=0.0f, float y=0.0f, float k=0.0f)

  Simplified color creation for DeviceCMYK colours. Default parameters will yield an opaque DeviceCMYK white.
- static EDL_API IDOMColorPtr create (IEDLClassFactory *pFactory, const IDOMColorSpacePtr &space, double alpha, double component)

  Simplified color creation routine for color spaces using a single component. Throws an IEDLError on failure.
- static EDL_API IDOMColorPtr create (IEDLClassFactory *pFactory, const IDOMColorSpacePtr &space, double alpha, double component1, double component2)

  Simplified color creation routine for color spaces using two components. Throws an IEDLError on failure.
- static EDL_API IDOMColorPtr create (IEDLClassFactory *pFactory, const IDOMColorSpacePtr &space, double alpha, double component1, double component2, double component3)

  Simplified color creation routine for color spaces using three components. Throws an IEDLError on failure.
- static EDL_API IDOMColorPtr create (IEDLClassFactory *pFactory, const IDOMColorSpacePtr &space, double alpha, double component1, double component2, double component3, double component4)

  Simplified color creation routine for color spaces that take four components. Throws an IEDLError on failure.
- static EDL_API IDOMColorPtr create (IEDLClassFactory *pFactory, const IDOMColorSpacePtr &space, double alpha, double component1, double component2, double component3, double component4, double component5)

  Simplified color creation routine for color spaces that take five components. Throws an IEDLError on failure.
- static EDL_API IDOMColorPtr create (IEDLClassFactory *pFactory, const IDOMColorSpacePtr &space, double alpha, double component1, double component2, double component3, double component4, double component5, double component6)

  Simplified color creation routine for color spaces that take six components. Throws an IEDLError on failure.

- static EDL_API IDOMColorPtr create (IEDLClassFactory *pFactory, const IDOMColorSpacePtr &space, double alpha, double component1, double component2, double component3, double component4, double component5, double component6, double component7, double component8=0.0, double component9=0.0, double component10=0.0, double component11=0.0, double component12=0.0, double component13=0.0, double component14=0.0, double component15=0.0, double component16=0.0, double component17=0.0, double component18=0.0, double component19=0.0, double component20=0.0, double component21=0.0, double component22=0.0, double component23=0.0, double component24=0.0, double component25=0.0, double component26=0.0, double component27=0.0, double component28=0.0, double component29=0.0, double component30=0.0, double component31=0.0, double component32=0.0)

  Simplified color creation routine for color spaces that take any number of components from 7 through to the maximum of 32. Throws an IEDLError on failure.
- static EDL_API IDOMColorPtr createFromVect (IEDLClassFactory *pFactory, const IDOMColorSpacePtr &space, float alpha, const CEDLVector< float > &components)

  Simplified color creation routine. Throws an IEDLError on failure.
- static EDL_API IDOMColorPtr createFromArray (IEDLClassFactory *pFactory, const IDOMColorSpacePtr &space, float alpha, const float *components)

  Simplified color creation routine. Throws an IEDLError on failure.
- static const CClassID & classID ()

  Retrieves the class id of IDOMColor.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.166.1 Detailed Description

Holds a single color value. The color values themselves are held as floating point values for all color spaces. For some spaces (such as indexed color spaces) the values will be integral, but still stored as floats.

The color values themselves cannot be interpreted without reference to the referenced color space.

IDOMColor nodes are used to specify the colors of lines and strokes, not of images. Hence if an ArcSegment is drawn with a blue line, the blue is specified by an IDOMColor node.

7.166.2 Member Function Documentation

classID()

```
static const CClassID & IDOMColor::classID ( )  [inline], [static]
```

Retrieves the class id of IDOMColor.

Returns

CClassID Returns the class id of the element.

create() [1/7]

```
static EDL_API IDOMColorPtr IDOMColor::create (
            IEDLClassFactory * pFactory,
            const IDOMColorSpacePtr & space,
            double alpha,
            double component )  [static]
```

Simplified color creation routine for color spaces using a single component. Throws an IEDLError on failure.

Parameters

pFactory	The EDL Class factory to use.
space	The color space to use. This must take one color component.
alpha	The alpha to use.
component	The color components to use.

Returns

IDOMColorPtr The new color.

create() [2/7]

```
static EDL_API IDOMColorPtr IDOMColor::create (
            IEDLClassFactory * pFactory,
            const IDOMColorSpacePtr & space,
            double alpha,
            double component1,
            double component2 )  [static]
```

Simplified color creation routine for color spaces using two components. Throws an IEDLError on failure.

Parameters

pFactory	The EDL Class factory to use.
space	The color space to use. This must take two color components.
alpha	The alpha to use.
component1	The first color components to use.
component2	The second color components to use.

Returns

IDOMColorPtr The new color.

create() [3/7]

```
static EDL_API IDOMColorPtr IDOMColor::create (
            IEDLClassFactory * pFactory,
            const IDOMColorSpacePtr & space,
            double alpha,
            double component1,
            double component2,
            double component3 )  [static]
```

Simplified color creation routine for color spaces using three components. Throws an IEDLError on failure.

Parameters

pFactory	The EDL Class factory to use.

Parameters

space	The color space to use. This must take three color components.
alpha	The alpha to use.
component1	The first color components to use.
component2	The second color components to use.
component3	The third color components to use.

Returns

IDOMColorPtr The new color.

create() [4/7]

```
static EDL_API IDOMColorPtr IDOMColor::create (
            IEDLClassFactory * pFactory,
            const IDOMColorSpacePtr & space,
            double alpha,
            double component1,
            double component2,
            double component3,
            double component4 )  [static]
```

Simplified color creation routine for color spaces that take four components. Throws an IEDLError on failure.

Parameters

pFactory	The EDL Class factory to use.
space	The color space to use. This must take four color components.
alpha	The alpha to use.
component1	The first color components to use.
component2	The second color components to use.
component3	The third color components to use.
component4	The fourth color components to use.

Returns

IDOMColorPtr The new color.

create() [5/7]

```
static EDL_API IDOMColorPtr IDOMColor::create (
            IEDLClassFactory * pFactory,
            const IDOMColorSpacePtr & space,
            double alpha,
            double component1,
            double component2,
            double component3,
```

```
            double component4,
            double component5 )  [static]
```

Simplified color creation routine for color spaces that take five components. Throws an IEDLError on failure.

Parameters

pFactory	The EDL Class factory to use.
space	The color space to use. This must take five color components.
alpha	The alpha to use.
component1	The first color components to use.
component2	The second color components to use.
component3	The third color components to use.
component4	The fourth color components to use.
component5	The fifth color components to use.

Returns

IDOMColorPtr The new color.

create() [6/7]

```
static EDL_API IDOMColorPtr IDOMColor::create (
            IEDLClassFactory * pFactory,
            const IDOMColorSpacePtr & space,
            double alpha,
            double component1,
            double component2,
            double component3,
            double component4,
            double component5,
            double component6 )  [static]
```

Simplified color creation routine for color spaces that take six components. Throws an IEDLError on failure.

Parameters

pFactory	The EDL Class factory to use.
space	The color space to use. This must take six color components.
alpha	The alpha to use.
component1	The first color components to use.
component2	The second color components to use.
component3	The third color components to use.
component4	The fourth color components to use.
component5	The fifth color components to use.
component6	The sixth color components to use.

Returns

IDOMColorPtr The new color.

create() [7/7]

```
static EDL_API IDOMColorPtr IDOMColor::create (
            IEDLClassFactory * pFactory,
```

```
const IDOMColorSpacePtr & space,
double alpha,
double component1,
double component2,
double component3,
double component4,
double component5,
double component6,
double component7,
double component8 = 0.0,
double component9 = 0.0,
double component10 = 0.0,
double component11 = 0.0,
double component12 = 0.0,
double component13 = 0.0,
double component14 = 0.0,
double component15 = 0.0,
double component16 = 0.0,
double component17 = 0.0,
double component18 = 0.0,
double component19 = 0.0,
double component20 = 0.0,
double component21 = 0.0,
double component22 = 0.0,
double component23 = 0.0,
double component24 = 0.0,
double component25 = 0.0,
double component26 = 0.0,
double component27 = 0.0,
double component28 = 0.0,
double component29 = 0.0,
double component30 = 0.0,
double component31 = 0.0,
double component32 = 0.0 )  [static]
```

Simplified color creation routine for color spaces that take any number of components from 7 through to the maximum of 32. Throws an IEDLError on failure.

Parameters

pFactory	The EDL Class factory to use.
space	The color space to use. This must take between 7 and 32 color components.
alpha	The alpha to use.
component1	The first color component to use.
component2	The second color component to use.
component3	The third color component to use.
component4	The fourth color component to use.
component5	The fifth color component to use.
component6	The sixth color component to use.
component7	The seventh color component to use.
component8	The eighth color component to use (if applicable).
component9	The ninth color component to use (if applicable).
component10	The tenth color component to use (if applicable).
component11	The eleventh color component to use (if applicable).
component12	The twelth color component to use (if applicable).
component13	The thirteenth color component to use (if applicable).

Parameters

component14	The fourteenth color component to use (if applicable).
component15	The fifteenth color component to use (if applicable).
component16	The sixteenth color component to use (if applicable).
component17	The seventeenth color component to use (if applicable).
component18	The eightteenth color component to use (if applicable).
component19	The nineteenth color component to use (if applicable).
component20	The twentieth color component to use (if applicable).
component21	The twenty-first color component to use (if applicable).
component22	The twenty-second color component to use (if applicable).
component23	The twenty-third color component to use (if applicable).
component24	The twenty-fourth color component to use (if applicable).
component25	The twenty-fifth color component to use (if applicable).
component26	The twenty-sixth color component to use (if applicable).
component27	The twenty-seventh color component to use (if applicable).
component28	The twenty-eighth color component to use (if applicable).
component29	The twenty-ninth color component to use (if applicable).
component30	The thirtieth color component to use (if applicable).
component31	The thirty-first color component to use (if applicable).
component32	The thirty-second color component to use (if applicable).

Returns

IDOMColorPtr The new color.

createFromArray()

```
static EDL_API IDOMColorPtr IDOMColor::createFromArray (
            IEDLClassFactory * pFactory,
            const IDOMColorSpacePtr & space,
            float alpha,
            const float * components )  [static]
```

Simplified color creation routine. Throws an IEDLError on failure.

Parameters

pFactory	The EDL Class factory to use.
space	The color space to use.
alpha	The alpha to use.
components	An array of color components to use, as floats or doubles. The number of floats MUST match the number of components expected of the color space.

Returns

IDOMColorPtr The new color.

createFromVect()

```
static EDL_API IDOMColorPtr IDOMColor::createFromVect (
            IEDLClassFactory * pFactory,
            const IDOMColorSpacePtr & space,
            float alpha,
            const CEDLVector< float > & components )  [static]
```

Simplified color creation routine. Throws an IEDLError on failure.

Parameters

pFactory	The EDL Class factory to use.
space	The color space to use.
alpha	The alpha to use.
components	A vector of color components to use, as floats. The number of components MUST match the number of components expected of the color space.

Returns

IDOMColorPtr The new color.

createSolidCmyk()

```
static IDOMColorPtr IDOMColor::createSolidCmyk (
            IEDLClassFactory * pFactory,
            float c = 0.0f,
            float m = 0.0f,
            float y = 0.0f,
            float k = 0.0f )  [inline], [static]
```

Simplified color creation for DeviceCMYK colours. Default parameters will yield an opaque DeviceCMYK white.

Parameters

pFactory	The EDL Class factory to use.
c	The cyan level to use.
m	The magenta level to use.
y	The yellow level to use.
k	The black level to use.

Returns

IDOMColorPtr The new brush.

createSolidGray()

```
static IDOMColorPtr IDOMColor::createSolidGray (
            IEDLClassFactory * pFactory,
            float gray = 0.0f )  [inline], [static]
```

Simplified color creation for DeviceGray colours. Default parameters will yield an opaque DeviceGray black.

Parameters

pFactory	The EDL Class factory to use.
gray	The gray level to use (0 is black).

Returns

IDOMColorPtr The new brush.

createSolidRgb()

```
static IDOMColorPtr IDOMColor::createSolidRgb (
            IEDLClassFactory * pFactory,
            float r = 0.0f,
            float g = 0.0f,
            float b = 0.0f )  [inline], [static]
```

Simplified color creation for DeviceRGB colours. Default parameters will yield an opaque DeviceRGB black.

Parameters

pFactory	The EDL Class factory to use.
r	The red level to use.
g	The green level to use.
b	The blue level to use.

Returns

IDOMColorPtr The new brush.

getAlpha()

```
virtual float IDOMColor::getAlpha ( ) const  [pure virtual]
```

Retrieves the alpha channel value. The alpha channel value specifies the transparency of the color.

Alpha values are between zero and 1. A value of 0 means that the color does not have any coverage information and is fully transparent; a value of 1 means that the color is fully opaque because the geometry completely overlapped the pixel.

Returns

float The alpha channel value

getColorSpace()

```
virtual IDOMColorSpacePtr IDOMColor::getColorSpace ( )  [pure virtual]
```

Retrieves the color space.

A device color space simply describes the range of colors, or gamut, that a camera can see, a printer can print, or a monitor can display. Editing color spaces, on the other hand, such as Adobe RGB or sRGB, are device-independent. They also determine a color range you can work in. Their design allows you to edit images in a controlled, consistent manner. A device color space is tied to the device it describes. An editing space, on the other hand, is gray-balanced colors with equal amounts of red, green, and blue appear neutral. Editing spaces also are perceptually uniform; that is, changes to lightness, hue, or saturation are applied equally to all the colors in the image.

Returns

smartptr Smart pointer to the colorspace.

getComponentValue()

```
virtual float IDOMColor::getComponentValue (
            uint32 component )  [pure virtual]
```

Retrieves the value of a component.

Parameters

component	Index of the component.

Returns

float Component value

setAlpha()

```
virtual void IDOMColor::setAlpha (
            float a )  [pure virtual]
```

Sets the alpha channel value.

Parameters

a	Desired new alpha channel value.

setColorSpace() [1/2]

```
virtual void IDOMColor::setColorSpace (
            const IDOMColorSpacePtr & colorSpace,
            bool setDefaultColor = true )  [pure virtual]
```

Set the color space.

If the new color space has has the same number of components as the old, the values will be retained and clipped to the range of the new color space, if applicable. Otherwise if setDefaultColor is true, a default value will be set based on the minima of the range of the new color space.

Parameters

colorSpace	Smart pointer to the desired new color space
setDefaultColor	If true, and the new space has a different number of components than the previous space, a default color will be set.

setColorSpace() [2/2]

```
virtual void IDOMColor::setColorSpace (
            const IDOMColorSpacePtr & colorSpace,
            eRenderingIntent intent,
            eBlackPointCompensation bpc,
            IEDLClassFactory * pFactory )  [pure virtual]
```

Set the color space, converting color values from the previous color space to the new.

The destination color space must not be an Indexed or DeviceN color space. If the current color space is a DeviceN or Indexed space, if will be converted to the Alternate or Underlying color space before color conversion takes place. Further, if the resulting color is not visible (due to a DeviceN colorspace containing nothing but the special "None" colorant, then the resulting colour will be given zero alpha and the default components for the destination color space will be set.

Parameters

colorSpace	Smart pointer to the desired new color space.
intent	The rendering intent to be used for conversion. The rendering intent refers to the way the CMM (Color Management Module) will handle out-of-gamut colors during a conversion from one color space to another. The ICC specification includes four different rendering intents: perceptual, relative colorimetric, absolute colorimetric, and saturation.

See also

eRenderingIntent

Parameters

bpc	Black point compensation handling. Black point compensation refers to the way the CMM (Color Management Module) will handle shadows in color conversion where the black point of the input and output color spaces differ. If in doubt, use eBPCDefault.
pFactory	The EDL class factory.

setComponentValue()

```
virtual void IDOMColor::setComponentValue (
```

```
            uint32 component,
            float value )  [pure virtual]
```

Sets a component value. The value will be clipped to the range of the color space, if known.

Parameters

component	Index of the component.
value	New value of the component.

testGamut()

```
virtual bool IDOMColor::testGamut (
            IEDLClassFactory * pFactory,
            const IDOMColorSpacePtr & colorSpace,
            eRenderingIntent intent,
            eBlackPointCompensation bpc )  [pure virtual]
```

Tests the color is within the color gamut for the given color space.

The color space must not be an Indexed or DeviceN color space. If the current color space is a DeviceN or Indexed space, if will be converted to the Alternate or Underlying color space before color conversion takes place.

Parameters

pFactory	The EDL class factory.
colorSpace	Smart pointer to the color space to test against.
intent	The rendering intent to be used for conversion. The rendering intent refers to the way the CMM (Color Management Module) will handle out-of-gamut colors during a conversion from one color space to another. The ICC specification includes four different rendering intents: perceptual, relative colorimetric, absolute colorimetric, and saturation.

See also

eRenderingIntent

Parameters

bpc	Black point compensation handling. Black point compensation refers to the way the CMM (Color Management Module) will handle shadows in color conversion where the black point of the input and output color spaces differ. If in doubt, use eBPCDefault.

Returns

bool false, if the color is Out-Of-Gamut, or true if within gamut.

The documentation for this class was generated from the following file:

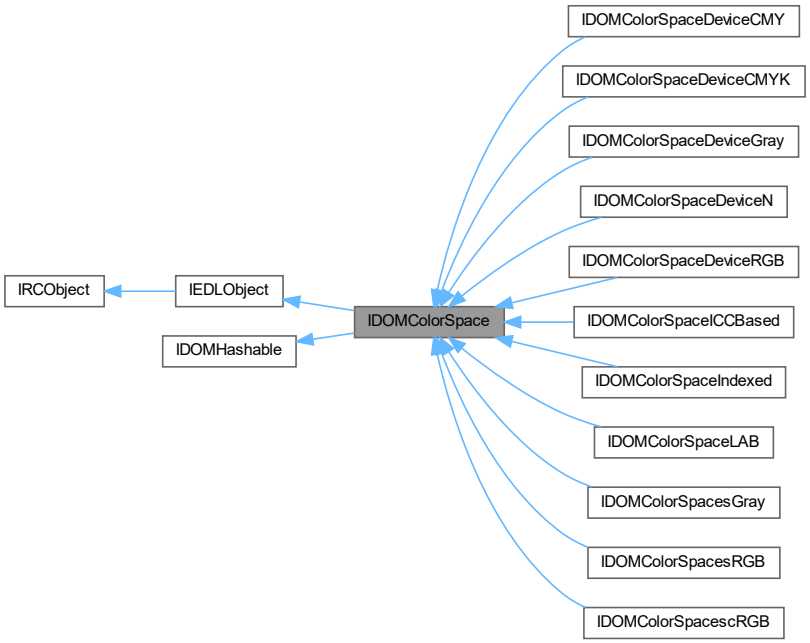
- idomcolor.h

7.167 IDOMColorSpace Class Reference

IDOMColorSpace interface.

`#include <idomcolorspace.h>`

Inheritance diagram for IDOMColorSpace:



Public Types

- enum eColorSpaceType {
  eDeviceRGB , eDeviceGray , eDeviceCMYK , esRGB ,
  esGray , escRGB , eICCBased , eIndexed ,
  eDeviceN , eLAB , eDeviceCMY , **eNumColorSpaceTypes** = 11 }

  Color spaces type enumeration.

Public Member Functions

- virtual eColorSpaceType getColorSpaceType ()=0

  Retrieves the color space type.
- virtual uint8 getNumComponents ()=0

  Retrieves the number of components that are in colors in this color space.
- virtual bool getComponentsHaveSameRange ()=0

  Checks if this color space has the same range for all its components. Some color spaces (such as Lab and ICC) may not have the same range for all its components. If a color space does have the same range for all its components, you only need to call getComponentRange() once to find the range for all components.
- virtual bool getComponentRange (int component, float &low, float &high)=0

  Retrieves the expected range of component values for a given channel, if applicable.

- virtual eRenderingIntent getDefaultRenderingIntent ()=0

  Retrieves the default rendering intent for this color space.
- virtual EDLRawString getColorantName (uint8 component) const =0

  Determine the colorant name for a colorant index. If a colorant name cannot be determined, an IEDLError with type EDL_ERR_NO_COLOR_NAME will be thrown. Cannot be called on Indexed or LAB color spaces. For device spaces and simple ICC color spaces, the basic process color name will be returned (ie one of Gray, Red, Green, Blue, Cyan, Magenta, Yellow, Black as appropriate).
- virtual bool equals (const IDOMColorSpacePtr &colorSpace, bool exact=false)=0

  Determines if the given color space is equivalent to this color space.
- virtual bool similar (const IDOMColorSpacePtr &colorSpace, eRenderingIntent intent, eBlackPointCompensation bpc)=0

  Determines if the given color space is very similar to this color space for the given rendering intent. Color spaces are tested via selective sampling and are considered similar if the samples show the same results to a 10 bit accuracy. This can be an expensive operation for complicated color spaces.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.167.1 Detailed Description

IDOMColorSpace interface.

A color space describes a range of colors appropriate to a given context. A device color space describes the range of colors that a camera can see, a printer can print, or a monitor can display, and is tied to the device it describes. Editing color spaces, on the other hand, such as Adobe RGB or sRGB, are device-independent, determining a color range you can work in. They are designed to be gray balanced, such that colors with equal amounts of red, green, and blue appear neutral, and allow you to edit images in a controlled, consistent manner. Editing spaces also are perceptually uniform; that is, changes to lightness, hue, or saturation are applied equally to all the colors in the image.

Instances of these objects may throw IEDLError exceptions on failure.

7.167.2 Member Function Documentation

equals()

```
virtual bool IDOMColorSpace::equals (
            const IDOMColorSpacePtr & colorSpace,
            bool exact = false )  [pure virtual]
```

Determines if the given color space is equivalent to this color space.

Parameters

colorSpace	The color space to compare with.
exact	If false, we consider spaces equivalent if their intercepted results are the same. For example, in the default Mako configuration, sRGB is the intercept space for DeviceRGB, and therefore if exact is false then sRGB is considered equivalent to DeviceRGB. Pass exact as true if we wish to treat these situations as non-equivalent.

Returns

bool True if equal, false otherwise.

getColorantName()

```
virtual EDLRawString IDOMColorSpace::getColorantName (
            uint8 component ) const  [pure virtual]
```

Determine the colorant name for a colorant index. If a colorant name cannot be determined, an IEDLError with type EDL_ERR_NO_COLOR_NAME will be thrown. Cannot be called on Indexed or LAB color spaces. For device spaces and simple ICC color spaces, the basic process color name will be returned (ie one of Gray, Red, Green, Blue, Cyan, Magenta, Yellow, Black as appropriate).

Returns

EDLRawString On success, the colorant name.

getColorSpaceType()

```
virtual eColorSpaceType IDOMColorSpace::getColorSpaceType ( )  [pure virtual]
```

Retrieves the color space type.

See also

eColorSpaceType

Returns

eColorSpaceType. Retrieves the color space type

getComponentRange()

```
virtual bool IDOMColorSpace::getComponentRange (
            int component,
            float & low,
            float & high )  [pure virtual]
```

Retrieves the expected range of component values for a given channel, if applicable.

Parameters

component	The index of the component whose range is being queried
low	Reference parameter to receive the low bound of the component's value range.
high	Reference parameter to receive the high bound of the component's value range.

Returns

bool Returns true if the given component has a range, false otherwise.

getComponentsHaveSameRange()

```
virtual bool IDOMColorSpace::getComponentsHaveSameRange ( )  [pure virtual]
```

Checks if this color space has the same range for all its components. Some color spaces (such as Lab and ICC) may not have the same range for all its components. If a color space does have the same range for all its components, you only need to call getComponentRange() once to find the range for all components.

Returns

bool Returns true if all the components in the color space have the same range, false if they do not.

getDefaultRenderingIntent()

```
virtual eRenderingIntent IDOMColorSpace::getDefaultRenderingIntent ( )  [pure virtual]
```

Retrieves the default rendering intent for this color space.

Returns

eRenderingIntent The default rendering intent.

getNumComponents()

```
virtual uint8 IDOMColorSpace::getNumComponents ( )  [pure virtual]
```

Retrieves the number of components that are in colors in this color space.

Returns

uint8 The number of components for colors in this color space.

similar()

```
virtual bool IDOMColorSpace::similar (
            const IDOMColorSpacePtr & colorSpace,
            eRenderingIntent intent,
            eBlackPointCompensation bpc )  [pure virtual]
```

Determines if the given color space is very similar to this color space for the given rendering intent. Color spaces are tested via selective sampling and are considered similar if the samples show the same results to a 10 bit accuracy. This can be an expensive operation for complicated color spaces.

Parameters

colorSpace	The color space to compare with.
intent	The rendering intent to use.
bpc	The black point compensation treatment. If in double, use eBPCDefault.

Returns

bool True if similar, false otherwise.

The documentation for this class was generated from the following file:

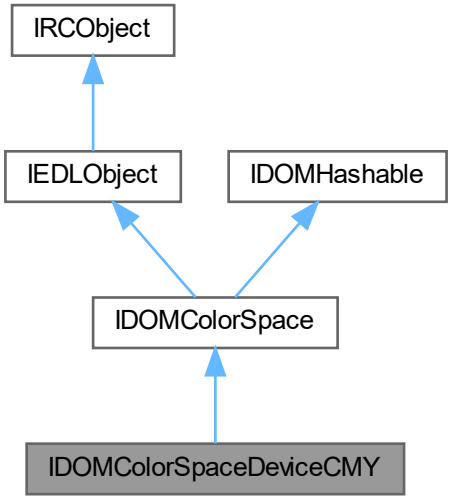
- idomcolorspace.h

7.168 IDOMColorSpaceDeviceCMY Class Reference

Represents the default CMY color space. NOTE: Currently for internal use only; Do not use this color space in your own applications.

`#include <idomcolorspace.h>`

Inheritance diagram for IDOMColorSpaceDeviceCMY:



Static Public Member Functions

- static EDL_API IDOMColorSpaceDeviceCMYPtr create (IEDLClassFactory *pFactory)

  Create a DeviceCMY Color Space. Throws an IEDLError on failure.
- static const CClassID & classID ()

  Retrieves class id of IDOMColorSpaceDeviceCMY.

Additional Inherited Members

Public Types inherited from IDOMColorSpace

- enum eColorSpaceType {
  eDeviceRGB , eDeviceGray , eDeviceCMYK , esRGB ,
  esGray , escRGB , eICCBased , eIndexed ,
  eDeviceN , eLAB , eDeviceCMY , **eNumColorSpaceTypes** = 11 }

  Color spaces type enumeration.

Public Member Functions inherited from IDOMColorSpace

- virtual eColorSpaceType getColorSpaceType ()=0

  Retrieves the color space type.
- virtual uint8 getNumComponents ()=0

  Retrieves the number of components that are in colors in this color space.
- virtual bool getComponentsHaveSameRange ()=0

  Checks if this color space has the same range for all its components. Some color spaces (such as Lab and ICC) may not have the same range for all its components. If a color space does have the same range for all its components, you only need to call getComponentRange() once to find the range for all components.
- virtual bool getComponentRange (int component, float &low, float &high)=0

  Retrieves the expected range of component values for a given channel, if applicable.
- virtual eRenderingIntent getDefaultRenderingIntent ()=0

  Retrieves the default rendering intent for this color space.
- virtual EDLRawString getColorantName (uint8 component) const =0

  Determine the colorant name for a colorant index. If a colorant name cannot be determined, an IEDLError with type EDL_ERR_NO_COLOR_NAME will be thrown. Cannot be called on Indexed or LAB color spaces. For device spaces and simple ICC color spaces, the basic process color name will be returned (ie one of Gray, Red, Green, Blue, Cyan, Magenta, Yellow, Black as appropriate).
- virtual bool equals (const IDOMColorSpacePtr &colorSpace, bool exact=false)=0

  Determines if the given color space is equivalent to this color space.
- virtual bool similar (const IDOMColorSpacePtr &colorSpace, eRenderingIntent intent, eBlackPointCompensation bpc)=0

  Determines if the given color space is very similar to this color space for the given rendering intent. Color spaces are tested via selective sampling and are considered similar if the samples show the same results to a 10 bit accuracy. This can be an expensive operation for complicated color spaces.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.168.1 Detailed Description

Represents the default CMY color space. NOTE: Currently for internal use only; Do not use this color space in your own applications.

7.168.2 Member Function Documentation

classID()

```
static const CClassID & IDOMColorSpaceDeviceCMY::classID ( )  [inline], [static]
```

Retrieves class id of IDOMColorSpaceDeviceCMY.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMColorSpaceDeviceCMYPtr IDOMColorSpaceDeviceCMY::create (
            IEDLClassFactory * pFactory )  [static]
```

Create a DeviceCMY Color Space. Throws an IEDLError on failure.

Parameters

pFactory	The EDL Class factory to use.

Returns

IDOMColorSpaceDeviceCMYPtr The new color space.

The documentation for this class was generated from the following file:

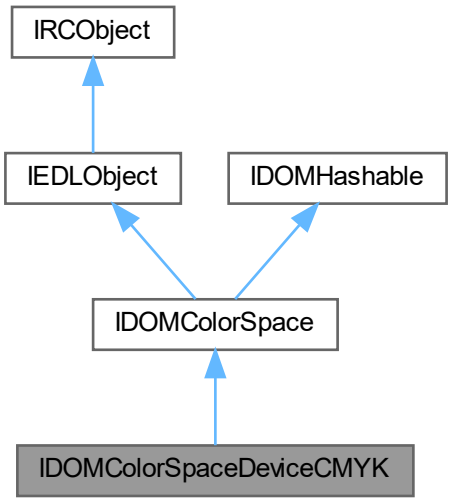
- idomcolorspace.h

7.169 IDOMColorSpaceDeviceCMYK Class Reference

Represents the default CMYK color space.

```
#include <idomcolorspace.h>
```

Inheritance diagram for IDOMColorSpaceDeviceCMYK:



Static Public Member Functions

- static EDL_API IDOMColorSpaceDeviceCMYKPtr create (IEDLClassFactory *pFactory)

  Create a DeviceCMYK Color Space. Throws an IEDLError on failure.
- static const CClassID & classID ()

  Retrieves class id of IDOMColorSpaceDeviceCMYK.

Additional Inherited Members

Public Types inherited from IDOMColorSpace

- enum eColorSpaceType {
  eDeviceRGB , eDeviceGray , eDeviceCMYK , esRGB ,
  esGray , escRGB , eICCBased , eIndexed ,
  eDeviceN , eLAB , eDeviceCMY , **eNumColorSpaceTypes** = 11 }

  Color spaces type enumeration.

Public Member Functions inherited from IDOMColorSpace

- virtual eColorSpaceType getColorSpaceType ()=0

  Retrieves the color space type.
- virtual uint8 getNumComponents ()=0

  Retrieves the number of components that are in colors in this color space.
- virtual bool getComponentsHaveSameRange ()=0

  Checks if this color space has the same range for all its components. Some color spaces (such as Lab and ICC) may not have the same range for all its components. If a color space does have the same range for all its components, you only need to call getComponentRange() once to find the range for all components.

- virtual bool getComponentRange (int component, float &low, float &high)=0

  Retrieves the expected range of component values for a given channel, if applicable.
- virtual eRenderingIntent getDefaultRenderingIntent ()=0

  Retrieves the default rendering intent for this color space.
- virtual EDLRawString getColorantName (uint8 component) const =0

  Determine the colorant name for a colorant index. If a colorant name cannot be determined, an IEDLError with type EDL_ERR_NO_COLOR_NAME will be thrown. Cannot be called on Indexed or LAB color spaces. For device spaces and simple ICC color spaces, the basic process color name will be returned (ie one of Gray, Red, Green, Blue, Cyan, Magenta, Yellow, Black as appropriate).
- virtual bool equals (const IDOMColorSpacePtr &colorSpace, bool exact=false)=0

  Determines if the given color space is equivalent to this color space.
- virtual bool similar (const IDOMColorSpacePtr &colorSpace, eRenderingIntent intent, eBlackPointCompensation bpc)=0

  Determines if the given color space is very similar to this color space for the given rendering intent. Color spaces are tested via selective sampling and are considered similar if the samples show the same results to a 10 bit accuracy. This can be an expensive operation for complicated color spaces.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.169.1 Detailed Description

Represents the default CMYK color space.

7.169.2 Member Function Documentation

classID()

```
static const CClassID & IDOMColorSpaceDeviceCMYK::classID ( )  [inline], [static]
```

Retrieves class id of IDOMColorSpaceDeviceCMYK.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMColorSpaceDeviceCMYKPtr IDOMColorSpaceDeviceCMYK::create (
            IEDLClassFactory * pFactory )  [static]
```

Create a DeviceCMYK Color Space. Throws an IEDLError on failure.

Parameters

pFactory	The EDL Class factory to use.

Returns

IDOMColorSpaceDeviceCMYKPtr The new color space.

The documentation for this class was generated from the following file:

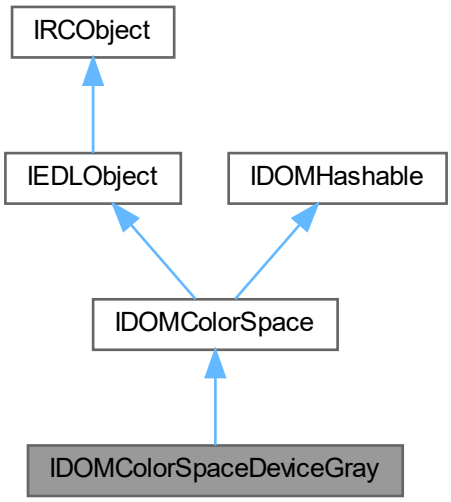
- idomcolorspace.h

7.170 IDOMColorSpaceDeviceGray Class Reference

IDOMColorSpaceDeviceGray interface.

```
#include <idomcolorspace.h>
```

Inheritance diagram for IDOMColorSpaceDeviceGray:



Static Public Member Functions

- static EDL_API IDOMColorSpaceDeviceGrayPtr create (IEDLClassFactory *pFactory)

  Create a DeviceGray Color Space. Throws an IEDLError on failure.
- static const CClassID & classID ()

  Retrieves class id of IDOMColorSpaceDeviceGray.

Additional Inherited Members

Public Types inherited from IDOMColorSpace

- enum eColorSpaceType {
  eDeviceRGB , eDeviceGray , eDeviceCMYK , esRGB ,
  esGray , escRGB , eICCBased , eIndexed ,
  eDeviceN , eLAB , eDeviceCMY , **eNumColorSpaceTypes** = 11 }

  Color spaces type enumeration.

Public Member Functions inherited from IDOMColorSpace

- virtual eColorSpaceType getColorSpaceType ()=0

  Retrieves the color space type.
- virtual uint8 getNumComponents ()=0

  Retrieves the number of components that are in colors in this color space.
- virtual bool getComponentsHaveSameRange ()=0

  Checks if this color space has the same range for all its components. Some color spaces (such as Lab and ICC) may not have the same range for all its components. If a color space does have the same range for all its components, you only need to call getComponentRange() once to find the range for all components.

- virtual bool getComponentRange (int component, float &low, float &high)=0

  Retrieves the expected range of component values for a given channel, if applicable.
- virtual eRenderingIntent getDefaultRenderingIntent ()=0

  Retrieves the default rendering intent for this color space.
- virtual EDLRawString getColorantName (uint8 component) const =0

  Determine the colorant name for a colorant index. If a colorant name cannot be determined, an IEDLError with type EDL_ERR_NO_COLOR_NAME will be thrown. Cannot be called on Indexed or LAB color spaces. For device spaces and simple ICC color spaces, the basic process color name will be returned (ie one of Gray, Red, Green, Blue, Cyan, Magenta, Yellow, Black as appropriate).
- virtual bool equals (const IDOMColorSpacePtr &colorSpace, bool exact=false)=0

  Determines if the given color space is equivalent to this color space.
- virtual bool similar (const IDOMColorSpacePtr &colorSpace, eRenderingIntent intent, eBlackPointCompensation bpc)=0

  Determines if the given color space is very similar to this color space for the given rendering intent. Color spaces are tested via selective sampling and are considered similar if the samples show the same results to a 10 bit accuracy. This can be an expensive operation for complicated color spaces.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.170.1 Detailed Description

IDOMColorSpaceDeviceGray interface.

7.170.2 Member Function Documentation

classID()

```
static const CClassID & IDOMColorSpaceDeviceGray::classID ( )  [inline], [static]
```

Retrieves class id of IDOMColorSpaceDeviceGray.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMColorSpaceDeviceGrayPtr IDOMColorSpaceDeviceGray::create (
            IEDLClassFactory * pFactory )  [static]
```

Create a DeviceGray Color Space. Throws an IEDLError on failure.

Parameters

pFactory	The EDL Class factory to use.

Returns

IDOMColorSpaceDeviceGrayPtr The new color space.

The documentation for this class was generated from the following file:

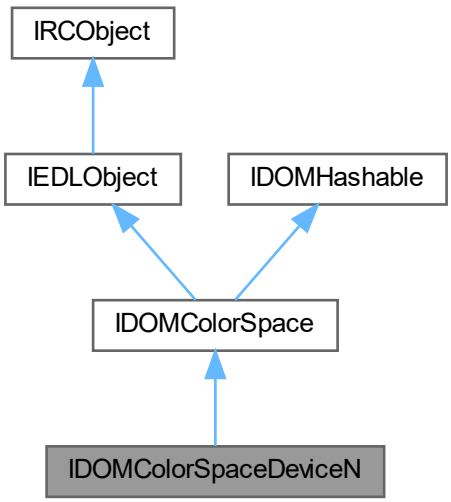
- idomcolorspace.h

7.171 IDOMColorSpaceDeviceN Class Reference

This color space is analogous to the PostScript/PDF DeviceN/Separation color spaces.

```
#include <idomcolorspace.h>
```

Inheritance diagram for IDOMColorSpaceDeviceN:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual bool getIsNChannel () const =0

  Get if the color space represents an NChannel color space.
- virtual IDOMColorSpacePtr getAlternateColorSpace () const =0

  Get the alternate color space for this color space, which must be present. An exception of type IEDLError is thrown if it is missing.
- virtual bool getHasRealColorant () const =0

  Determine if this DeviceN color space contains only the special "None" colorant (or the strange case where there are multiple colorants all with the name "None"). Such color spaces will never render or mark a page.
- virtual IDOMFunctionPtr getTintTransform () const =0

  Get the function that maps components of this space to component(s) of the alternate color space. Must be present. An exception of type IEDLError is thrown if it is missing.
- virtual IDOMDeviceNColorantPtr getColorant (uint8 component) const =0

  Get the colorant information for a component at the given index. An exception of type IEDLError is thrown on failure.
- virtual IDOMColorSpacePtr getProcessColorSpace () const =0

  Get the Process color space associated with this DeviceN space. (See Table 4.22 of the PDF 1.7 Reference). Optional.
- virtual const CEDLRawStringVect & getProcessComponentNames () const =0

  Get the vector of Process component names to those in the associated process color space (see Table 4.22 of the PDF 1.7 reference). An exception of IEDLError is thrown on failure, which should not happen.
- virtual const CEDLRawStringVect & getPrintingOrder () const =0

Get the list of component names describing the printing order of the device N colorants. An exception of IEDLError is thrown on failure, which should not happen.

- virtual CColorantInfoVect getColorantInfo (IEDLClassFactory *factory, const IDOMColorSpacePtr &colorantSpace=IDOMColorSpacePtr()) const =0

  Convenience method to populate a CColorantInfoVect from the DeviceN colorspace.

Public Member Functions inherited from IDOMColorSpace

- virtual eColorSpaceType getColorSpaceType ()=0

  Retrieves the color space type.
- virtual uint8 getNumComponents ()=0

  Retrieves the number of components that are in colors in this color space.
- virtual bool getComponentsHaveSameRange ()=0

  Checks if this color space has the same range for all its components. Some color spaces (such as Lab and ICC) may not have the same range for all its components. If a color space does have the same range for all its components, you only need to call getComponentRange() once to find the range for all components.
- virtual bool getComponentRange (int component, float &low, float &high)=0

  Retrieves the expected range of component values for a given channel, if applicable.
- virtual eRenderingIntent getDefaultRenderingIntent ()=0

  Retrieves the default rendering intent for this color space.
- virtual EDLRawString getColorantName (uint8 component) const =0

  Determine the colorant name for a colorant index. If a colorant name cannot be determined, an IEDLError with type EDL_ERR_NO_COLOR_NAME will be thrown. Cannot be called on Indexed or LAB color spaces. For device spaces and simple ICC color spaces, the basic process color name will be returned (ie one of Gray, Red, Green, Blue, Cyan, Magenta, Yellow, Black as appropriate).
- virtual bool equals (const IDOMColorSpacePtr &colorSpace, bool exact=false)=0

  Determines if the given color space is equivalent to this color space.
- virtual bool similar (const IDOMColorSpacePtr &colorSpace, eRenderingIntent intent, eBlackPointCompensation bpc)=0

  Determines if the given color space is very similar to this color space for the given rendering intent. Color spaces are tested via selective sampling and are considered similar if the samples show the same results to a 10 bit accuracy. This can be an expensive operation for complicated color spaces.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static EDL_API IDOMColorSpaceDeviceNPtr create (IEDLClassFactory *pFactory, const CEDLRaw↩ StringVect &colorants, const IDOMColorSpacePtr &alternate, const IDOMFunctionPtr &tintTransform, const IDOMColorSpacePtr &processColorSpace=IDOMColorSpacePtr())

  Create a simple DeviceN Color Space. Throws an IEDLError on failure.
- static EDL_API IDOMColorSpaceDeviceNPtr create (IEDLClassFactory *pFactory, const CDevice↩ NColorantVect &colorants, const IDOMColorSpacePtr &alternate, const IDOMFunctionPtr &tintTransform, const IDOMColorSpacePtr &processColorSpace=IDOMColorSpacePtr(), const CEDLRawStringVect &processComponents=CEDLRawStringVect(), const CEDLRawStringVect &printingOrder=CEDLRaw↩ StringVect(), bool isNChannel=false)

  Create a DeviceN Color Space. Throws an IEDLError on failure.
- static EDL_API IDOMColorSpaceDeviceNPtr create (IEDLClassFactory *pFactory, const CColorant↩ InfoVect &colorants, const IDOMColorSpacePtr &alternate, const IDOMColorSpacePtr &processColor↩ Space=IDOMColorSpacePtr())

  Create a simple DeviceN Color Space, with automatic generation of the tint transform function. Throws an IEDLError on failure.
- static const CClassID & classID ()

  Retrieves class id of IDOMColorSpaceDeviceN.

Additional Inherited Members

Public Types inherited from IDOMColorSpace

- enum eColorSpaceType {
  eDeviceRGB , eDeviceGray , eDeviceCMYK , esRGB ,
  esGray , escRGB , eICCBased , eIndexed ,
  eDeviceN , eLAB , eDeviceCMY , **eNumColorSpaceTypes** = 11 }

  Color spaces type enumeration.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.171.1 Detailed Description

This color space is analogous to the PostScript/PDF DeviceN/Separation color spaces.

```
See section 4.5.5 of the PDF Reference, version 1.7
```

7.171.2 Member Function Documentation

classID()

```
static const CClassID & IDOMColorSpaceDeviceN::classID ( )  [inline], [static]
```

Retrieves class id of IDOMColorSpaceDeviceN.

Returns

CClassID Class id of the element

create() [1/3]

```
static EDL_API IDOMColorSpaceDeviceNPtr IDOMColorSpaceDeviceN::create (
            IEDLClassFactory * pFactory,
            const CColorantInfoVect & colorants,
            const IDOMColorSpacePtr & alternate,
            const IDOMColorSpacePtr & processColorSpace = IDOMColorSpacePtr() )  [static]
```

Create a simple DeviceN Color Space, with automatic generation of the tint transform function. Throws an IEDLError on failure.

Parameters

pFactory	The EDL Class factory to use.
colorants	A vector of CColorantInfo classes, specifying the name of each colorant, the color components of that colorant in its alternate color space, and (optionally) an alternate color space for that particular colorant. If the alternate color space is not provided for a given colorant then the components are interpreted using the DeviceN alternate.
alternate	The alternate color space.
processColorSpace	The PDF process color space associated with this DeviceN color space. Optional.

Returns

IDOMColorSpaceDeviceN The new color space.

create() [2/3]

```
static EDL_API IDOMColorSpaceDeviceNPtr IDOMColorSpaceDeviceN::create (
            IEDLClassFactory * pFactory,
            const CDeviceNColorantVect & colorants,
            const IDOMColorSpacePtr & alternate,
            const IDOMFunctionPtr & tintTransform,
            const IDOMColorSpacePtr & processColorSpace = IDOMColorSpacePtr(),
            const CEDLRawStringVect & processComponents = CEDLRawStringVect(),
            const CEDLRawStringVect & printingOrder = CEDLRawStringVect(),
            bool isNChannel = false )  [static]
```

Create a DeviceN Color Space. Throws an IEDLError on failure.

Parameters

pFactory	The EDL Class factory to use.
colorants	A vector of smart pointers to IDOMDeviceNColorant objects
alternate	The alternate color space.
tintTransform	A function that maps colors in the space to the alternate color space.
processColorSpace	The PDF process color space associated with this DeviceN color space. Optional.
processComponents	A vector containing a list of process components. Optional
printingOrder	A vector containing a list defining the printing order. Optional
isNChannel	Specifies if the new color space is NChannel. Optional

Returns

IDOMColorSpaceDeviceN The new color space.

create() [3/3]

```
static EDL_API IDOMColorSpaceDeviceNPtr IDOMColorSpaceDeviceN::create (
            IEDLClassFactory * pFactory,
            const CEDLRawStringVect & colorants,
            const IDOMColorSpacePtr & alternate,
            const IDOMFunctionPtr & tintTransform,
            const IDOMColorSpacePtr & processColorSpace = IDOMColorSpacePtr() )  [static]
```

Create a simple DeviceN Color Space. Throws an IEDLError on failure.

Parameters

pFactory	The EDL Class factory to use.
colorants	A vector of colorant names for the device inks
alternate	The alternate color space.
tintTransform	A function that maps colors in the space to the alternate color space.
processColorSpace	The PDF process color space associated with this DeviceN color space. Optional.

Returns

IDOMColorSpaceDeviceN The new color space.

getAlternateColorSpace()

```
virtual IDOMColorSpacePtr IDOMColorSpaceDeviceN::getAlternateColorSpace ( ) const  [pure virtual]
```

Get the alternate color space for this color space, which must be present. An exception of type IEDLError is thrown if it is missing.

Returns

IDOMColorSpacePtr The alternate space.

getColorant()

```
virtual IDOMDeviceNColorantPtr IDOMColorSpaceDeviceN::getColorant (
            uint8 component ) const  [pure virtual]
```

Get the colorant information for a component at the given index. An exception of type IEDLError is thrown on failure.

Parameters

component	The 0 base component index of the desired colorant

Returns

IDOMDeviceNColorantPtr The colorant information.

getColorantInfo()

```
virtual CColorantInfoVect IDOMColorSpaceDeviceN::getColorantInfo (
            IEDLClassFactory * factory,
            const IDOMColorSpacePtr & colorantSpace = IDOMColorSpacePtr() ) const  [pure virtual]
```

Convenience method to populate a CColorantInfoVect from the DeviceN colorspace.

Parameters

factory	The EDL Class factory to use.
colorantSpace	The target color space for the components. If not provided, the alternate color space will be used.

getHasRealColorant()

```
virtual bool IDOMColorSpaceDeviceN::getHasRealColorant ( ) const  [pure virtual]
```

Determine if this DeviceN color space contains only the special "None" colorant (or the strange case where there are multiple colorants all with the name "None"). Such color spaces will never render or mark a page.

Returns

bool Returns true if there is at least one none "None" colorant present, false otherwise.

getIsNChannel()

```
virtual bool IDOMColorSpaceDeviceN::getIsNChannel ( ) const  [pure virtual]
```

Get if the color space represents an NChannel color space.

Returns

bool True if an NChannel color space

getPrintingOrder()

```
virtual const CEDLRawStringVect & IDOMColorSpaceDeviceN::getPrintingOrder ( ) const  [pure
virtual]
```

Get the list of component names describing the printing order of the device N colorants. An exception of IEDLError is thrown on failure, which should not happen.

Returns

CEDLRawStringVect The printing order component names vector.

getProcessColorSpace()

```
virtual IDOMColorSpacePtr IDOMColorSpaceDeviceN::getProcessColorSpace ( ) const  [pure virtual]
```

Get the Process color space associated with this DeviceN space. (See Table 4.22 of the PDF 1.7 Reference). Optional.

Returns

IDOMColorSpacePtr The process color space if present, NULL otherwise.

getProcessComponentNames()

```
virtual const CEDLRawStringVect & IDOMColorSpaceDeviceN::getProcessComponentNames ( ) const
[pure virtual]
```

Get the vector of Process component names to those in the associated process color space (see Table 4.22 of the PDF 1.7 reference). An exception of IEDLError is thrown on failure, which should not happen.

Returns

CEDLRawStringVect The component names vector.

getTintTransform()

```
virtual IDOMFunctionPtr IDOMColorSpaceDeviceN::getTintTransform ( ) const  [pure virtual]
```

Get the function that maps components of this space to component(s) of the alternate color space. Must be present. An exception of type IEDLError is thrown if it is missing.

Returns

IDOMFunctionPtr The function.

The documentation for this class was generated from the following file:

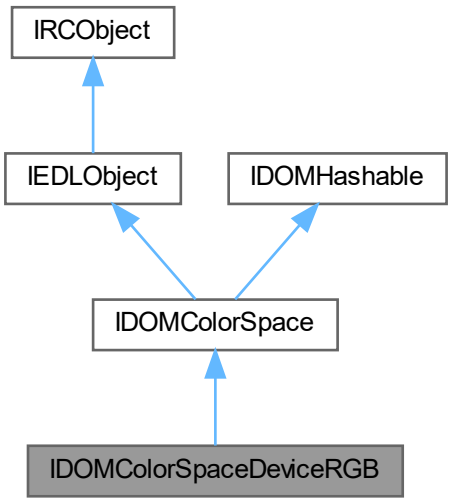
- idomcolorspace.h

7.172 IDOMColorSpaceDeviceRGB Class Reference

IDOMColorSpaceDeviceRGB interface.

`#include <idomcolorspace.h>`

Inheritance diagram for IDOMColorSpaceDeviceRGB:



Static Public Member Functions

- static EDL_API IDOMColorSpaceDeviceRGBPtr create (IEDLClassFactory *pFactory)

  Create a DeviceRGB Color Space. Throws an IEDLError on failure.
- static const CClassID & classID ()

  Retrieves class id of IDOMColorSpaceDeviceRGB.

Additional Inherited Members

Public Types inherited from IDOMColorSpace

- enum eColorSpaceType {
  eDeviceRGB , eDeviceGray , eDeviceCMYK , esRGB ,
  esGray , escRGB , eICCBased , eIndexed ,
  eDeviceN , eLAB , eDeviceCMY , **eNumColorSpaceTypes** = 11 }

  Color spaces type enumeration.

Public Member Functions inherited from IDOMColorSpace

- virtual eColorSpaceType getColorSpaceType ()=0

  Retrieves the color space type.
- virtual uint8 getNumComponents ()=0

  Retrieves the number of components that are in colors in this color space.
- virtual bool getComponentsHaveSameRange ()=0

  Checks if this color space has the same range for all its components. Some color spaces (such as Lab and ICC) may not have the same range for all its components. If a color space does have the same range for all its components, you only need to call getComponentRange() once to find the range for all components.
- virtual bool getComponentRange (int component, float &low, float &high)=0

  Retrieves the expected range of component values for a given channel, if applicable.
- virtual eRenderingIntent getDefaultRenderingIntent ()=0

  Retrieves the default rendering intent for this color space.
- virtual EDLRawString getColorantName (uint8 component) const =0

  Determine the colorant name for a colorant index. If a colorant name cannot be determined, an IEDLError with type EDL_ERR_NO_COLOR_NAME will be thrown. Cannot be called on Indexed or LAB color spaces. For device spaces and simple ICC color spaces, the basic process color name will be returned (ie one of Gray, Red, Green, Blue, Cyan, Magenta, Yellow, Black as appropriate).
- virtual bool equals (const IDOMColorSpacePtr &colorSpace, bool exact=false)=0

  Determines if the given color space is equivalent to this color space.
- virtual bool similar (const IDOMColorSpacePtr &colorSpace, eRenderingIntent intent, eBlackPointCompensation bpc)=0

  Determines if the given color space is very similar to this color space for the given rendering intent. Color spaces are tested via selective sampling and are considered similar if the samples show the same results to a 10 bit accuracy. This can be an expensive operation for complicated color spaces.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()
  Virtual destructor.

7.172.1 Detailed Description

IDOMColorSpaceDeviceRGB interface.

7.172.2 Member Function Documentation

classID()

```
static const CClassID & IDOMColorSpaceDeviceRGB::classID ( )  [inline], [static]
```

Retrieves class id of IDOMColorSpaceDeviceRGB.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMColorSpaceDeviceRGBPtr IDOMColorSpaceDeviceRGB::create (
            IEDLClassFactory * pFactory )  [static]
```

Create a DeviceRGB Color Space. Throws an IEDLError on failure.

Parameters

pFactory	The EDL Class factory to use.

Returns

IDOMColorSpaceDeviceRGB The new color space.

The documentation for this class was generated from the following file:

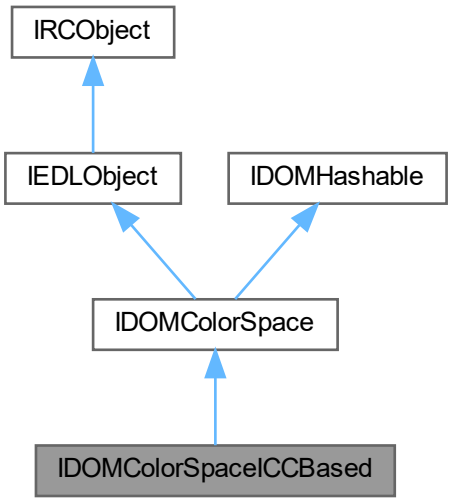
- idomcolorspace.h

7.173 IDOMColorSpaceICCBased Class Reference

Represents a color space described by an ICC profile.

```
#include <idomcolorspace.h>
```

Inheritance diagram for IDOMColorSpaceICCBased:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual IDOMICCProfilePtr getICCProfile ()=0

  Retrieve the ICC profile for this color space. As a profile is required and should have been available at initialisation time, an exception of type IEDLError is thrown on failure.
- virtual void setICCProfile (const IDOMICCProfilePtr &profile)=0

  Set the ICC profile for the color space.
- virtual IDOMColorSpacePtr getAlternateColorSpace ()=0

  Gets the PDF alternate color space for this ICC-based color space. Useful for PDF input filter context.
- virtual bool getIsLabColorSpace ()=0

  Determine if this ICC space represents a LAB color space.

Public Member Functions inherited from IDOMColorSpace

- virtual eColorSpaceType getColorSpaceType ()=0

  Retrieves the color space type.
- virtual uint8 getNumComponents ()=0

  Retrieves the number of components that are in colors in this color space.
- virtual bool getComponentsHaveSameRange ()=0

  Checks if this color space has the same range for all its components. Some color spaces (such as Lab and ICC) may not have the same range for all its components. If a color space does have the same range for all its components, you only need to call getComponentRange() once to find the range for all components.

- virtual bool getComponentRange (int component, float &low, float &high)=0

  Retrieves the expected range of component values for a given channel, if applicable.
- virtual eRenderingIntent getDefaultRenderingIntent ()=0

  Retrieves the default rendering intent for this color space.
- virtual EDLRawString getColorantName (uint8 component) const =0

  Determine the colorant name for a colorant index. If a colorant name cannot be determined, an IEDLError with type EDL_ERR_NO_COLOR_NAME will be thrown. Cannot be called on Indexed or LAB color spaces. For device spaces and simple ICC color spaces, the basic process color name will be returned (ie one of Gray, Red, Green, Blue, Cyan, Magenta, Yellow, Black as appropriate).
- virtual bool equals (const IDOMColorSpacePtr &colorSpace, bool exact=false)=0

  Determines if the given color space is equivalent to this color space.
- virtual bool similar (const IDOMColorSpacePtr &colorSpace, eRenderingIntent intent, eBlackPointCompensation bpc)=0

  Determines if the given color space is very similar to this color space for the given rendering intent. Color spaces are tested via selective sampling and are considered similar if the samples show the same results to a 10 bit accuracy. This can be an expensive operation for complicated color spaces.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static EDL_API IDOMColorSpaceICCBasedPtr create (IEDLClassFactory *pFactory, const IDOMICCProfile↩ Ptr &profile, const IDOMColorSpacePtr &alternate=IDOMColorSpacePtr())

  Creation function for an IDOMColorspaceICCBased color space. Throws an IEDLError exception on failure.
- static const CClassID & classID ()

  Retrieves class id of IDOMColorSpaceICCBased.

Additional Inherited Members

Public Types inherited from IDOMColorSpace

- enum eColorSpaceType {
  eDeviceRGB , eDeviceGray , eDeviceCMYK , esRGB ,
  esGray , escRGB , eICCBased , eIndexed ,
  eDeviceN , eLAB , eDeviceCMY , **eNumColorSpaceTypes** = 11 }
  Color spaces type enumeration.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()
  Virtual destructor.

7.173.1 Detailed Description

Represents a color space described by an ICC profile.

7.173.2 Member Function Documentation

classID()

```
static const CClassID & IDOMColorSpaceICCBased::classID ( )  [inline], [static]
```

Retrieves class id of IDOMColorSpaceICCBased.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMColorSpaceICCBasedPtr IDOMColorSpaceICCBased::create (
            IEDLClassFactory * pFactory,
            const IDOMICCProfilePtr & profile,
            const IDOMColorSpacePtr & alternate = IDOMColorSpacePtr() )  [static]
```

Creation function for an IDOMColorspaceICCBased color space. Throws an IEDLError exception on failure.

Parameters

pFactory	The class factory.
profile	The profile to use
alternate	The alternate color space, if any. If not supplied and the profile has 1, 3, or 4 components, a Device colour space alternate will be created.

Returns

IDOMColorSpaceICCBasedPtr The new color space.

getAlternateColorSpace()

```
virtual IDOMColorSpacePtr IDOMColorSpaceICCBased::getAlternateColorSpace ( )  [pure virtual]
```

Gets the PDF alternate color space for this ICC-based color space. Useful for PDF input filter context.

Returns

IDOMColorSpacePtr The alternate space, or NULL if no present.

getICCProfile()

```
virtual IDOMICCProfilePtr IDOMColorSpaceICCBased::getICCProfile ( )  [pure virtual]
```

Retrieve the ICC profile for this color space. As a profile is required and should have been available at initialisation time, an exception of type IEDLError is thrown on failure.

Returns

IDOMICCProfilePtr The profile.

getIsLabColorSpace()

```
virtual bool IDOMColorSpaceICCBased::getIsLabColorSpace ( )  [pure virtual]
```

Determine if this ICC space represents a LAB color space.

Returns

bool true if the ICC profile represents a LAB color space, false otherwise.

setICCProfile()

```
virtual void IDOMColorSpaceICCBased::setICCProfile (
            const IDOMICCProfilePtr & profile )  [pure virtual]
```

Set the ICC profile for the color space.

Parameters

profile	A reference to the new ICC profile.

The documentation for this class was generated from the following file:

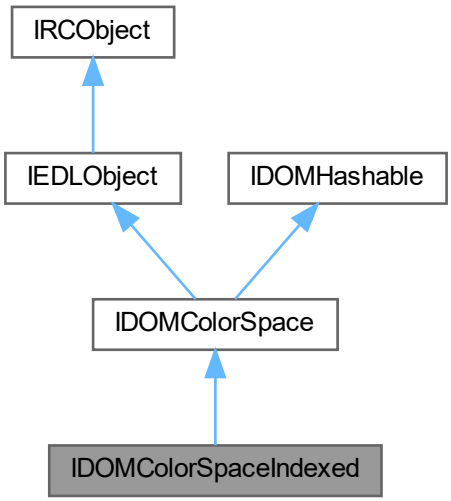
- idomcolorspace.h

7.174 IDOMColorSpaceIndexed Class Reference

This color space is analogous to the PostScript/PDF Indexed color space.

`#include <idomcolorspace.h>`

Inheritance diagram for IDOMColorSpaceIndexed:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual IDOMColorSpacePtr getUnderlyingColorSpace ()=0

  Get the underlying color space for this color space, which must be present. Throws an exception of type IEDLError on failure.
- virtual IDOMFunctionPtr getMappingFunction ()=0

  Get the function that maps an index value to component(s) in the underlying color space. Must always be present. An exception of type IEDLError is thrown if it is missing.

Public Member Functions inherited from IDOMColorSpace

- virtual eColorSpaceType getColorSpaceType ()=0

  Retrieves the color space type.
- virtual uint8 getNumComponents ()=0

  Retrieves the number of components that are in colors in this color space.
- virtual bool getComponentsHaveSameRange ()=0

  Checks if this color space has the same range for all its components. Some color spaces (such as Lab and ICC) may not have the same range for all its components. If a color space does have the same range for all its components, you only need to call getComponentRange() once to find the range for all components.
- virtual bool getComponentRange (int component, float &low, float &high)=0

  Retrieves the expected range of component values for a given channel, if applicable.
- virtual eRenderingIntent getDefaultRenderingIntent ()=0

  Retrieves the default rendering intent for this color space.
- virtual EDLRawString getColorantName (uint8 component) const =0

  Determine the colorant name for a colorant index. If a colorant name cannot be determined, an IEDLError with type EDL_ERR_NO_COLOR_NAME will be thrown. Cannot be called on Indexed or LAB color spaces. For device spaces and simple ICC color spaces, the basic process color name will be returned (ie one of Gray, Red, Green, Blue, Cyan, Magenta, Yellow, Black as appropriate).
- virtual bool equals (const IDOMColorSpacePtr &colorSpace, bool exact=false)=0

  Determines if the given color space is equivalent to this color space.
- virtual bool similar (const IDOMColorSpacePtr &colorSpace, eRenderingIntent intent, eBlackPointCompensation bpc)=0

  Determines if the given color space is very similar to this color space for the given rendering intent. Color spaces are tested via selective sampling and are considered similar if the samples show the same results to a 10 bit accuracy. This can be an expensive operation for complicated color spaces.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static EDL_API IDOMColorSpaceIndexedPtr create (IEDLClassFactory *pFactory, const IDOMColorSpace↩ Ptr &underlying, const IDOMFunctionPtr &mappingFunction)

  Create an Indexed Color Space. Throws an IEDLError on failure.
- static EDL_API IDOMColorSpaceIndexedPtr create (IEDLClassFactory *pFactory, const IDOMColorSpace↩ Ptr &underlying, const CEDLVector< uint8 > &table)

  Create an Indexed Color Space from an 8bpc table. Throws an IEDLError on failure.
- static const CClassID & classID ()

  Retrieves class id of IDOMColorSpaceICCBased.

Additional Inherited Members

Public Types inherited from IDOMColorSpace

- enum eColorSpaceType {
  eDeviceRGB , eDeviceGray , eDeviceCMYK , esRGB ,
  esGray , escRGB , eICCBased , eIndexed ,
  eDeviceN , eLAB , eDeviceCMY , **eNumColorSpaceTypes** = 11 }

  Color spaces type enumeration.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.174.1 Detailed Description

This color space is analogous to the PostScript/PDF Indexed color space.

Instances of this type represent a mapping from an integral component value to components of an underlying color space using a Function.

- Indexed color spaces cannot be used as the underlying space.
- See section 4.8.4 of the PostScript Language Reference, 3rd Edition, and section 4.5.5 of the PDF Reference, version 1.7

7.174.2 Member Function Documentation

classID()

```
static const CClassID & IDOMColorSpaceIndexed::classID ( )  [inline], [static]
```

Retrieves class id of IDOMColorSpaceICCBased.

Returns

CClassID Class id of the element

create() [1/2]

```
static EDL_API IDOMColorSpaceIndexedPtr IDOMColorSpaceIndexed::create (
            IEDLClassFactory * pFactory,
            const IDOMColorSpacePtr & underlying,
            const CEDLVector< uint8 > & table )  [static]
```

Create an Indexed Color Space from an 8bpc table. Throws an IEDLError on failure.

Parameters

pFactory	The EDL Class factory to use.
underlying	The underlying color space.
table	An 8-bit lookup table consisting of a color values for each index. For example, for an RGB underlying color space, the values are packed as [<R for index 0>, <G for index 0>, <B for index 0>, <R for index 1>, <G for index 1> ...]

Returns

IDOMColorSpaceIndexedPtr The new color space.

create() [2/2]

```
static EDL_API IDOMColorSpaceIndexedPtr IDOMColorSpaceIndexed::create (
            IEDLClassFactory * pFactory,
            const IDOMColorSpacePtr & underlying,
            const IDOMFunctionPtr & mappingFunction )  [static]
```

Create an Indexed Color Space. Throws an IEDLError on failure.

Parameters

pFactory	The EDL Class factory to use.
underlying	The underlying color space.
mappingFunction	A function that maps indexes to colors in the underlying color space.

Returns

IDOMColorSpaceIndexedPtr The new color space.

getMappingFunction()

```
virtual IDOMFunctionPtr IDOMColorSpaceIndexed::getMappingFunction ( )  [pure virtual]
```

Get the function that maps an index value to component(s) in the underlying color space. Must always be present. An exception of type IEDLError is thrown if it is missing.

Returns

IDOMFunctionPtr The mapping function.

getUnderlyingColorSpace()

```
virtual IDOMColorSpacePtr IDOMColorSpaceIndexed::getUnderlyingColorSpace ( )  [pure virtual]
```

Get the underlying color space for this color space, which must be present. Throws an exception of type IEDLError on failure.

Returns

IDOMColorSpacePtr The underlying color space.

The documentation for this class was generated from the following file:

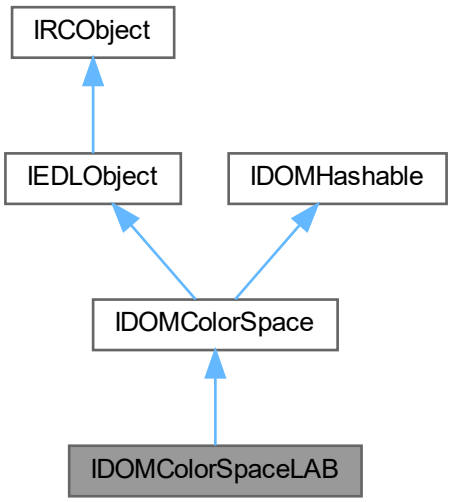
- idomcolorspace.h

7.175 IDOMColorSpaceLAB Class Reference

This color space is as described in section 4.5.4 of the PDF 1.7 Reference Manual.

`#include <idomcolorspace.h>`

Inheritance diagram for IDOMColorSpaceLAB:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual void getRangeAB (float &lowA, float &highA, float &lowB, float &highB)=0

  Get the input range for this color space. The range of the L component is always implicitly 0 to 100.
- virtual void getWhitePoint (float &x, float &y, float &z)=0

  Get the White Point in CIE 1931 XYZ space.
- virtual void getBlackPoint (float &x, float &y, float &z)=0

  Get the Black Point in CIE 1931 XYZ space.

Public Member Functions inherited from IDOMColorSpace

- virtual eColorSpaceType getColorSpaceType ()=0

  Retrieves the color space type.
- virtual uint8 getNumComponents ()=0

  Retrieves the number of components that are in colors in this color space.
- virtual bool getComponentsHaveSameRange ()=0

  Checks if this color space has the same range for all its components. Some color spaces (such as Lab and ICC) may not have the same range for all its components. If a color space does have the same range for all its components, you only need to call getComponentRange() once to find the range for all components.
- virtual bool getComponentRange (int component, float &low, float &high)=0

  Retrieves the expected range of component values for a given channel, if applicable.
- virtual eRenderingIntent getDefaultRenderingIntent ()=0

  Retrieves the default rendering intent for this color space.
- virtual EDLRawString getColorantName (uint8 component) const =0

  Determine the colorant name for a colorant index. If a colorant name cannot be determined, an IEDLError with type EDL_ERR_NO_COLOR_NAME will be thrown. Cannot be called on Indexed or LAB color spaces. For device spaces and simple ICC color spaces, the basic process color name will be returned (ie one of Gray, Red, Green, Blue, Cyan, Magenta, Yellow, Black as appropriate).
- virtual bool equals (const IDOMColorSpacePtr &colorSpace, bool exact=false)=0

  Determines if the given color space is equivalent to this color space.
- virtual bool similar (const IDOMColorSpacePtr &colorSpace, eRenderingIntent intent, eBlackPointCompensation bpc)=0

  Determines if the given color space is very similar to this color space for the given rendering intent. Color spaces are tested via selective sampling and are considered similar if the samples show the same results to a 10 bit accuracy. This can be an expensive operation for complicated color spaces.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static EDL_API IDOMColorSpaceLABPtr create (IEDLClassFactory *pFactory, float whitePointX, float whitePointY, float whitePointZ, float blackPointX=0.0f, float blackPointY=0.0f, float blackPointZ=0.0f, float rangeALo=-100.0f, float rangeAHi=100.0f, float rangeBLo=-100.0f, float rangeBHi=100.0f)

  Create a LAB Color Space. Throws an IEDLError on failure.
- static const CClassID & classID ()

  Retrieves class id of IDOMColorSpaceLAB.

Additional Inherited Members

Public Types inherited from IDOMColorSpace

- enum eColorSpaceType {
  eDeviceRGB , eDeviceGray , eDeviceCMYK , esRGB ,
  esGray , escRGB , eICCBased , eIndexed ,
  eDeviceN , eLAB , eDeviceCMY , **eNumColorSpaceTypes** = 11 }

  Color spaces type enumeration.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.175.1 Detailed Description

This color space is as described in section 4.5.4 of the PDF 1.7 Reference Manual.

7.175.2 Member Function Documentation

classID()

```
static const CClassID & IDOMColorSpaceLAB::classID ( )  [inline], [static]
```

Retrieves class id of IDOMColorSpaceLAB.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMColorSpaceLABPtr IDOMColorSpaceLAB::create (
            IEDLClassFactory * pFactory,
            float whitePointX,
            float whitePointY,
            float whitePointZ,
            float blackPointX = 0.0f,
            float blackPointY = 0.0f,
            float blackPointZ = 0.0f,
            float rangeALo = -100.0f,
            float rangeAHi = 100.0f,
            float rangeBLo = -100.0f,
            float rangeBHi = 100.0f )  [static]
```

Create a LAB Color Space. Throws an IEDLError on failure.

Parameters

pFactory	The EDL Class factory to use.
whitePointX	The CIE X value for the white point.
whitePointY	The CIE Y value for the white point.
whitePointZ	The CIE V value for the white point.
blackPointX	The CIE X value for the black point.
blackPointY	The CIE Y value for the black point.
blackPointZ	The CIE V value for the black point.
rangeALo	The low bound of the A parameter
rangeAHi	The high bound of the A parameter
rangeBLo	The low bound of the B parameter
rangeBHi	The high bound of the B parameter

Returns

IDOMColorSpaceLABPtr The new color space.

getBlackPoint()

```
virtual void IDOMColorSpaceLAB::getBlackPoint (
             float & x,
             float & y,
             float & z )  [pure virtual]
```

Get the Black Point in CIE 1931 XYZ space.

Parameters

x	Reference to receive the black point X value
y	Reference to receive the black point Y value
z	Reference to receive the black point Z value

getRangeAB()

```
virtual void IDOMColorSpaceLAB::getRangeAB (
             float & lowA,
             float & highA,
             float & lowB,
             float & highB )  [pure virtual]
```

Get the input range for this color space. The range of the L component is always implicitly 0 to 100.

Parameters

lowA	Reference to receive the low bound of the A parameter
highA	Reference to receive the high bound of the A parameter
lowB	Reference to receive the low bound of the B parameter
highB	Reference to receive the high bound of the B parameter

getWhitePoint()

```
virtual void IDOMColorSpaceLAB::getWhitePoint (
             float & x,
             float & y,
             float & z )  [pure virtual]
```

Get the White Point in CIE 1931 XYZ space.

Parameters

x	Reference to receive the white point X value
y	Reference to receive the white point Y value
z	Reference to receive the white point Z value

The documentation for this class was generated from the following file:

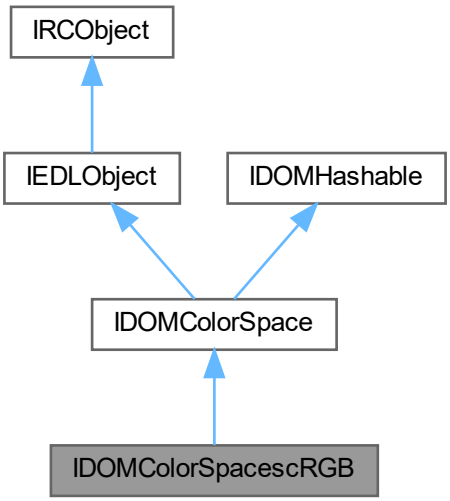
- idomcolorspace.h

7.176 IDOMColorSpacescRGB Class Reference

Represents the scRGB color space.

`#include <idomcolorspace.h>`

Inheritance diagram for IDOMColorSpacescRGB:



Static Public Member Functions

- static EDL_API IDOMColorSpacescRGBPtr create (IEDLClassFactory *pFactory)

  Create an scRGB Color Space. Throws an IEDLError on failure.
- static const CClassID & classID ()

  Retrieves the class id of IDOMColorSpacescRGB.

Additional Inherited Members

Public Types inherited from IDOMColorSpace

- enum eColorSpaceType {
  eDeviceRGB , eDeviceGray , eDeviceCMYK , esRGB ,
  esGray , escRGB , eICCBased , eIndexed ,
  eDeviceN , eLAB , eDeviceCMY , **eNumColorSpaceTypes** = 11 }

  Color spaces type enumeration.

Public Member Functions inherited from IDOMColorSpace

- virtual eColorSpaceType getColorSpaceType ()=0

  Retrieves the color space type.
- virtual uint8 getNumComponents ()=0

  Retrieves the number of components that are in colors in this color space.
- virtual bool getComponentsHaveSameRange ()=0

  Checks if this color space has the same range for all its components. Some color spaces (such as Lab and ICC) may not have the same range for all its components. If a color space does have the same range for all its components, you only need to call getComponentRange() once to find the range for all components.
- virtual bool getComponentRange (int component, float &low, float &high)=0

  Retrieves the expected range of component values for a given channel, if applicable.
- virtual eRenderingIntent getDefaultRenderingIntent ()=0

  Retrieves the default rendering intent for this color space.
- virtual EDLRawString getColorantName (uint8 component) const =0

  Determine the colorant name for a colorant index. If a colorant name cannot be determined, an IEDLError with type EDL_ERR_NO_COLOR_NAME will be thrown. Cannot be called on Indexed or LAB color spaces. For device spaces and simple ICC color spaces, the basic process color name will be returned (ie one of Gray, Red, Green, Blue, Cyan, Magenta, Yellow, Black as appropriate).
- virtual bool equals (const IDOMColorSpacePtr &colorSpace, bool exact=false)=0

  Determines if the given color space is equivalent to this color space.
- virtual bool similar (const IDOMColorSpacePtr &colorSpace, eRenderingIntent intent, eBlackPointCompensation bpc)=0

  Determines if the given color space is very similar to this color space for the given rendering intent. Color spaces are tested via selective sampling and are considered similar if the samples show the same results to a 10 bit accuracy. This can be an expensive operation for complicated color spaces.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.176.1 Detailed Description

Represents the scRGB color space.

The scRGB color space allows you to specify a color using the full scRGB color space, which is much larger than the sRGB color space and can represent the entire range of colors perceivable by the human eye.

7.176.2 Member Function Documentation

classID()

```
static const CClassID & IDOMColorSpacescRGB::classID ( )  [inline], [static]
```

Retrieves the class id of IDOMColorSpacescRGB.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMColorSpacescRGBPtr IDOMColorSpacescRGB::create (
            IEDLClassFactory * pFactory )  [static]
```

Create an scRGB Color Space. Throws an IEDLError on failure.

Parameters

pFactory	The EDL Class factory to use

Returns

IDOMColorSpacescRGBPtr The new color space

The documentation for this class was generated from the following file:

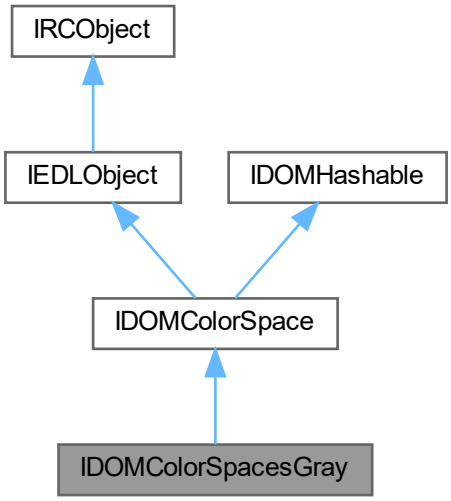
- idomcolorspace.h

7.177 IDOMColorSpacesGray Class Reference

Represents a gray color space using the sRGB gamma and WhitePoint.

`#include <idomcolorspace.h>`

Inheritance diagram for IDOMColorSpacesGray:



Static Public Member Functions

- static EDL_API IDOMColorSpacesGrayPtr create (IEDLClassFactory *pFactory)

  Create an sGray Color Space. Throws an IEDLError on failure.
- static const CClassID & classID ()

  Retrieves class id of IDOMColorSpacesGray.

Additional Inherited Members

Public Types inherited from IDOMColorSpace

- enum eColorSpaceType {
  eDeviceRGB , eDeviceGray , eDeviceCMYK , esRGB ,
  esGray , escRGB , eICCBased , eIndexed ,
  eDeviceN , eLAB , eDeviceCMY , **eNumColorSpaceTypes** = 11 }

  Color spaces type enumeration.

Public Member Functions inherited from IDOMColorSpace

- virtual eColorSpaceType getColorSpaceType ()=0

  Retrieves the color space type.
- virtual uint8 getNumComponents ()=0

  Retrieves the number of components that are in colors in this color space.
- virtual bool getComponentsHaveSameRange ()=0

  Checks if this color space has the same range for all its components. Some color spaces (such as Lab and ICC) may not have the same range for all its components. If a color space does have the same range for all its components, you only need to call getComponentRange() once to find the range for all components.
- virtual bool getComponentRange (int component, float &low, float &high)=0

  Retrieves the expected range of component values for a given channel, if applicable.
- virtual eRenderingIntent getDefaultRenderingIntent ()=0

  Retrieves the default rendering intent for this color space.
- virtual EDLRawString getColorantName (uint8 component) const =0

  Determine the colorant name for a colorant index. If a colorant name cannot be determined, an IEDLError with type EDL_ERR_NO_COLOR_NAME will be thrown. Cannot be called on Indexed or LAB color spaces. For device spaces and simple ICC color spaces, the basic process color name will be returned (ie one of Gray, Red, Green, Blue, Cyan, Magenta, Yellow, Black as appropriate).
- virtual bool equals (const IDOMColorSpacePtr &colorSpace, bool exact=false)=0

  Determines if the given color space is equivalent to this color space.
- virtual bool similar (const IDOMColorSpacePtr &colorSpace, eRenderingIntent intent, eBlackPointCompensation bpc)=0

  Determines if the given color space is very similar to this color space for the given rendering intent. Color spaces are tested via selective sampling and are considered similar if the samples show the same results to a 10 bit accuracy. This can be an expensive operation for complicated color spaces.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.177.1 Detailed Description

Represents a gray color space using the sRGB gamma and WhitePoint.

Gray colors for vector graphics can be specified as sRGB or scRGB colors with the red, green and blue components set to the same value. Gray colors for raster images can be specified without using any image format.

7.177.2 Member Function Documentation

classID()

```
static const CClassID & IDOMColorSpacesGray::classID ( )  [inline], [static]
```

Retrieves class id of IDOMColorSpacesGray.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMColorSpacesGrayPtr IDOMColorSpacesGray::create (
            IEDLClassFactory * pFactory )  [static]
```

Create an sGray Color Space. Throws an IEDLError on failure.

Parameters

pFactory	The EDL Class factory to use.

Returns

IDOMColorSpacesGrayPtr The new color space.

The documentation for this class was generated from the following file:

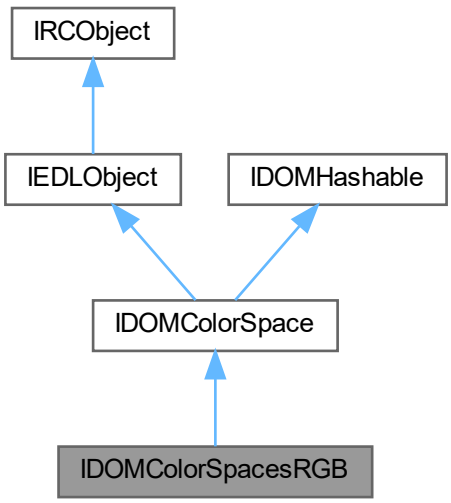
- idomcolorspace.h

7.178 IDOMColorSpacesRGB Class Reference

Represents the RGB color space.

`#include <idomcolorspace.h>`

Inheritance diagram for IDOMColorSpacesRGB:



Static Public Member Functions

- static EDL_API IDOMColorSpacesRGBPtr create (IEDLClassFactory *pFactory)

  Create an sRGB Color Space. Throws an IEDLError on failure.
- static const CClassID & classID ()

  Retrieves the class id of IDOMColorSpacesRGB.

Additional Inherited Members

Public Types inherited from IDOMColorSpace

- enum eColorSpaceType {
  eDeviceRGB , eDeviceGray , eDeviceCMYK , esRGB ,
  esGray , escRGB , eICCBased , eIndexed ,
  eDeviceN , eLAB , eDeviceCMY , **eNumColorSpaceTypes** = 11 }

  Color spaces type enumeration.

Public Member Functions inherited from IDOMColorSpace

- virtual eColorSpaceType getColorSpaceType ()=0

  Retrieves the color space type.
- virtual uint8 getNumComponents ()=0

  Retrieves the number of components that are in colors in this color space.
- virtual bool getComponentsHaveSameRange ()=0

  Checks if this color space has the same range for all its components. Some color spaces (such as Lab and ICC) may not have the same range for all its components. If a color space does have the same range for all its components, you only need to call getComponentRange() once to find the range for all components.
- virtual bool getComponentRange (int component, float &low, float &high)=0

  Retrieves the expected range of component values for a given channel, if applicable.
- virtual eRenderingIntent getDefaultRenderingIntent ()=0

  Retrieves the default rendering intent for this color space.
- virtual EDLRawString getColorantName (uint8 component) const =0

  Determine the colorant name for a colorant index. If a colorant name cannot be determined, an IEDLError with type EDL_ERR_NO_COLOR_NAME will be thrown. Cannot be called on Indexed or LAB color spaces. For device spaces and simple ICC color spaces, the basic process color name will be returned (ie one of Gray, Red, Green, Blue, Cyan, Magenta, Yellow, Black as appropriate).
- virtual bool equals (const IDOMColorSpacePtr &colorSpace, bool exact=false)=0

  Determines if the given color space is equivalent to this color space.
- virtual bool similar (const IDOMColorSpacePtr &colorSpace, eRenderingIntent intent, eBlackPointCompensation bpc)=0

  Determines if the given color space is very similar to this color space for the given rendering intent. Color spaces are tested via selective sampling and are considered similar if the samples show the same results to a 10 bit accuracy. This can be an expensive operation for complicated color spaces.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.178.1 Detailed Description

Represents the RGB color space.

7.178.2 Member Function Documentation

classID()

```
static const CClassID & IDOMColorSpacesRGB::classID ( )  [inline], [static]
```

Retrieves the class id of IDOMColorSpacesRGB.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMColorSpacesRGBPtr IDOMColorSpacesRGB::create (
            IEDLClassFactory * pFactory )  [static]
```

Create an sRGB Color Space. Throws an IEDLError on failure.

Parameters

pFactory	The EDL Class factory to use.

Returns

IDOMColorSpacesRGBPtr The new color space.

The documentation for this class was generated from the following file:

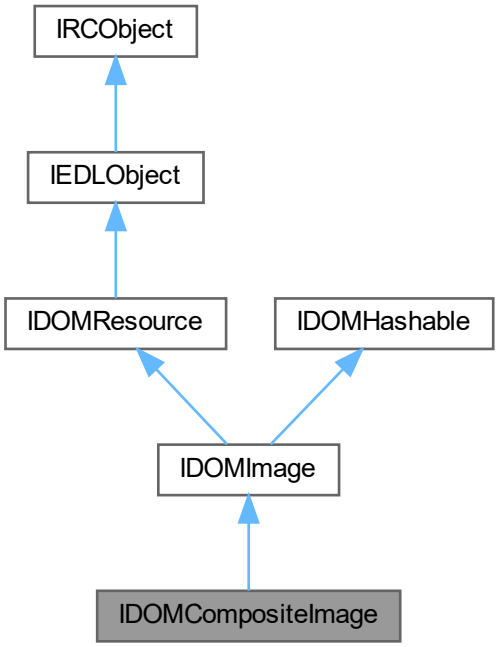
- idomcolorspace.h

7.179 IDOMCompositeImage Class Reference

Interface to a class representing a image made up of separate images joined together vertically, appearing as a single image. All images must use the same color space, depth, width, and the same number of channels.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMCompositeImage:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual bool getStream (IInputStreamPtr &stream) const

  This image type does not allow direct access to the underlying streams.
- virtual void setStream (const IInputStreamPtr &stream)

  This image type does not allow direct access to the underlying streams.

Public Member Functions inherited from IDOMImage

- virtual IImageDecoderPtr createImageDecoder (IEDLClassFactory *factory, const IDOMImagePropertiesPtr &imageProperties)=0

  Creates a properly initialized image decoder object that reads from an inputstream that is specific to that image format.
- virtual IImageFramePtr getImageFrame (IEDLClassFactory *factory)

  Fetch the image frame; convenience.
- virtual IImageEncoderPtr createImageEncoder (const ISessionPtr &session, const IOutputStreamPtr &imageDest, const IDOMImagePropertiesPtr &imageProperties)=0

  Creates a properly initialized image encoder object that writes to an outputstream that is specific to that image format.
- virtual IDOMImagePropertiesPtr getImageProperties ()=0

  Returns an object that stores the properties for this image object. The properties can then be inspected (or more added) by clients that need to manipulate the image resource.
- virtual eDOMImageType getImageType ()=0

  Retrieves the image type.
- virtual bool getIsRendered ()=0

  Determine if the image is as a result of rendering. This is indicated if the image type is eDITRendered or if the image explicitly notes this is the case (such as for IDOMPDFImage).
- virtual IDOMImagePtr getImageWithSubstitutedColorSpace (IEDLClassFactory *factory, const IDOMColor↩SpacePtr &colorSpace)

  Obtain an image that is the same as this image, but with the colorspace substituted for another.

Public Member Functions inherited from IDOMResource

- virtual IInputStreamPtr getStream () const =0

  Retrieves the resource stream.
- virtual uint64 getStreamLength () const =0

  Retrieves the stream length, if it is available.
- virtual const EDLSysString & getUri () const =0

  Retrieves the resource URI.
- virtual void setUri (const EDLSysString &uri)=0

  Sets the resource URI.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static EDL_API IDOMCompositeImagePtr create (IEDLClassFactory *pFactory, const CEDLVector< IDOMImagePtr > componentImages)

  Simplified creator for a IDOMCompositeImage.
- static const CClassID & classID ()

  Retrieves class id of IDOMRecombineImage.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.179.1 Detailed Description

Interface to a class representing a image made up of separate images joined together vertically, appearing as a single image. All images must use the same color space, depth, width, and the same number of channels.

7.179.2 Member Function Documentation

classID()

```
static const CClassID & IDOMCompositeImage::classID ( )  [inline], [static]
```

Retrieves class id of IDOMRecombineImage.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMCompositeImagePtr IDOMCompositeImage::create (
            IEDLClassFactory * pFactory,
            const CEDLVector< IDOMImagePtr > componentImages )  [static]
```

Simplified creator for a IDOMCompositeImage.

Parameters

pFactory	The EDL Class Factory
componentImages	The images to be concatenated

Returns

IDOMCompositeImagePtr The combined result.

getStream()

```
virtual bool IDOMCompositeImage::getStream (
            IInputStreamPtr & stream ) const  [inline], [virtual]
```

This image type does not allow direct access to the underlying streams.

Parameters

stream	A smart pointer to the stream

Returns

bool Always false

setStream()

```
virtual void IDOMCompositeImage::setStream (
            const IInputStreamPtr & stream )  [inline], [virtual]
```

This image type does not allow direct access to the underlying streams.

Parameters

stream	A smart pointer to the stream

Implements IDOMResource.

The documentation for this class was generated from the following file:

- idomimageresource.h

7.180 IDOMDePremultiplyFilter Class Reference

An image filter that presents an image with premultiplied alpha as a plain image with plain alpha. It can be applied to any source image, and will do nothing if not required.

```
#include <idomimageresource.h>
```

7.180.1 Detailed Description

An image filter that presents an image with premultiplied alpha as a plain image with plain alpha. It can be applied to any source image, and will do nothing if not required.

The documentation for this class was generated from the following file:

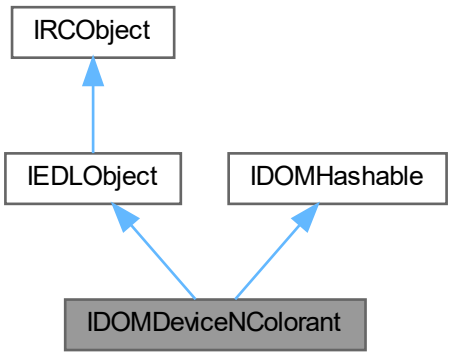
- idomimageresource.h

7.181 IDOMDeviceNColorant Class Reference

This class enables the specification of colorant information for PDF style NChannel variants of DeviceN color spaces.

`#include <idomcolorspace.h>`

Inheritance diagram for IDOMDeviceNColorant:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual const EDLRawString & getName ()=0

  Get the name of the colorant, which must be present. An exception of type IEDLError will be thrown if it is missing.
- virtual bool getSolidity (float &solidity)=0

  Get the solidity of the colorant.
- virtual IDOMFunctionPtr getTintTransform ()=0

  Get the function that maps the colorant to component(s) of the alternate color space. Optional.
- virtual IDOMFunctionPtr getDotGainFunction ()=0

  Get the function that describes the dot gain of the colorant on the intended output device. Optional.
- virtual IDOMColorSpacePtr getAlternateColorSpace ()=0

  Get the alternate color space for this colorant. Optional.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves class id fo IDOMDeviceNColorant.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.181.1 Detailed Description

This class enables the specification of colorant information for PDF style NChannel variants of DeviceN color spaces.

7.181.2 Member Function Documentation

classID()

```
static const CClassID & IDOMDeviceNColorant::classID ( )  [inline], [static]
```

Retrieves class id fo IDOMDeviceNColorant.

Returns

CClassID Class id of the element

getAlternateColorSpace()

```
virtual IDOMColorSpacePtr IDOMDeviceNColorant::getAlternateColorSpace ( )  [pure virtual]
```

Get the alternate color space for this colorant. Optional.

Returns

IDOMColorSpacePtr The alternate space if present, NULL otherwise.

getDotGainFunction()

```
virtual IDOMFunctionPtr IDOMDeviceNColorant::getDotGainFunction ( )  [pure virtual]
```

Get the function that describes the dot gain of the colorant on the intended output device. Optional.

Returns

IDOMFunctionPtr The function if present, NULL otherwise.

getName()

```
virtual const EDLRawString & IDOMDeviceNColorant::getName ( )  [pure virtual]
```

Get the name of the colorant, which must be present. An exception of type IEDLError will be thrown if it is missing.

Returns

EDLRawString The name of the colorant.

getSolidity()

```
virtual bool IDOMDeviceNColorant::getSolidity (
            float & solidity )  [pure virtual]
```

Get the solidity of the colorant.

Parameters

solidity	Reference to receive the solidity

Returns

bool Returns true on success, false if the colorant has no solidity specified.

getTintTransform()

```
virtual IDOMFunctionPtr IDOMDeviceNColorant::getTintTransform ( )  [pure virtual]
```

Get the function that maps the colorant to component(s) of the alternate color space. Optional.

Returns

IDOMFunctionPtr The function if present, NULL otherwise.

The documentation for this class was generated from the following file:

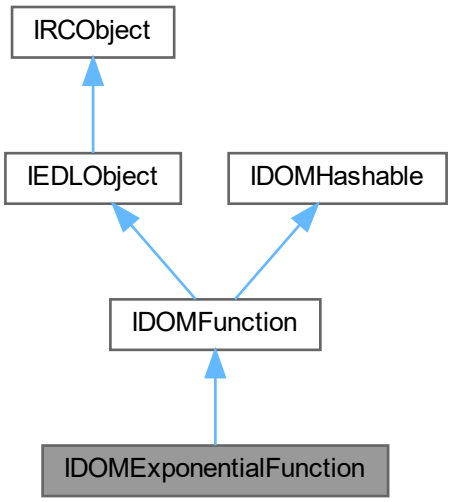
- idomcolorspace.h

7.182 IDOMExponentialFunction Class Reference

Interface for exponential functions. See section 3.9.2 of the PDF 1.7 Reference. Default values are as per described in that reference. There can be only one input for this function type.

```
#include <idomfunction.h>
```

Inheritance diagram for IDOMExponentialFunction:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual float getOutputC0 (uint32 outputNum) const =0

  Get the output C0 value for a given output to the function.
- virtual float getOutputC1 (uint32 outputNum) const =0

  Get the C1 value for a given output to the function.
- virtual float getExponent () const =0

  Get the exponent (N) value for the function.

Public Member Functions inherited from IDOMFunction

- virtual eFunctionType getFunctionType () const =0

  Retrieves function type.
- virtual uint32 getNumInputValues () const =0

  Get the number of input values that this function will operate on.
- virtual uint32 getNumOutputValues () const =0

  Get the number of output values that this function will produce.
- virtual void getInputDomain (uint32 inputNum, float &low, float &high) const =0

  Get the input domain for a given input to the function.
- virtual bool getOutputRange (uint32 outputNum, float &low, float &high) const =0

  Get the output range for a given input to the function.
- virtual void evaluate (const float *inputValues, float *outputValues) const =0

  Evaluate the input through the function and return the result.
- virtual void evaluate (const int32 *inputValues, float *outputValues) const =0

  Evaluate the input through the function and return the result.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()
  Virtual destructor.
- virtual bool hash (uint64 &hash)=0
  Retrieve a hash for this object.
- virtual uint64 hashE ()
  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static const CClassID & classID ()
  Retrieves class id of IDOM.

Additional Inherited Members

Public Types inherited from IDOMFunction

- enum eFunctionType
  An enum for the various types of functions.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()
  Virtual destructor.

7.182.1 Detailed Description

Interface for exponential functions. See section 3.9.2 of the PDF 1.7 Reference. Default values are as per described in that reference. There can be only one input for this function type.

7.182.2 Member Function Documentation

classID()

```
static const CClassID & IDOMExponentialFunction::classID ( )  [inline], [static]
```

Retrieves class id of IDOM.

Returns

CClassID Class id of the element

getExponent()

```
virtual float IDOMExponentialFunction::getExponent ( ) const  [pure virtual]
```

Get the exponent (N) value for the function.

Returns

float The exponent

getOutputC0()

```
virtual float IDOMExponentialFunction::getOutputC0 (
            uint32 outputNum ) const  [pure virtual]
```

Get the output C0 value for a given output to the function.

Parameters

outputNum	The 0-indexed output number

Returns

float The C0 value

getOutputC1()

```
virtual float IDOMExponentialFunction::getOutputC1 (
             uint32 outputNum ) const  [pure virtual]
```

Get the C1 value for a given output to the function.

Parameters

outputNum	The 0-indexed input number

Returns

float The C1 value

The documentation for this class was generated from the following file:

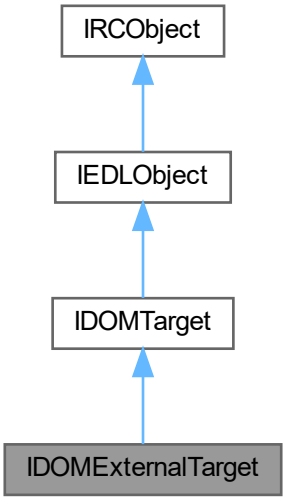
- idomfunction.h

7.183 IDOMExternalTarget Class Reference

IDOMExternalTarget interface.

```
#include <idomtarget.h>
```

Inheritance diagram for IDOMExternalTarget:



Public Member Functions

- virtual EDLSysString getTargetUri () const =0

  Retrieves the target URI.
- virtual void setTargetUri (const EDLSysString &uri)=0

  Sets the target URI.
- virtual bool getIsMap () const =0

  Retrieves the value of the IsMap flag. This is a flag specifying whether to track the mouse position when the URI is resolved.
- virtual void setIsMap (bool isMap)=0

  Sets the value of the IsMap flag.
- virtual eTargetType getTargetType () const

  Retrieves the target URI type.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Public Types inherited from IDOMTarget

- enum eTargetType {
  eExternal , eInternal , ePage , ePageRect ,
  eActionGoToR , eActionGoToE , eActionLaunch , eActionThread ,
  eActionSound , eActionMovie , eActionHide , eActionNamed ,
  eActionSubmitForm , eActionResetForm , eActionImportData , eActionJavaScript ,
  eActionSetOCGState , eActionRendition , eActionTrans , eActionGoTo3DView ,
  eActionArray }

  An enumeration of target types.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.183.1 Detailed Description

IDOMExternalTarget interface.

This interface describes the existence of links to documents and resources that are not part of the present document.

7.183.2 Member Function Documentation

getIsMap()

```
virtual bool IDOMExternalTarget::getIsMap ( ) const  [pure virtual]
```

Retrieves the value of the IsMap flag. This is a flag specifying whether to track the mouse position when the URI is resolved.

Returns

bool. The function returns the target IsMap flag value.

getTargetType()

```
virtual eTargetType IDOMExternalTarget::getTargetType ( ) const  [inline], [virtual]
```

Retrieves the target URI type.

Returns

eTargetType. The target type specified by the eTargetType enumeration.

Implements IDOMTarget.

getTargetUri()

```
virtual EDLSysString IDOMExternalTarget::getTargetUri ( ) const  [pure virtual]
```

Retrieves the target URI.

Returns

EDLSysString. The function retrieves the target URI.

setIsMap()

```
virtual void IDOMExternalTarget::setIsMap (
            bool isMap )  [pure virtual]
```

Sets the value of the IsMap flag.

Parameters

isMap	The new value of IsMap flag.

setTargetUri()

```
virtual void IDOMExternalTarget::setTargetUri (
            const EDLSysString & uri )  [pure virtual]
```

Sets the target URI.

Parameters

uri	The new target URI.

The documentation for this class was generated from the following file:

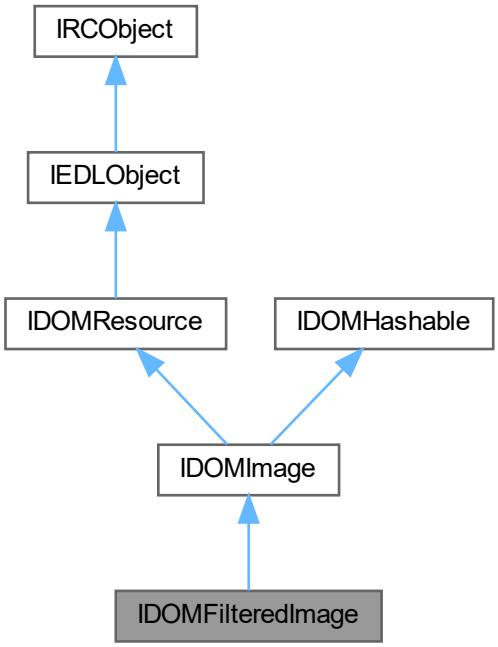
- idomtarget.h

7.184 IDOMFilteredImage Class Reference

IDOMFilteredImage interface. Provides a method for filtering of an underlying image without requiring converted image data to be stored. It maintains a list of filters that are successively applied.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMFilteredImage:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual void pushFilter (const IDOMImageFilterPtr &filter)=0

  Push a filter onto the end of the internal filter chain.
- virtual uint32 getNumFilters () const =0

  Get the number of individual filters in the internal filter chain.
- virtual IDOMImageFilterPtr getFilterAtIndex (uint32 index) const =0

  Get a filter from the filter chain at the given index.
- virtual IDOMImagePtr getSourceImage () const =0

  Get the source image.
- virtual bool getStream (IInputStreamPtr &stream) const

  This image type does not allow direct access to the underlying streams.
- virtual void setStream (const IInputStreamPtr &stream)

  This image type does not allow direct access to the underlying streams.

Public Member Functions inherited from IDOMImage

- virtual IImageDecoderPtr createImageDecoder (IEDLClassFactory *factory, const IDOMImagePropertiesPtr &imageProperties)=0

  Creates a properly initialized image decoder object that reads from an inputstream that is specific to that image format.
- virtual IImageFramePtr getImageFrame (IEDLClassFactory *factory)

  Fetch the image frame; convenience.
- virtual IImageEncoderPtr createImageEncoder (const ISessionPtr &session, const IOutputStreamPtr &imageDest, const IDOMImagePropertiesPtr &imageProperties)=0

  Creates a properly initialized image encoder object that writes to an outputstream that is specific to that image format.
- virtual IDOMImagePropertiesPtr getImageProperties ()=0

  Returns an object that stores the properties for this image object. The properties can then be inspected (or more added) by clients that need to manipulate the image resource.
- virtual eDOMImageType getImageType ()=0

  Retrieves the image type.
- virtual bool getIsRendered ()=0

  Determine if the image is as a result of rendering. This is indicated if the image type is eDITRendered or if the image explicitly notes this is the case (such as for IDOMPDFImage).
- virtual IDOMImagePtr getImageWithSubstitutedColorSpace (IEDLClassFactory *factory, const IDOMColor↩SpacePtr &colorSpace)

  Obtain an image that is the same as this image, but with the colorspace substituted for another.

Public Member Functions inherited from IDOMResource

- virtual IInputStreamPtr getStream () const =0

  Retrieves the resource stream.
- virtual uint64 getStreamLength () const =0

  Retrieves the stream length, if it is available.
- virtual const EDLSysString & getUri () const =0

  Retrieves the resource URI.
- virtual void setUri (const EDLSysString &uri)=0

  Sets the resource URI.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static EDL_API IDOMFilteredImagePtr create (IEDLClassFactory *pFactory, const IDOMImagePtr &source↩ Image, const IDOMImageFilterPtr &filter)

  Simplified creator for a filtered image with a single filter.
- static EDL_API IDOMFilteredImagePtr create (IEDLClassFactory *pFactory, const IDOMImagePtr &source↩ Image, const CDOMImageFilterVect &filters)

  Simplified creator for a filtered image with a vector of filters.
- static const CClassID & classID ()

  Retrieves class id of IDOMFilteredImage.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.184.1 Detailed Description

IDOMFilteredImage interface. Provides a method for filtering of an underlying image without requiring converted image data to be stored. It maintains a list of filters that are successively applied.

7.184.2 Member Function Documentation

classID()

```
static const CClassID & IDOMFilteredImage::classID ( )  [inline], [static]
```

Retrieves class id of IDOMFilteredImage.

Returns

CClassID Class id of the element

create() [1/2]

```
static EDL_API IDOMFilteredImagePtr IDOMFilteredImage::create (
            IEDLClassFactory * pFactory,
            const IDOMImagePtr & sourceImage,
            const CDOMImageFilterVect & filters )  [static]
```

Simplified creator for a filtered image with a vector of filters.

Parameters

pFactory	The EDL Class Factory
sourceImage	The source image.
filters	The vector of filters to instantiate with.

Returns

IDOMFilteredImagePtr The filtered image.

create() [2/2]

```
static EDL_API IDOMFilteredImagePtr IDOMFilteredImage::create (
            IEDLClassFactory * pFactory,
            const IDOMImagePtr & sourceImage,
            const IDOMImageFilterPtr & filter )  [static]
```

Simplified creator for a filtered image with a single filter.

Parameters

pFactory	The EDL Class Factory
sourceImage	The source image.
filter	The filter to use.

Returns

IDOMFilteredImagePtr The filtered image.

getFilterAtIndex()

```
virtual IDOMImageFilterPtr IDOMFilteredImage::getFilterAtIndex (
            uint32 index ) const  [pure virtual]
```

Get a filter from the filter chain at the given index.

Parameters

index	Index of the desired filter in the chain (beginning at 0)

Returns

IDOMImageFilterPtr The filter

getNumFilters()

```
virtual uint32 IDOMFilteredImage::getNumFilters ( ) const  [pure virtual]
```

Get the number of individual filters in the internal filter chain.

Returns

uint32 The number of filters.

getSourceImage()

```
virtual IDOMImagePtr IDOMFilteredImage::getSourceImage ( ) const  [pure virtual]
```

Get the source image.

Returns

IDOMImagePtr Smart pointer to the source image.

getStream()

```
virtual bool IDOMFilteredImage::getStream (
            IInputStreamPtr & stream ) const  [inline], [virtual]
```

This image type does not allow direct access to the underlying streams.

Parameters

stream	A smart pointer to the stream

Returns

bool Always false

pushFilter()

```
virtual void IDOMFilteredImage::pushFilter (
            const IDOMImageFilterPtr & filter )  [pure virtual]
```

Push a filter onto the end of the internal filter chain.

Parameters

filter	A pointer to the image filter to be added.

setStream()

```
virtual void IDOMFilteredImage::setStream (
            const IInputStreamPtr & stream )  [inline], [virtual]
```

This image type does not allow direct access to the underlying streams.

Parameters

stream	A smart pointer to the stream

Implements IDOMResource.

The documentation for this class was generated from the following file:

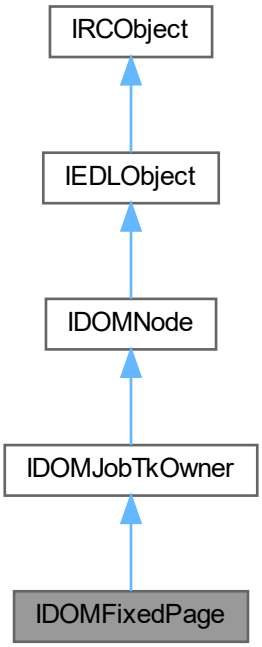
- idomimageresource.h

7.185 IDOMFixedPage Class Reference

Represents <FixedPage> element.

`#include <idompage.h>`

Inheritance diagram for IDOMFixedPage:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual void setWidth (double width)=0

  Sets the fixed page width.
- virtual double getWidth () const =0

  Retrieves the fixed page width.
- virtual void setHeight (double height)=0

  Sets the fixed page height.
- virtual double getHeight () const =0

  Retrieves the fixed page height.
- virtual const FRect & getContentBox () const =0

  Retrieves the fixed page's content box. The content bos is the area of the page containing imageable content that is to fit within the imageable area when printing or viewing.
- virtual void setContentBox (const FRect &bb)=0

  Sets the content box for the fixed page. The content bos is the area of the page containing imageable content that is to fit within the imageable area when printing or viewing. If omitted, the default value is a content box with an origin of (0,0) (that is, the top left hand corner of the page), with a width set to the width of the page and a height the same as that of the page.
- virtual const FRect & getBleedBox () const =0

  Retrieves the fixed page's bleed box. The bleed box is the area, inclusive of crop marks, that extends outside of the physical page. See Figure 8.11.
- virtual void setBleedBox (const FRect &bb)=0

  Sets the fixed page's bleed box. The bleed box is the area, inclusive of crop marks, that extends outside of the physical page. See Figure 8.11.
- virtual const FRect & getCropBox () const =0

  Retrieves the fixed page's crop box - the region to which the contents of the page are to be clipped (cropped) when displayed or printed.
- virtual void setCropBox (const FRect &cropBox)=0

  Sets the fixed page's crop box.
- virtual const FRect & getTrimBox () const =0

  Retrieves the fixed page's trim box. The trim box defines the intended dimensions of the finished page after trimming.
- virtual void setTrimBox (const FRect &trimBox)=0

  Sets the fixed page's trim box. The trim box defines the intended dimensions of the finished page after trimming.
- virtual EDLString getLanguage () const =0

  Retrieves default language of the fixed page and any of its children.
- virtual IDOMTransparencyGroupPtr getPageGroup () const =0

  Retrieves the page transparency group, if present. Generally this is used to dictate the color space in which transparency blending will be performed.
- virtual void setPageGroup (const IDOMTransparencyGroupPtr &pageGroup)=0

  Set the page transparency group.
- virtual void setLanguage (const EDLString &lang)=0

  Sets the language of the fixed page and any of its children.
- virtual IDOMResourceDictionaryPtr getResourceDictionary () const =0

  Retrieves a smart pointer to the resource dictionary. The resource dictionary contains the IDs of resources stored at this level of the document.
- virtual void setResourceDictionary (const IDOMResourceDictionaryPtr &ptrResources)=0

  Sets the resource dictionary.
- virtual IDOMImagePtr getThumbnail () const =0

  Retrieves a smart pointer to the fixed document thumbnail image.
- virtual void setThumbnail (const IDOMImagePtr &ptrThumbnail)=0

  Sets a new thumbnail image for the document sequence. The image must be in either JPEG or PNG format. Setting thumbnail to NULL deletes the document sequence thumbnail.

Public Member Functions inherited from IDOMJobTkOwner

- virtual IDOMJobTkPtr getJobTicket () const =0

  Retrieves the JobTicket held by this node.
- virtual void setJobTicket (const IDOMJobTkPtr &ptrJobTicket)=0

  Sets the JobTicket for the node.

Public Member Functions inherited from IDOMNode

- virtual ~**IDOMNode** ()

  virtual destructor
- virtual DOMid getDOMid () const =0

  Retrieves the node ID.
- virtual void setDOMid (DOMid id)=0

  Sets the node ID.
- virtual eDOMNodeType getNodeType () const =0

  Retrieves the DOM node type.
- virtual bool getProperty (const EDLSysString &propertyName, PValue &propertyValue) const =0

  Retrieves the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.
- virtual void setProperty (const EDLSysString &propertyName, const PValue &propertyValue)=0

  Sets the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.
- virtual void removeProperty (const EDLSysString &propertyName)=0

  Removes property.
- virtual IEDLSysStringCollectionEnumPtr getPropertyCollectionEnum ()=0

  Retrieves a navigable list of the property names stored on this node.
- virtual bool hasChildNodes () const =0

  Function that indicates whether this node is a parent to other nodes.
- virtual IDOMNodePtr getParentNode () const =0

  Gets the parent node of this node.
- virtual IDOMNodePtr getFirstChild () const =0

  Gets the first child node of this node.
- virtual IDOMNodePtr getLastChild () const =0

  Gets the last child node of this node.
- virtual IDOMNodePtr getNextChild (const IDOMNodePtr &child) const =0

  Gets the child node which follows the node passed in.
- virtual IDOMNodePtr getPreviousChild (const IDOMNodePtr &child) const =0

  Gets the child node which precedes the node passed in.
- virtual IDOMNodePtr getPreviousSibling () const =0

  Retrieves the node's previous sibling node.
- virtual IDOMNodePtr getNextSibling () const =0

  Retrieves node's next sibling node.
- virtual void appendChild (const IDOMNodePtr &child)=0

  Appends a node to the end of the node's child list.
- virtual void insertChild (const IDOMNodePtr &ptrPreviousSibling, const IDOMNodePtr &child, bool bCheck↩ Complete=true)=0

  Insert a child node after ptrPreviousSibling.
- virtual IDOMNodePtr extractChild (const IDOMNodePtr &child)=0

Extracts (that is, finds and removes) a child node from the node children. After extraction the child node is no longer a part of the DOM. If no node is specified, the first available node will be extracted from the node's children.

- virtual void replaceChild (const IDOMNodePtr &oldChild, const IDOMNodePtr &newChild)=0

  Replaces the child node with another.
- virtual bool isComplete () const =0

  Signals the completeness of the node.
  A complete node is one that has no more children to be added to it.
- virtual void **setComplete** ()=0

  Sets the node's completeness status to "true".
- virtual IDOMNodeFlags * getFlags ()=0

  Retrieves the node's flags property.
- virtual void setParentNode (const IDOMNodePtr &ptrParent)=0

  Sets the parent node.
- virtual void setPreviousSibling (const IDOMNodePtr &ptrPreviousSibling)=0

  Sets the previous sibling node.
- virtual void setNextSibling (const IDOMNodePtr &ptrNextSibling)=0

  Sets the next sibling node.
- virtual bool isAncestor (const IDOMNodePtr &ptrCandidate)=0

  Function tests whether a candidate node is a descendant of the node.
- virtual FRect getBounds (bool applyTransform=true, bool applyClip=true)

  Find the conservative bounding box of the marking content of the node.
- virtual bool copyNodeData (IDOMNode *pSourceNode)=0

  Copy the properties collection, the flags and the DOM ID from the given source node to this one.
- virtual IDOMNodePtr cloneNode (IEDLClassFactory *pFactory) const =0

  Simpified node cloning. An exception of type IEDLError will be thrown on failure.
- virtual IDOMNodePtr cloneTree (IEDLClassFactory *pFactory) const =0

  Clone the tree of nodes beginning at this node. An exception of type IEDLError will be thrown on failure.
- virtual void cloneTreeAndAppend (IEDLClassFactory *pFactory, const IDOMNodePtr &dest) const =0

  Clone the tree of nodes beginning at this node, and append the result to the destination tree.
- virtual void **completeTree** ()=0

  Mark the entire tree from this point as complete. You should not ordinarily need to call this function.
- virtual void **removeCompleteFlagFromTree** ()=0

  Mark the entire tree from this point as complete.
- virtual void findChildrenOfType (eDOMNodeType type, CDOMNodeVect &nodes, bool searchForms=false)=0

  Find all children of this node with the given type, appending to the given vector. Does not descend into brushes.
- virtual void walkTree (WalkTreeFunc func, void *priv, bool descendIntoBrushes=false, bool descendInto↩ Forms=false)=0

  Walk through the DOM calling a given function on each node. The function is allowed to:
- virtual void notifyOnDestruct (NodeDeleteFunc func, void *priv)=0

  Register interest in being told when this node is about to be destroyed.
- virtual void unregisterNotify (NodeDeleteFunc func, void *priv)=0

  Unregister interest in being told when this node is about to be destroyed.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static EDL_API IDOMFixedPagePtr create (IEDLClassFactory *pFactory, double width=793.7, double height=1122.5)

  Simplified creation function for IDOMFixedPage Throws an IEDLError exception on failure.
- static const CClassID & classID ()

  Retrieves the class id of IDOMFixedPage.

Static Public Member Functions inherited from IDOMJobTkOwner

- static const CClassID & classID ()

  Retrieves the class id of IDOMJobTkOwner.

Static Public Member Functions inherited from IDOMNode

- static EDL_API FMatrix effectiveTransformationOfNode (const IDOMNodePtr &node)

  Attempt to find the effective transformation matrix external to the specified node relative to either a containing page or ultimate parent.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.185.1 Detailed Description

Represents <FixedPage> element.

A fixed page describes the contents of a page. A fixed page contains all of the visual elements on each page. Each page has a fixed size and orientation. The layout of the visual elements on a page is determined by the fixed page markup. This applies to both graphics and text. The fixed page contains the elements that together form the basis for all markings rendered on the page: that is, Paths, Glyphs and Canvases.

The fixed page specifies a height, width and a default language.

7.185.2 Member Function Documentation

classID()

```
static const CClassID & IDOMFixedPage::classID ( )  [inline], [static]
```

Retrieves the class id of IDOMFixedPage.

Returns

CClassID Class id of the IDOMFixedPage

create()

```
static EDL_API IDOMFixedPagePtr IDOMFixedPage::create (
            IEDLClassFactory * pFactory,
            double width = 793.7,
            double height = 1122.5 )  [static]
```

Simplified creation function for IDOMFixedPage Throws an IEDLError exception on failure.

Parameters

pFactory	The EDL class factory
width	Description for width (defaults to 210mm, width of A4)
height	Description for height (defaults to 297mm, height of A4)

Returns

IDOMFixedPagePtr A smart pointer to the fixed page

getBleedBox()

```
virtual const FRect & IDOMFixedPage::getBleedBox ( ) const  [pure virtual]
```

Retrieves the fixed page's bleed box. The bleed box is the area, inclusive of crop marks, that extends outside of the physical page. See Figure 8.11.

These values are specified in units of 1/96 inch.

Returns

FRect The fixed page's bleed box.

getContentBox()

```
virtual const FRect & IDOMFixedPage::getContentBox ( ) const  [pure virtual]
```

Retrieves the fixed page's content box. The content bos is the area of the page containing imageable content that is to fit within the imageable area when printing or viewing.

These values are specified in units of 1/96 inch.

Returns

FRect The fixed page's content box

getCropBox()

```
virtual const FRect & IDOMFixedPage::getCropBox ( ) const  [pure virtual]
```

Retrieves the fixed page's crop box - the region to which the contents of the page are to be clipped (cropped) when displayed or printed.

Returns

FRect The fixed page's crop box.

getHeight()

```
virtual double IDOMFixedPage::getHeight ( ) const  [pure virtual]
```

Retrieves the fixed page height.

Returns

double Returns the fixed page height

getLanguage()

```
virtual EDLString IDOMFixedPage::getLanguage ( ) const  [pure virtual]
```

Retrieves default language of the fixed page and any of its children.

The language is specified according to RFC 3066. English is defined as en_GB and American English as en_US. There is no default setting. If the language is not known it is set to und (undetermined). For further information see http://www.w3.org/International/articles/language-tags/

Returns

EDLString The language of the fixed page.

getPageGroup()

```
virtual IDOMTransparencyGroupPtr IDOMFixedPage::getPageGroup ( ) const  [pure virtual]
```

Retrieves the page transparency group, if present. Generally this is used to dictate the color space in which transparency blending will be performed.

Returns

IDOMTransparencyGroupPtr Smart pointer to the page group, or NULL if no page group is supplied.

getResourceDictionary()

```
virtual IDOMResourceDictionaryPtr IDOMFixedPage::getResourceDictionary ( ) const  [pure virtual]
```

Retrieves a smart pointer to the resource dictionary. The resource dictionary contains the IDs of resources stored at this level of the document.

Returns

IDOMResourceDictionaryPtr The resource dictionary or NULL of no resource dictionary is present.

getThumbnail()

```
virtual IDOMImagePtr IDOMFixedPage::getThumbnail ( ) const  [pure virtual]
```

Retrieves a smart pointer to the fixed document thumbnail image.

Returns

IDOMImagePtr The thumbnail image or NULL if no thumbmail is present.

getTrimBox()

```
virtual const FRect & IDOMFixedPage::getTrimBox ( ) const  [pure virtual]
```

Retrieves the fixed page's trim box. The trim box defines the intended dimensions of the finished page after trimming.

These values are specified in units of 1/96 inch.

Returns

FRect The fixed page's trim box.

getWidth()

```
virtual double IDOMFixedPage::getWidth ( ) const  [pure virtual]
```

Retrieves the fixed page width.

Returns

double Returns the fixed page width.

setBleedBox()

```
virtual void IDOMFixedPage::setBleedBox (
            const FRect & bb )  [pure virtual]
```

Sets the fixed page's bleed box. The bleed box is the area, inclusive of crop marks, that extends outside of the physical page. See Figure 8.11.

These values are specified in units of 1/96 inch.

Parameters

bb	The new bleed box settings.

setContentBox()

```
virtual void IDOMFixedPage::setContentBox (
            const FRect & bb )  [pure virtual]
```

Sets the content box for the fixed page. The content bos is the area of the page containing imageable content that is to fit within the imageable area when printing or viewing. If omitted, the default value is a content box with an origin of (0,0) (that is, the top left hand corner of the page), with a width set to the width of the page and a height the same as that of the page.

These values are specified in units of 1/96 inch.

Parameters

bb	The new content box settings

setCropBox()

```
virtual void IDOMFixedPage::setCropBox (
            const FRect & cropBox )  [pure virtual]
```

Sets the fixed page's crop box.

Parameters

cropBox	The new crop box settings.

setHeight()

```
virtual void IDOMFixedPage::setHeight (
            double height )  [pure virtual]
```

Sets the fixed page height.

The height of the page is expressed as a real number in units of 1/96th of an inch. If either the width or height of the page (or both) is set to greater than the original page width or height, a larger page will result. This would have more white space toward the right and/or bottom of the page when the document is viewed in an XPS viewer, since the document origin is in the top-left corner. Changing the page dimensions has no effect on the coordinates of the page content.

Parameters

height	New fixed page height.

setLanguage()

```
virtual void IDOMFixedPage::setLanguage (
            const EDLString & lang )  [pure virtual]
```

Sets the language of the fixed page and any of its children.

The language is specified according to RFC 3066. English is defined as en_GB and American English as en_US. There is no default setting. For further information see http://www.w3.org/↩International/articles/language-tags/

Parameters

lang	The new language setting.

setPageGroup()

```
virtual void IDOMFixedPage::setPageGroup (
            const IDOMTransparencyGroupPtr & pageGroup )  [pure virtual]
```

Set the page transparency group.

Parameters

pageGroup	A reference to the desired transparency group, or NULL.

setResourceDictionary()

```
virtual void IDOMFixedPage::setResourceDictionary (
            const IDOMResourceDictionaryPtr & ptrResources )  [pure virtual]
```

Sets the resource dictionary.

Parameters

ptrResources	Smart pointer to the new resource dictionary The resource dictionary contains the IDs of resources stored at this level of the document.

setThumbnail()

```
virtual void IDOMFixedPage::setThumbnail (
            const IDOMImagePtr & ptrThumbnail )  [pure virtual]
```

Sets a new thumbnail image for the document sequence. The image must be in either JPEG or PNG format. Setting thumbnail to NULL deletes the document sequence thumbnail.

Parameters

ptrThumbnail	Smart pointer to new thumbnail image.

setTrimBox()

```
virtual void IDOMFixedPage::setTrimBox (
            const FRect & trimBox )  [pure virtual]
```

Sets the fixed page's trim box. The trim box defines the intended dimensions of the finished page after trimming.

These values are specified in units of 1/96 inch.

Parameters

trimBox	The new trim box settings.

setWidth()

```
virtual void IDOMFixedPage::setWidth (
            double width )  [pure virtual]
```

Sets the fixed page width.

The width of the page is expressed as a real number in units of 1/96th of an inch. If either the width or height of the page (or both) is set to greater than the original page width or height, a larger page will result. This would have more white space toward the right and/or bottom of the page when the document is viewed in an XPS viewer, since the document origin is in the top-left corner. Changing the page dimensions has no effect on the coordinates of the page content.

Parameters

width	New fixed page width

The documentation for this class was generated from the following file:

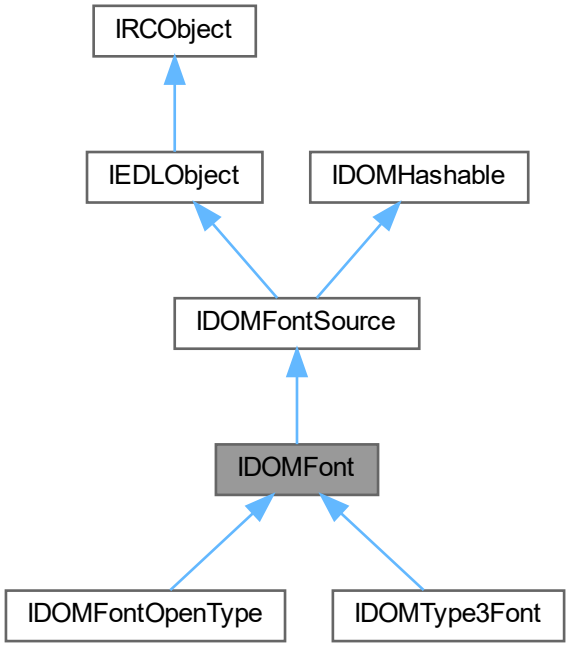
- idompage.h

7.186 IDOMFont Class Reference

IDOMFont Base Class.

```
#include <idomfont.h>
```

Inheritance diagram for IDOMFont:



Classes

- class Data

  Initialization data.

Public Types

- enum eFontType { eFontTypeUnknown , eFontTypeOpenType , eFontType3 , eFontTypePCLXL , eFontTypePCL5 }

  type used to uniquely identify the type of font
- typedef uint32 **CharCode**

  type used to uniquely identify a character code
- typedef std::map< CharCode, IDOMGlyph::GlyphID > **CCharacterMap**

  Map type used for storing character code to glyph ID mappings.

Public Types inherited from IDOMFontSource

- enum eFontSourceType { eFontSourceTypeNone , eFontSourceTypeStreamFilter , eFontSourceTypeStream , eFontSourceTypeFont }

  type used to uniquely identify the source type of a font

Public Member Functions

- virtual eFontType getFontType () const =0

  Gets the font type. See eFontType for more information on font types.
- virtual IDOMFontSourcePtr getFontSource () const =0

  Get the font source of this font.
- virtual void setFontSource (const IDOMFontSourcePtr &fontSource)=0

  Sets the font source for this font.
- virtual IInputStreamPtr getFontBaseStream () const =0

  Return the base stream for this font, obtaining it from the font source.
- virtual void getCharacterMap (CCharacterMap &characterMap, uint32 fontIndex=0U)=0

  Get the character map for this font.

Public Member Functions inherited from IDOMFontSource

- virtual eFontSourceType getFontSourceType () const =0

  Gets the font source type.
- virtual const EDLSysString & determineUri () const =0

  Determines the URI based on the font source (underlying font sources may be searched)

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static const CClassID & classID ()
  Retrieves class id of IDOMFont.

Static Public Member Functions inherited from IDOMFontSource

- static const CClassID & classID ()
  Retrieves the class id of IDOMFontSource.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()
  Virtual destructor.

7.186.1 Detailed Description

IDOMFont Base Class.

7.186.2 Member Function Documentation

classID()

```
static const CClassID & IDOMFont::classID ( )  [inline], [static]
```

Retrieves class id of IDOMFont.

Returns

CClassID class id of the element

getCharacterMap()

```
virtual void IDOMFont::getCharacterMap (
            CCharacterMap & characterMap,
            uint32 fontIndex = 0U )  [pure virtual]
```

Get the character map for this font.

Parameters

characterMap	The character map to populate.
fontIndex	The index of the font - only used for TTC fonts.

getFontBaseStream()

```
virtual IInputStreamPtr IDOMFont::getFontBaseStream ( ) const  [pure virtual]
```

Return the base stream for this font, obtaining it from the font source.

Returns

IInputStreamPtr The base stream

getFontSource()

```
virtual IDOMFontSourcePtr IDOMFont::getFontSource ( ) const  [pure virtual]
```

Get the font source of this font.

Returns

IDOMFontSourcePtr The font source.

getFontType()

```
virtual eFontType IDOMFont::getFontType ( ) const  [pure virtual]
```

Gets the font type. See eFontType for more information on font types.

Returns

eFontType. Returns the font type.

setFontSource()

```
virtual void IDOMFont::setFontSource (
            const IDOMFontSourcePtr & fontSource )  [pure virtual]
```

Sets the font source for this font.

Parameters

fontSource	The new font source

The documentation for this class was generated from the following file:

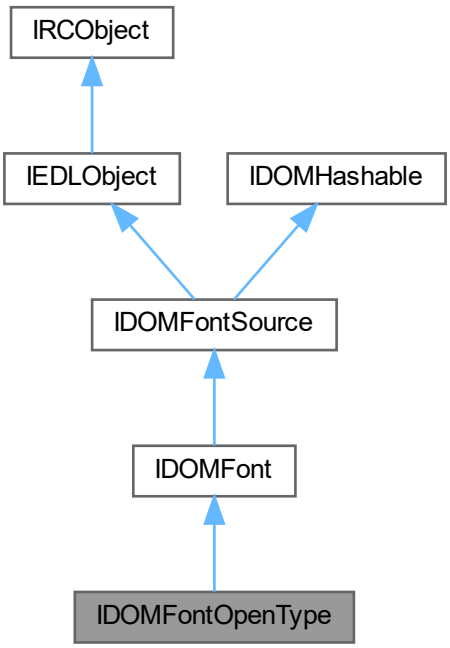
- idomfont.h

7.187 IDOMFontOpenType Class Reference

IDOMFontOpenType interface.

`#include <idomfont.h>`

Inheritance diagram for IDOMFontOpenType:



Classes

- class CCIDMap

  For TrueType-based CIDFonts, the mapping from CIDs to GlyphIds for non-identity orderings.
- class Data

  Initialization data.

Public Types

- enum eOpenTypeFontType { eOpenTypeFontTypeUnknown , eOpenTypeFontTypeTTF , eOpenTypeFontTypeCFF , eOpenTypeFontTypeTTC }

  Type used to uniquely identify the type of OpenType font.
- enum eOriginalFontType { eOriginalTypeOpenType , eOriginalType1 , eOriginalType2 , eOriginalType42 , eOriginalType9 , eOriginalType11 , eOriginalPclTrueType }

  Type used to uniquely identify the original type of Font. In the PostScript/PDF inputs, most font types are converted to OpenType before insertion into the DOM. This allows the ability to discover what the original type was.

Public Types inherited from IDOMFont

- enum eFontType { eFontTypeUnknown , eFontTypeOpenType , eFontType3 , eFontTypePCLXL , eFontTypePCL5 }

  type used to uniquely identify the type of font
- typedef uint32 **CharCode**

  type used to uniquely identify a character code
- typedef std::map< CharCode, IDOMGlyph::GlyphID > **CCharacterMap**

  Map type used for storing character code to glyph ID mappings.

Public Types inherited from IDOMFontSource

- enum eFontSourceType { eFontSourceTypeNone , eFontSourceTypeStreamFilter , eFontSourceTypeStream , eFontSourceTypeFont }

  type used to uniquely identify the source type of a font

Public Member Functions

- virtual bool getObfuscated () const =0

  Returns true if font is obfuscated. Obfuscated fonts are only found in XPS Documents.
- virtual bool getIsPSStandardFont () const =0

  Establishes whether the font is a standard PostScript font.
- virtual bool getIsPDFStandardFont () const =0

  Establishes whether the font is a standard PDF font.
- virtual bool getEmbedded () const =0

  Establishes whether the font is flagged for embeddeding.
- virtual void setEmbedded (bool embedded)=0

  Sets whether of not the font is flagged for embeddeding.
- virtual bool getSubsetted () const =0

  Requests if the font reports to be a subset.
- virtual EDLSysString getRequestedFontName () const =0

  Get the name of the font as requested by the input document. This is only useful for PDF and PostScript input. If a font has to be substituted for another requested font, this will return the name of the font the input desired.
- virtual EDLSysString getPostScriptName (IEDLClassFactory *pFactory, int32 fontIndex)=0

  Get the "PostScript" Name of the font, from the font data itself. This is usually extracted from the 'name' OpenType table, but may be synthesized if the name cannot be obtained.
- virtual EDLSysString getFullName (IEDLClassFactory *pFactory, int32 fontIndex)=0

  Get the "Full" Name of the font, from the font data itself. This is usually extracted from the 'name' OpenType table, but may be synthesized if the name cannot be obtained.
- virtual eOpenTypeFontType getOpenTypeFontType ()=0

  Returns the sub font type for this opentype font.
- virtual eOriginalFontType getOriginalFontType ()=0

  Returns the original font type for this opentype font.
- virtual IFontOpenTypeTableAccessorPtr getFontOpenTypeTableAccessor (uint32 fontIndex=0)=0

  Creates an OpenType font table accessor.
- virtual IFontTrueTypeGlyphAccessorPtr getFontTrueTypeGlyphAccessor (uint32 fontIndex=0, bool stripInstructions=false)=0

  Creates a TrueType glyph accessor.
- virtual uint16 getFontLicenseFromOS2Table (uint32 fontIndex=0)=0

  Gets the fsType field (embedded licensing information) from the OS/2 table.

- virtual bool getIsRestricted (uint32 fontIndex=0)=0

  Is the font marked as restricted from embedding according to its license?
- virtual IDOMFontOpenTypeTTPtr createTrueTypeOnlyFontVersion (const ISessionPtr &ptrSession, bool regenerateStream)=0

  Create a TrueType only font from this OpenType font that may contain CFF fonts.
- virtual IDOMFontOpenTypePtr createSubsetFont (const ISessionPtr &ptrSession, const CEDLVector< uint16 > &usedGlyphs, const CEDLVector< uint32 > &usedUnicode)=0

  Create a Subsetted version if this font.
- virtual IDOMFontOpenTypePtr createRenamedFont (const ISessionPtr &ptrSession, const EDLSysString &fontName, uint32 fontIndex=0)=0

  Get a renamed version of this font. An exception of type IEDLError is thrown on failure.
- virtual const CEDLSimpleBuffer & getOriginalOS2Table () const =0

  Obtain the original OS/2 Table, if present, that was included in the font.
- virtual CCIDMapConstPtr getCidMap () const =0

  Obtain the CID Map for this font, if present.
- virtual void setCidMap (const CCIDMapConstPtr &cidMap)=0

  Set or clear the CIDMap for the font. Valid only for TrueType fonts.
- virtual bool validateInstructions (const ISessionPtr &session, uint32 fontIndex=0)=0

  Validate glyph instructions.
- virtual IDOMFontOpenTypePtr stripInstructions (const ISessionPtr &session, uint32 fontIndex=0, bool all=false)=0

  Strip glyph instructions.

Public Member Functions inherited from IDOMFont

- virtual eFontType getFontType () const =0

  Gets the font type. See eFontType for more information on font types.
- virtual IDOMFontSourcePtr getFontSource () const =0

  Get the font source of this font.
- virtual void setFontSource (const IDOMFontSourcePtr &fontSource)=0

  Sets the font source for this font.
- virtual IInputStreamPtr getFontBaseStream () const =0

  Return the base stream for this font, obtaining it from the font source.
- virtual void getCharacterMap (CCharacterMap &characterMap, uint32 fontIndex=0U)=0

  Get the character map for this font.

Public Member Functions inherited from IDOMFontSource

- virtual eFontSourceType getFontSourceType () const =0

  Gets the font source type.
- virtual const EDLSysString & determineUri () const =0

  Determines the URI based on the font source (underlying font sources may be searched)

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static EDL_API IDOMFontOpenTypePtr create (IEDLClassFactory *pFactory, const IInputStreamPtr &stream)

  Simplified creator of a font from a stream. Throws an IEDLError exception on failure.
- static const CClassID & classID ()

  Retrieves the class id of IDOMFontOpenType.

Static Public Member Functions inherited from IDOMFont

- static const CClassID & classID ()

  Retrieves class id of IDOMFont.

Static Public Member Functions inherited from IDOMFontSource

- static const CClassID & classID ()

  Retrieves the class id of IDOMFontSource.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.187.1 Detailed Description

IDOMFontOpenType interface.

7.187.2 Member Function Documentation

classID()

```
static const CClassID & IDOMFontOpenType::classID ( )  [inline], [static]
```

Retrieves the class id of IDOMFontOpenType.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMFontOpenTypePtr IDOMFontOpenType::create (
            IEDLClassFactory * pFactory,
            const IInputStreamPtr & stream )  [static]
```

Simplified creator of a font from a stream. Throws an IEDLError exception on failure.

Parameters

pFactory	EDL class factory to use.
stream	The font stream.

Returns

IDOMFontOpenTypePtr the new font

createRenamedFont()

```
virtual IDOMFontOpenTypePtr IDOMFontOpenType::createRenamedFont (
            const ISessionPtr & ptrSession,
            const EDLSysString & fontName,
            uint32 fontIndex = 0 )  [pure virtual]
```

Get a renamed version of this font. An exception of type IEDLError is thrown on failure.

Parameters

ptrSession	A pointer to the EDL session.
fontName	The desired font name
fontIndex	The index of the font, if the font being renamed is a TrueType collection. The default is zero.

Returns

IDOMFontOpenTypePtr The renamed font.

createSubsetFont()

```
virtual IDOMFontOpenTypePtr IDOMFontOpenType::createSubsetFont (
            const ISessionPtr & ptrSession,
            const CEDLVector< uint16 > & usedGlyphs,
            const CEDLVector< uint32 > & usedUnicode )  [pure virtual]
```

Create a Subsetted version if this font.

Provide a vector of glyph IDs and/or unicode codepoints, and a subset font containing only those glyphs required will be generated. Currently only fonts with 3,0 or 3,1 format 4 subtables or a 3,10 format 12 subtable are supported. Glyphs used by composite glyphs that are to be retained in the subfont will be automatically included. Glyphs or codepoints that are out of range will be ignored.

Parameters

ptrSession	A pointer to the EDL session.
usedGlyphs	A vector of glyph IDs that should be retained in the subset font.
usedUnicode	A vector of unicode code points whose underlying glyphs should be retained in the subset font.

Returns

IDOMFontOpenTypePtr The subsetted font

createTrueTypeOnlyFontVersion()

```
virtual IDOMFontOpenTypeTTPtr IDOMFontOpenType::createTrueTypeOnlyFontVersion (
            const ISessionPtr & ptrSession,
            bool regenerateStream )  [pure virtual]
```

Create a TrueType only font from this OpenType font that may contain CFF fonts.

Parameters

ptrSession	A pointer to the EDL session.
regenerateStream	Determines whether the stream should be regenerated

Returns

IDOMFontOpenTypeTTPtr The resulting TrueType-only font.

getCidMap()

```
virtual CCIDMapConstPtr IDOMFontOpenType::getCidMap ( ) const  [pure virtual]
```

Obtain the CID Map for this font, if present.

```
The CIDMap in a PDF CID font describes the mapping from PDF CIDs to
glyph IDs. In PDF, CIDs may come from a particular ordering and supplement.
That is, the CIDs conform to a known and understood character set.

This information is only useful for creating PDF output where the CID information
needs to be preserved for best interoperability, especially if the font is not
to be embedded. This information is not required to use this font.

Mako's PDF input will only create this entry for CID fonts where the CIDMap is
present and uses an ordering other than Identity.

Only relevant for TrueType-based CID Fonts.
```

Returns

CCIDMapConstPtr The CIDMap, or NULL if not present.

getEmbedded()

```
virtual bool IDOMFontOpenType::getEmbedded ( ) const  [pure virtual]
```

Establishes whether the font is flagged for embeddeding.

Inputs will set this to true if the font was embedded in the input or false otherwise. Outputs will honour this setting, if possible, or not overridden. Currently this will only occur for PDF and PS output.

Returns

bool True if the font is flagged for embedding

getFontLicenseFromOS2Table()

```
virtual uint16 IDOMFontOpenType::getFontLicenseFromOS2Table (
            uint32 fontIndex = 0 )  [pure virtual]
```

Gets the fsType field (embedded licensing information) from the OS/2 table.

Parameters

fontIndex	The index of the font

Returns

uint16 The function returns the fsType field from the OS/2 table, or zero if the table does not exist in the font.

getFontOpenTypeTableAccessor()

```
virtual IFontOpenTypeTableAccessorPtr IDOMFontOpenType::getFontOpenTypeTableAccessor (
            uint32 fontIndex = 0 )  [pure virtual]
```

Creates an OpenType font table accessor.

Parameters

fontIndex	The index of the font - only used for TTC fonts.

Returns

IFontOpenTypeTableAccessorPtr The accessor

getFontTrueTypeGlyphAccessor()

```
virtual IFontTrueTypeGlyphAccessorPtr IDOMFontOpenType::getFontTrueTypeGlyphAccessor (
            uint32 fontIndex = 0,
            bool stripInstructions = false )  [pure virtual]
```

Creates a TrueType glyph accessor.

Parameters

fontIndex	The index of the font - only used for TTC fonts.
stripInstructions	Determines whether instructions are stripped. Default is false

Returns

IFontTrueTypeGlyphAccessorPtr The accessor

getFullName()

```
virtual EDLSysString IDOMFontOpenType::getFullName (
            IEDLClassFactory * pFactory,
            int32 fontIndex )  [pure virtual]
```

Get the "Full" Name of the font, from the font data itself. This is usually extracted from the 'name' OpenType table, but may be synthesized if the name cannot be obtained.

Parameters

fontIndex	The index of the desired font (if a TrueType collection)
pFactory	A pointer to an EDL Class factory.

Returns

EDLSysString The full font name

getIsPDFStandardFont()

```
virtual bool IDOMFontOpenType::getIsPDFStandardFont ( ) const  [pure virtual]
```

Establishes whether the font is a standard PDF font.

Returns

bool True if the font was selected when the input requested a standard PDF font

getIsPSStandardFont()

```
virtual bool IDOMFontOpenType::getIsPSStandardFont ( ) const  [pure virtual]
```

Establishes whether the font is a standard PostScript font.

Returns

bool True if the font was selected when the input requested a standard PostScript font

getIsRestricted()

```
virtual bool IDOMFontOpenType::getIsRestricted (
            uint32 fontIndex = 0 )  [pure virtual]
```

Is the font marked as restricted from embedding according to its license?

Parameters

fontIndex	The index of the font

Returns

bool true if the font is restricted, false otherwise.

getObfuscated()

```
virtual bool IDOMFontOpenType::getObfuscated ( ) const  [pure virtual]
```

Returns true if font is obfuscated. Obfuscated fonts are only found in XPS Documents.

Embedded font obfuscation is a means of preventing end users from using standard ZIP utilities to extract fonts from XPS format documents and install them on their own systems.

Returns

bool Returns true if the font is obfuscated, false if it is not obfuscated

getOpenTypeFontType()

```
virtual eOpenTypeFontType IDOMFontOpenType::getOpenTypeFontType ( )  [pure virtual]
```

Returns the sub font type for this opentype font.

Returns

eOpenTypeFontType The opentype font type

getOriginalFontType()

```
virtual eOriginalFontType IDOMFontOpenType::getOriginalFontType ( )  [pure virtual]
```

Returns the original font type for this opentype font.

Returns

eOriginalFontType The original font type.

getOriginalOS2Table()

```
virtual const CEDLSimpleBuffer & IDOMFontOpenType::getOriginalOS2Table ( ) const  [pure virtual]
```

Obtain the original OS/2 Table, if present, that was included in the font.

```
Currently this information is only populated for PDF input, and only if
the font had an OS/2 table. OS/2 information in fonts embedded in PDF tends
to be problematic, and as such, much of the OS/2 table is regenerated partially
or completely by PDF input.

For uses which require the original OS/2 table data, it is made available here.

Please note that this information is not used when font hashing is performed.
```

Returns

CEDLSimpleBuffer A reference to the original OS/2 table, or an empty buffer.

getPostScriptName()

```
virtual EDLSysString IDOMFontOpenType::getPostScriptName (
            IEDLClassFactory * pFactory,
            int32 fontIndex )  [pure virtual]
```

Get the "PostScript" Name of the font, from the font data itself. This is usually extracted from the 'name' OpenType table, but may be synthesized if the name cannot be obtained.

Parameters

pFactory	A pointer to an EDL Class factory.
fontIndex	The index of the desired font (if a TrueType collection)

Returns

EDLSysString The PostScript font name

getRequestedFontName()

```
virtual EDLSysString IDOMFontOpenType::getRequestedFontName ( ) const  [pure virtual]
```

Get the name of the font as requested by the input document. This is only useful for PDF and PostScript input. If a font has to be substituted for another requested font, this will return the name of the font the input desired.

Returns

EDLSysString The requested font name if applicable, an empty string otherwise.

getSubsetted()

```
virtual bool IDOMFontOpenType::getSubsetted ( ) const  [pure virtual]
```

Requests if the font reports to be a subset.

Currently, this is only set to true for PDF input, where the PDF indicates that the font is subset.

Returns

bool True if the font reports that it is a subset

setCidMap()

```
virtual void IDOMFontOpenType::setCidMap (
            const CCIDMapConstPtr & cidMap )  [pure virtual]
```

Set or clear the CIDMap for the font. Valid only for TrueType fonts.

Parameters

cidMap	The CID map to set. Ownership is taken by the font.

setEmbedded()

```
virtual void IDOMFontOpenType::setEmbedded (
            bool embedded )  [pure virtual]
```

Sets whether of not the font is flagged for embeddeding.

Outputs will honour this setting, if possible, or not overridden. In particular, if a font is a standard font in the output format being used, this setting will be ignored. Currently this will only occur for PDF and PS output.

Parameters

embedded	Set to true to flag the font for embedding

stripInstructions()

```
virtual IDOMFontOpenTypePtr IDOMFontOpenType::stripInstructions (
```

```
            const ISessionPtr & session,
            uint32 fontIndex = 0,
            bool all = false )  [pure virtual]
```

Strip glyph instructions.

Applies only to TrueType glyphs. This will attempt to strip some cases of broken instructions (hints) from a font.

Note that this it cannot detect issues seen only when rendering at a specific resolution and point size.

Parameters

session	A pointer to the session.
fontIndex	The index of the font - only used for TTC fonts.
all	Whether to strip all instructions from the font, or only instructions found to be invalid (the default).

Returns

IDOMFontOpenTypePtr The stripped font.

validateInstructions()

```
virtual bool IDOMFontOpenType::validateInstructions (
            const ISessionPtr & session,
            uint32 fontIndex = 0 )  [pure virtual]
```

Validate glyph instructions.

Applies only to TrueType glyphs. This will attempt to detect some cases of broken instructions (hints) by rendering each glyph in the font and return a boolean value to indicate if any issues were found.

Note that while this function may be useful to detect general issues with fonts/glyph instructions it cannot detect issues seen only when rendering at a specific resolution and point size.

Parameters

session	A pointer to the session.
fontIndex	The index of the font - only used for TTC fonts.

Returns

bool true on success, or false if invalid instructions were found.

The documentation for this class was generated from the following file:

- idomfont.h

7.188 IDOMFontOTFTrueType Class Reference

Opentype Font.

```
#include <idomfont.h>
```

7.188.1 Detailed Description

Opentype Font.

The documentation for this class was generated from the following file:

- idomfont.h

7.189 IDOMFontPCL5 Class Reference

IDOMFontPCL5 (PCL5 Truetype) derived from an OpenType font source.

`#include <idomfontpcl.h>`

Inherits IDOMFontOpenTypeTT.

Classes

- class Data

  Initialization data.

Public Types

- enum ePCL5FontType

  Type used to uniquely identify the type of PCL5 font (Currently, only TTF supported).
- typedef uint16 **SymbolSetIDCode**

  Type represents the PCL5 Font Symbol Set.

Public Member Functions

- virtual IDOMFontPCL5::ePCL5FontType getPCL5FontType () const =0

  Returns the sub font type for this PCLXL font.
- virtual bool getFontName (EDLSysString &fontName)=0

  Returns the XL font name.
- virtual bool setSymbolSetIDCode (IDOMFontPCL5::SymbolSetIDCode symbolSet)=0

  Set the font symbol set to be used in the generated font.
- virtual IDOMFontPCL5::SymbolSetIDCode getSymbolSetIDCode () const =0

  Returns the font symbol set ID as an ID Code.
- virtual bool getSymbolSetID (EDLSysString &symbolSetID) const =0

  Returns the font symbol set ID.
- virtual bool getFormat15FontBlockEnumerator (IFontPCL5WriteSegmentBlockEnumeratorPtr &enumerator)=0

  Creates a Format 15 font block enumerator.
- virtual bool getFormat16FontBlockEnumerator (IFontPCL5WriteSegmentBlockEnumeratorPtr &enumerator)=0

  Creates a Format 16 font block enumerator.
- virtual bool getFontTrueTypeGlyphAccessor (IFontPCL5TrueTypeGlyphAccessorPtr &accessor)=0

  Creates a TrueType glyph accessor.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves class id of IDOMFontPCL5.

7.189.1 Detailed Description

IDOMFontPCL5 (PCL5 Truetype) derived from an OpenType font source.

7.189.2 Member Function Documentation

classID()

```
static const CClassID & IDOMFontPCL5::classID ( )  [inline], [static]
```

Retrieves class id of IDOMFontPCL5.

Returns

CClassID class id of the element

getFontName()

```
virtual bool IDOMFontPCL5::getFontName (
            EDLSysString & fontName )  [pure virtual]
```

Returns the XL font name.

Parameters

fontName	The font name returned

Returns

bool True on success

getFontTrueTypeGlyphAccessor()

```
virtual bool IDOMFontPCL5::getFontTrueTypeGlyphAccessor (
            IFontPCL5TrueTypeGlyphAccessorPtr & accessor )  [pure virtual]
```

Creates a TrueType glyph accessor.

Parameters

accessor	Returns the accessor.

Returns

bool True on success.

getFormat15FontBlockEnumerator()

```
virtual bool IDOMFontPCL5::getFormat15FontBlockEnumerator (
            IFontPCL5WriteSegmentBlockEnumeratorPtr & enumerator )  [pure virtual]
```

Creates a Format 15 font block enumerator.

Parameters

enumerator	Returns the enumerator

Returns

bool True on success

getFormat16FontBlockEnumerator()

```
virtual bool IDOMFontPCL5::getFormat16FontBlockEnumerator (
            IFontPCL5WriteSegmentBlockEnumeratorPtr & enumerator )  [pure virtual]
```

Creates a Format 16 font block enumerator.

Parameters

enumerator	Returns the enumerator

Returns

bool True on success

getPCL5FontType()

```
virtual IDOMFontPCL5::ePCL5FontType IDOMFontPCL5::getPCL5FontType ( ) const  [pure virtual]
```

Returns the sub font type for this PCLXL font.

Returns

IDOMFontPCL5::ePCL5FontType Returns the sub-font type. Currently only TTF is supported.

getSymbolSetID()

```
virtual bool IDOMFontPCL5::getSymbolSetID (
            EDLSysString & symbolSetID ) const  [pure virtual]
```

Returns the font symbol set ID.

Parameters

symbolSetID	The font symbol set ID

Returns

bool True on success

getSymbolSetIDCode()

```
virtual IDOMFontPCL5::SymbolSetIDCode IDOMFontPCL5::getSymbolSetIDCode ( ) const  [pure virtual]
```

Returns the font symbol set ID as an ID Code.

Returns

IDOMFontPCL5::SymbolSet Returns the symbol set

setSymbolSetIDCode()

```
virtual bool IDOMFontPCL5::setSymbolSetIDCode (
            IDOMFontPCL5::SymbolSetIDCode symbolSet )  [pure virtual]
```

Set the font symbol set to be used in the generated font.

Parameters

symbolSet	The font symbol set

Returns

bool True on success

The documentation for this class was generated from the following file:

- idomfontpcl.h

7.190 IDOMFontPCLXL Class Reference

This class models PCL XL TrueType and bitmap fonts derived from an OpenType font source.

```
#include <idomfontpcl.h>
```

Inherits IDOMFontOpenTypeTT.

Classes

- class Data

  Initialization data.

Public Types

- enum ePCLXLFontType { ePCLXLFontTypeTTF , ePCLXLFontTypeBitmap }

  The enumeration used to identify the type of the PCL XL font.
- typedef uint16 **SymbolSet**

  Type represents the PCL XL Font Symbol Set.

Public Member Functions

- virtual IDOMFontPCLXL::ePCLXLFontType getPCLXLFontType () const =0

  Returns the sub font type for this PCL XL font.
- virtual bool getFontName (EDLSysString &fontName)=0

  Returns the XL font name. For example, "Times New Roman".
- virtual bool getFontHeaderSegmentBlockEnumerator (IFontHeaderWriteSegmentBlockEnumeratorPtr &enumerator)=0

  Creates a PCL XL FontHeader enumerator of the font for the PCL XL ReadFontHeader operator.
- virtual bool getFontTrueTypeGlyphAccessor (IFontPCLXLTrueTypeGlyphAccessorPtr &accessor)=0

  Creates a TrueType glyph accessor.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves the class id of IDOMFontPCLXL.

7.190.1 Detailed Description

This class models PCL XL TrueType and bitmap fonts derived from an OpenType font source.

7.190.2 Member Function Documentation

classID()

```
static const CClassID & IDOMFontPCLXL::classID ( )  [inline], [static]
```

Retrieves the class id of IDOMFontPCLXL.

Returns

CClassID The class id of the element

getFontHeaderSegmentBlockEnumerator()

```
virtual bool IDOMFontPCLXL::getFontHeaderSegmentBlockEnumerator (
            IFontHeaderWriteSegmentBlockEnumeratorPtr & enumerator )  [pure virtual]
```

Creates a PCL XL FontHeader enumerator of the font for the PCL XL ReadFontHeader operator.

Parameters

enumerator	Reference parameter to receive a smart pointer to the enumerator.

Returns

bool True on success, false if the call fails.

getFontName()

```
virtual bool IDOMFontPCLXL::getFontName (
            EDLSysString & fontName )  [pure virtual]
```

Returns the XL font name. For example, "Times New Roman".

Parameters

fontName	Reference parameter to receive a pointer to the font name.

Returns

bool True on success, false if the call fails.

getFontTrueTypeGlyphAccessor()

```
virtual bool IDOMFontPCLXL::getFontTrueTypeGlyphAccessor (
            IFontPCLXLTrueTypeGlyphAccessorPtr & accessor )  [pure virtual]
```

Creates a TrueType glyph accessor.

Parameters

accessor	A reference parameter to receive the glyph accessor.

Returns

bool True on success, false if the call fails.

getPCLXLFontType()

```
virtual IDOMFontPCLXL::ePCLXLFontType IDOMFontPCLXL::getPCLXLFontType ( ) const  [pure virtual]
```

Returns the sub font type for this PCL XL font.

Returns

IDOMFontPCLXL::ePCLXLFontType. Returns the sub-font type

The documentation for this class was generated from the following file:

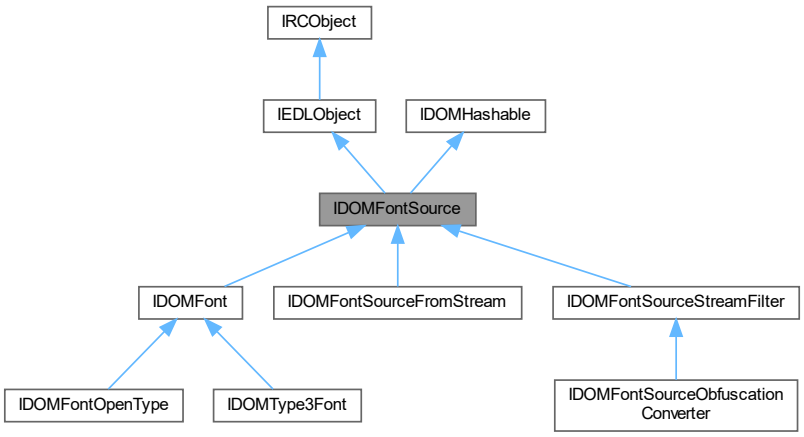
- idomfontpcl.h

7.191 IDOMFontSource Class Reference

The font source for the class IDOMFont. This class describes the different ways fonts are constructed.

`#include <idomfont.h>`

Inheritance diagram for IDOMFontSource:



Classes

- class Data

Public Types

- enum eFontSourceType { eFontSourceTypeNone , eFontSourceTypeStreamFilter , eFontSourceTypeStream , eFontSourceTypeFont }

  type used to uniquely identify the source type of a font

Public Member Functions

- virtual eFontSourceType getFontSourceType () const =0

  Gets the font source type.
- virtual const EDLSysString & determineUri () const =0

  Determines the URI based on the font source (underlying font sources may be searched)

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves the class id of IDOMFontSource.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.191.1 Detailed Description

The font source for the class IDOMFont. This class describes the different ways fonts are constructed.

7.191.2 Member Function Documentation

classID()

```
static const CClassID & IDOMFontSource::classID ( )  [inline], [static]
```

Retrieves the class id of IDOMFontSource.

Returns

CClassID The class id of the element

determineUri()

```
virtual const EDLSysString & IDOMFontSource::determineUri ( ) const  [pure virtual]
```

Determines the URI based on the font source (underlying font sources may be searched)

Returns

EDLSysString Returns the uri.

Implemented in IDOMFontSourceStreamFilter, IDOMFontSourceFromStream, and IDOMFontSourceObfuscationConverter.

getFontSourceType()

```
virtual eFontSourceType IDOMFontSource::getFontSourceType ( ) const  [pure virtual]
```

Gets the font source type.

Returns

eFontSourceType The font source type.

The documentation for this class was generated from the following file:

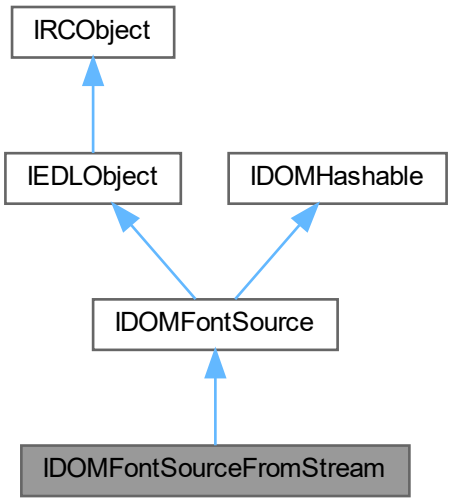
- idomfont.h

7.192 IDOMFontSourceFromStream Class Reference

The source for IDOMFont when sourced from an existing stream.

```
#include <idomfont.h>
```

Inheritance diagram for IDOMFontSourceFromStream:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual IInputStreamPtr getStream () const =0

  Return the actual stream.
- virtual int64 getStreamLength () const =0

  Retrieves the stream length, if possible.
- virtual const EDLSysString & determineUri () const =0

  Determines the URI based on the font source (underlying font sources may be searched)
- virtual bool getFromPcl () const =0

  Determine if this source was created from PCL/5 or PCL/XL.

Public Member Functions inherited from IDOMFontSource

- virtual eFontSourceType getFontSourceType () const =0

  Gets the font source type.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves the class id of IDOMFontSourceFromStream.

Static Public Member Functions inherited from IDOMFontSource

- static const CClassID & classID ()

  Retrieves the class id of IDOMFontSource.

Additional Inherited Members

Public Types inherited from IDOMFontSource

- enum eFontSourceType { eFontSourceTypeNone , eFontSourceTypeStreamFilter , eFontSourceTypeStream , eFontSourceTypeFont }

  type used to uniquely identify the source type of a font

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.192.1 Detailed Description

The source for IDOMFont when sourced from an existing stream.

7.192.2 Member Function Documentation

classID()

```
static const CClassID & IDOMFontSourceFromStream::classID ( )  [inline], [static]
```

Retrieves the class id of IDOMFontSourceFromStream.

Returns

CClassID Returns the class id of the element.

determineUri()

```
virtual const EDLSysString & IDOMFontSourceFromStream::determineUri ( ) const  [pure virtual]
```

Determines the URI based on the font source (underlying font sources may be searched)

Returns

EDLSysString Returns the uri.

Implements IDOMFontSource.

getFromPcl()

```
virtual bool IDOMFontSourceFromStream::getFromPcl ( ) const  [pure virtual]
```

Determine if this source was created from PCL/5 or PCL/XL.

Returns

bool True if created for PCL/5 or PCL/XL.

getStream()

```
virtual IInputStreamPtr IDOMFontSourceFromStream::getStream ( ) const  [pure virtual]
```

Return the actual stream.

Returns

IInputStreamPtr The stream

getStreamLength()

```
virtual int64 IDOMFontSourceFromStream::getStreamLength ( ) const  [pure virtual]
```

Retrieves the stream length, if possible.

Returns

int64 The stream length, or a negative length if it is not available

The documentation for this class was generated from the following file:

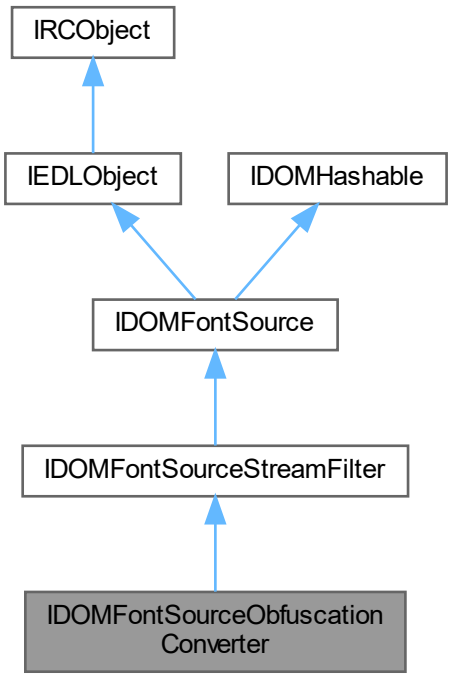
- idomfont.h

7.193 IDOMFontSourceObfuscationConverter Class Reference

Interface for a font sourced from a converter that performs obfuscation and deobfuscation.

`#include <idomfont.h>`

Inheritance diagram for IDOMFontSourceObfuscationConverter:



Classes

- class Data

  Initialization data.

Public Types

- enum eOperation { eObfuscate , eDeobfuscate }

  type used to uniquely identify the conversion direction

Public Types inherited from IDOMFontSourceStreamFilter

- enum eFontStreamFilterType { eFontStreamFilterTypeNone , eFontStreamFilterTypeObfuscation }

  An enumeration type used to identify the source type of a font.

Public Types inherited from IDOMFontSource

- enum eFontSourceType { eFontSourceTypeNone , eFontSourceTypeStreamFilter , eFontSourceTypeStream , eFontSourceTypeFont }

  type used to uniquely identify the source type of a font

Public Member Functions

- virtual IInputStreamPtr getStream () const =0

  Get the output stream of the processor.
- virtual const EDLSysString & determineUri () const =0

  Determines the URI based on the font source (underlying font sources may be searched)
- virtual IDOMFontSourcePtr getInputFontSource () const =0

  Get the underlying font source of this font stream processor.

Public Member Functions inherited from IDOMFontSourceStreamFilter

- virtual eFontStreamFilterType getFontStreamFilterType () const =0

  Get the font stream filter type.

Public Member Functions inherited from IDOMFontSource

- virtual eFontSourceType getFontSourceType () const =0

  Gets the font source type.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves the class id of IDOMFontSourceObfuscationConverter.

Static Public Member Functions inherited from IDOMFontSource

- static const CClassID & classID ()

  Retrieves the class id of IDOMFontSource.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.193.1 Detailed Description

Interface for a font sourced from a converter that performs obfuscation and deobfuscation.

7.193.2 Member Function Documentation

classID()

```
static const CClassID & IDOMFontSourceObfuscationConverter::classID ( )  [inline], [static]
```

Retrieves the class id of IDOMFontSourceObfuscationConverter.

Returns

CClassID The class id of the element

determineUri()

```
virtual const EDLSysString & IDOMFontSourceObfuscationConverter::determineUri ( ) const  [pure virtual]
```

Determines the URI based on the font source (underlying font sources may be searched)

Returns

EDLSysString. Returns the uri.

Implements IDOMFontSourceStreamFilter.

getInputFontSource()

```
virtual IDOMFontSourcePtr IDOMFontSourceObfuscationConverter::getInputFontSource ( ) const [pure virtual]
```

Get the underlying font source of this font stream processor.

Returns

IDOMFontSourcePtr The input font source

Implements IDOMFontSourceStreamFilter.

getStream()

```
virtual IInputStreamPtr IDOMFontSourceObfuscationConverter::getStream ( ) const  [pure virtual]
```

Get the output stream of the processor.

Returns

IInputStreamPtr The stream

Implements IDOMFontSourceStreamFilter.

The documentation for this class was generated from the following file:

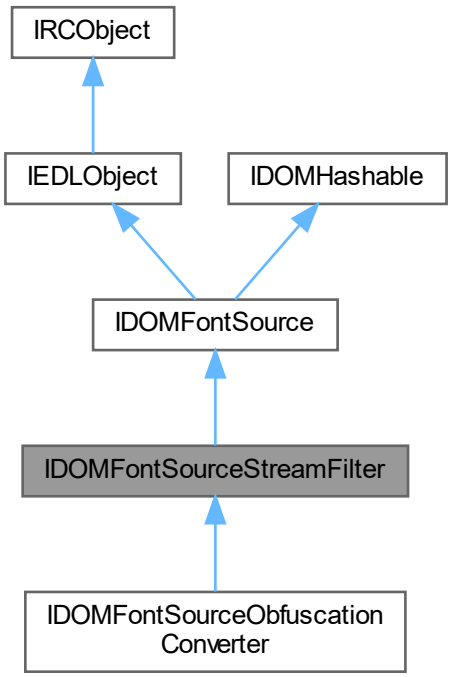
- idomfont.h

7.194 IDOMFontSourceStreamFilter Class Reference

An abstract interface for fonts sourced from a font stream filter.

`#include <idomfont.h>`

Inheritance diagram for IDOMFontSourceStreamFilter:



Classes

- class Data

  Initialization data.

Public Types

- enum eFontStreamFilterType { eFontStreamFilterTypeNone , eFontStreamFilterTypeObfuscation }

  An enumeration type used to identify the source type of a font.

Public Types inherited from IDOMFontSource

- enum eFontSourceType { eFontSourceTypeNone , eFontSourceTypeStreamFilter , eFontSourceTypeStream , eFontSourceTypeFont }

  type used to uniquely identify the source type of a font

Public Member Functions

- virtual eFontStreamFilterType getFontStreamFilterType () const =0

  Get the font stream filter type.
- virtual IInputStreamPtr getStream () const =0

  Retrieves the output stream of the processor.
- virtual const EDLSysString & determineUri () const =0

  Determines the URI based on the font source (underlying font sources may be searched)
- virtual IDOMFontSourcePtr getInputFontSource () const =0

  Get the input font source of this font stream processor.

Public Member Functions inherited from IDOMFontSource

- virtual eFontSourceType getFontSourceType () const =0

  Gets the font source type.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Additional Inherited Members

Static Public Member Functions inherited from IDOMFontSource

- static const CClassID & classID ()

  Retrieves the class id of IDOMFontSource.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()
  Virtual destructor.

7.194.1 Detailed Description

An abstract interface for fonts sourced from a font stream filter.

7.194.2 Member Function Documentation

determineUri()

```
virtual const EDLSysString & IDOMFontSourceStreamFilter::determineUri ( ) const  [pure virtual]
```

Determines the URI based on the font source (underlying font sources may be searched)

Returns

EDLSysString Returns the uri.

Implements IDOMFontSource.

Implemented in IDOMFontSourceObfuscationConverter.

getFontStreamFilterType()

```
virtual eFontStreamFilterType IDOMFontSourceStreamFilter::getFontStreamFilterType ( ) const
[pure virtual]
```

Get the font stream filter type.

Returns

eFontStreamFilterType The font stream filter type

getInputFontSource()

```
virtual IDOMFontSourcePtr IDOMFontSourceStreamFilter::getInputFontSource ( ) const  [pure
virtual]
```

Get the input font source of this font stream processor.

Returns

IDOMFontSourcePtr The source font source

Implemented in IDOMFontSourceObfuscationConverter.

getStream()

```
virtual IInputStreamPtr IDOMFontSourceStreamFilter::getStream ( ) const  [pure virtual]
```

Retrieves the output stream of the processor.

Returns

IInputStreamPtr The stream

Implemented in IDOMFontSourceObfuscationConverter.

The documentation for this class was generated from the following file:

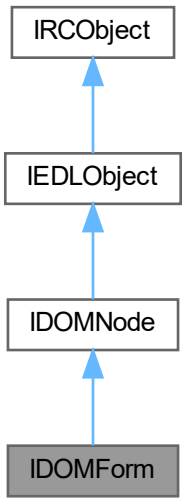
- idomfont.h

7.195 IDOMForm Class Reference

Interface to DOM node representing PDF-style Form XObjects.

```
#include <idomform.h>
```

Inheritance diagram for IDOMForm:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual DOMid getFormId () const =0

  Obtain the uique DOMid of this form.
- virtual const FMatrix & getMatrix () const =0

  Retrieves the Matrix of the form contents (equivalent to the /Matrix entry in a PDF/PS form dictionary).
- virtual void setMatrix (const FMatrix &matrix)=0

  Sets the Matrix of the form contents (equivalent to the /Matrix entry in a PDF/PS form dictionary).
- virtual void setBounds (const FRect &bounds)=0

  Sets the form bounding box.
- virtual JawsMako::IStructureElementReferencePtr getStructureElementReference () const =0

  Get the JawsMako Logical structure element reference associated with this form.
- virtual void setStructureElementReference (const JawsMako::IStructureElementReferencePtr &element↩ Ref)=0

  Set the JawsMako logical structure element reference to be associated with this form, or NULL to clear.
- virtual JawsMako::IPDFDictionaryPtr getPdfPropertiesDictionary () const =0

  Get the dictionary containing PDF properties attached to the form object. This dictionary will be as per the XObject dictionary in the original PDF, but with entries handled by existing DOM features stripped. In particular, the following entries described in the PDF specification will not be present in the dictionary:
- virtual void setPdfPropertiesDictionary (const JawsMako::IPDFDictionaryPtr &propertiesDictionary)=0

  Set the dictionary containing PDF properties attached to the form object. Please see getPdfPropertiesDictionary() above for information on the form of this dictionary.

Public Member Functions inherited from IDOMNode

- virtual ~**IDOMNode** ()

  virtual destructor
- virtual DOMid getDOMid () const =0

  Retrieves the node ID.
- virtual void setDOMid (DOMid id)=0

  Sets the node ID.
- virtual eDOMNodeType getNodeType () const =0

  Retrieves the DOM node type.
- virtual bool getProperty (const EDLSysString &propertyName, PValue &propertyValue) const =0

  Retrieves the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.
- virtual void setProperty (const EDLSysString &propertyName, const PValue &propertyValue)=0

  Sets the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.
- virtual void removeProperty (const EDLSysString &propertyName)=0

  Removes property.
- virtual IEDLSysStringCollectionEnumPtr getPropertyCollectionEnum ()=0

  Retrieves a navigable list of the property names stored on this node.
- virtual bool hasChildNodes () const =0

  Function that indicates whether this node is a parent to other nodes.
- virtual IDOMNodePtr getParentNode () const =0

  Gets the parent node of this node.
- virtual IDOMNodePtr getFirstChild () const =0

  Gets the first child node of this node.
- virtual IDOMNodePtr getLastChild () const =0

Gets the last child node of this node.

- virtual IDOMNodePtr getNextChild (const IDOMNodePtr &child) const =0

  Gets the child node which follows the node passed in.
- virtual IDOMNodePtr getPreviousChild (const IDOMNodePtr &child) const =0

  Gets the child node which precedes the node passed in.
- virtual IDOMNodePtr getPreviousSibling () const =0

  Retrieves the node's previous sibling node.
- virtual IDOMNodePtr getNextSibling () const =0

  Retrieves node's next sibling node.
- virtual void appendChild (const IDOMNodePtr &child)=0

  Appends a node to the end of the node's child list.
- virtual void insertChild (const IDOMNodePtr &ptrPreviousSibling, const IDOMNodePtr &child, bool bCheck↩ Complete=true)=0

  Insert a child node after ptrPreviousSibling.
- virtual IDOMNodePtr extractChild (const IDOMNodePtr &child)=0

  Extracts (that is, finds and removes) a child node from the node children. After extraction the child node is no longer a part of the DOM. If no node is specified, the first available node will be extracted from the node's children.
- virtual void replaceChild (const IDOMNodePtr &oldChild, const IDOMNodePtr &newChild)=0

  Replaces the child node with another.
- virtual bool isComplete () const =0

  Signals the completeness of the node.
  A complete node is one that has no more children to be added to it.
- virtual void **setComplete** ()=0

  Sets the node's completeness status to "true".
- virtual IDOMNodeFlags ∗ getFlags ()=0

  Retrieves the node's flags property.
- virtual void setParentNode (const IDOMNodePtr &ptrParent)=0

  Sets the parent node.
- virtual void setPreviousSibling (const IDOMNodePtr &ptrPreviousSibling)=0

  Sets the previous sibling node.
- virtual void setNextSibling (const IDOMNodePtr &ptrNextSibling)=0

  Sets the next sibling node.
- virtual bool isAncestor (const IDOMNodePtr &ptrCandidate)=0

  Function tests whether a candidate node is a descendant of the node.
- virtual FRect getBounds (bool applyTransform=true, bool applyClip=true)

  Find the conservative bounding box of the marking content of the node.
- virtual bool copyNodeData (IDOMNode ∗pSourceNode)=0

  Copy the properties collection, the flags and the DOM ID from the given source node to this one.
- virtual IDOMNodePtr cloneNode (IEDLClassFactory ∗pFactory) const =0

  Simpified node cloning. An exception of type IEDLError will be thrown on failure.
- virtual IDOMNodePtr cloneTree (IEDLClassFactory ∗pFactory) const =0

  Clone the tree of nodes beginning at this node. An exception of type IEDLError will be thrown on failure.
- virtual void cloneTreeAndAppend (IEDLClassFactory ∗pFactory, const IDOMNodePtr &dest) const =0

  Clone the tree of nodes beginning at this node, and append the result to the destination tree.
- virtual void **completeTree** ()=0

  Mark the entire tree from this point as complete. You should not ordinarily need to call this function.
- virtual void **removeCompleteFlagFromTree** ()=0

  Mark the entire tree from this point as complete.
- virtual void findChildrenOfType (eDOMNodeType type, CDOMNodeVect &nodes, bool searchForms=false)=0

  Find all children of this node with the given type, appending to the given vector. Does not descend into brushes.

- virtual void walkTree (WalkTreeFunc func, void *priv, bool descendIntoBrushes=false, bool descendInto↩ Forms=false)=0

  Walk through the DOM calling a given function on each node. The function is allowed to:
- virtual void notifyOnDestruct (NodeDeleteFunc func, void *priv)=0

  Register interest in being told when this node is about to be destroyed.
- virtual void unregisterNotify (NodeDeleteFunc func, void *priv)=0

  Unregister interest in being told when this node is about to be destroyed.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static EDL_API IDOMFormPtr create (IEDLClassFactory *pFactory, const FMatrix &matrix=FMatrix(), const FRect &bounds=FRect(), const JawsMako::IPDFDictionaryPtr &propertiesDictionary=JawsMako↩ ::IPDFDictionaryPtr())

  Simplified creation function for IDOMForm. Throws an IEDLError exception on failure.
- static const CClassID & classID ()

  Retrieves class id of IDOMForm.

Static Public Member Functions inherited from IDOMNode

- static EDL_API FMatrix effectiveTransformationOfNode (const IDOMNodePtr &node)

  Attempt to find the effective transformation matrix external to the specified node relative to either a containing page or ultimate parent.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.195.1 Detailed Description

Interface to DOM node representing PDF-style Form XObjects.

7.195.2 Member Function Documentation

classID()

```
static const CClassID & IDOMForm::classID ( )  [inline], [static]
```

Retrieves class id of IDOMForm.

Returns

CClassID class id of the element

create()

```
static EDL_API IDOMFormPtr IDOMForm::create (
            IEDLClassFactory * pFactory,
            const FMatrix & matrix = FMatrix(),
            const FRect & bounds = FRect(),
            const JawsMako::IPDFDictionaryPtr & propertiesDictionary = JawsMako::IPDFDictionaryPtr()
)  [static]
```

Simplified creation function for IDOMForm. Throws an IEDLError exception on failure.

Parameters

pFactory	The EDL class factory
matrix	The desired form matrix
bounds	The desired form bounds
propertiesDictionary	Any additional PDF properties

Returns

IDOMFormPtr A smart pointer to the IDOMForm

getFormId()

```
virtual DOMid IDOMForm::getFormId ( ) const  [pure virtual]
```

Obtain the uique DOMid of this form.

```
This is distinct from that received from IDOMNode::getDOMid() which
is user set and is copied when the node is cloned.

This DOMid is unique to the form object, and is allocated when
the form is allocated. Forms cloned from this object will receive
a new DOMid.
```

Returns

DOMid The DOMid

getMatrix()

```
virtual const FMatrix & IDOMForm::getMatrix ( ) const  [pure virtual]
```

Retrieves the Matrix of the form contents (equivalent to the /Matrix entry in a PDF/PS form dictionary).

Returns

FMatrix The matrix

getPdfPropertiesDictionary()

```
virtual JawsMako::IPDFDictionaryPtr IDOMForm::getPdfPropertiesDictionary ( ) const  [pure
virtual]
```

Get the dictionary containing PDF properties attached to the form object. This dictionary will be as per the XObject dictionary in the original PDF, but with entries handled by existing DOM features stripped. In particular, the following entries described in the PDF specification will not be present in the dictionary:

- Type
- Subtype
- FormType
- BBox
- Matrix
- Resources
- Group
- StructParent
- StructParents
- OC
- Name

Returns

JawsMako::IPDFDictionary The properties dictionary, or NULL if not present. If non-null, the dictionary may be edited freely.

getStructureElementReference()

```
virtual JawsMako::IStructureElementReferencePtr IDOMForm::getStructureElementReference ( )
const  [pure virtual]
```

Get the JawsMako Logical structure element reference associated with this form.

Returns

JawsMako::IStructureElementReferencePtr A smart pointer to the structure element reference

setBounds()

```
virtual void IDOMForm::setBounds (
            const FRect & bounds )  [pure virtual]
```

Sets the form bounding box.

Parameters

bounds	Reference parameter to receive the bounding box.

setMatrix()

```
virtual void IDOMForm::setMatrix (
            const FMatrix & matrix )  [pure virtual]
```

Sets the Matrix of the form contents (equivalent to the /Matrix entry in a PDF/PS form dictionary).

Parameters

matrix	Reference to receive the matrix

setPdfPropertiesDictionary()

```
virtual void IDOMForm::setPdfPropertiesDictionary (
            const JawsMako::IPDFDictionaryPtr & propertiesDictionary )  [pure virtual]
```

Set the dictionary containing PDF properties attached to the form object. Please see getPdfPropertiesDictionary() above for information on the form of this dictionary.

Parameters

propertiesDictionary	The properties dictionary to set, or NULL to clear.

setStructureElementReference()

```
virtual void IDOMForm::setStructureElementReference (
            const JawsMako::IStructureElementReferencePtr & elementRef )  [pure virtual]
```

Set the JawsMako logical structure element reference to be associated with this form, or NULL to clear.

Parameters

elementRef	The structure element reference to set

The documentation for this class was generated from the following file:

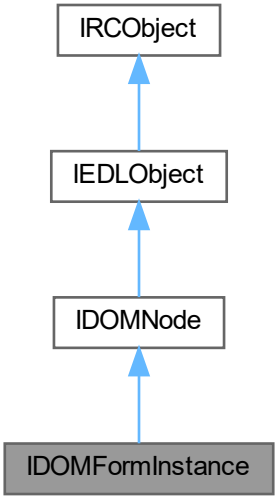
- idomform.h

7.196 IDOMFormInstance Class Reference

IDOMFormInstance interface. This describes an instance of an IDOMForm in a DOM tree.

`#include <idomform.h>`

Inheritance diagram for IDOMFormInstance:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual float getOpacity () const =0

  Get the form alpha/opacity.
- virtual void setOpacity (float opacity)=0

  Set the form alpha/opacity.
- virtual const FMatrix & getRenderTransform () const =0

  Retrieves render transform applied to this instance of the form.
- virtual void setRenderTransform (const FMatrix &renderTransform)=0

  Retrieves render transform applied to this instance of the form.
- virtual IDOMFormPtr getForm () const =0

  Retrieves the Form associated with this instance.
- virtual void setForm (const IDOMFormPtr &form)=0

Sets the Form associated with this instance.

- virtual eBlendMode getBlendMode () const =0

  Get the blend mode to be used for compositing this form.
- virtual void setBlendMode (eBlendMode blendMode)=0

  Set the blend mode to be used for compositing this form.
- virtual IDOMBrushPtr getOpacityMask () const =0

  Retrieves smart pointer to opacity mask.
- virtual void setOpacityMask (const IDOMBrushPtr &ptrOpacityMask)=0

  Sets the opacity mask.

Public Member Functions inherited from IDOMNode

- virtual ~**IDOMNode** ()

  virtual destructor
- virtual DOMid getDOMid () const =0

  Retrieves the node ID.
- virtual void setDOMid (DOMid id)=0

  Sets the node ID.
- virtual eDOMNodeType getNodeType () const =0

  Retrieves the DOM node type.
- virtual bool getProperty (const EDLSysString &propertyName, PValue &propertyValue) const =0

  Retrieves the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.
- virtual void setProperty (const EDLSysString &propertyName, const PValue &propertyValue)=0

  Sets the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.
- virtual void removeProperty (const EDLSysString &propertyName)=0

  Removes property.
- virtual IEDLSysStringCollectionEnumPtr getPropertyCollectionEnum ()=0

  Retrieves a navigable list of the property names stored on this node.
- virtual bool hasChildNodes () const =0

  Function that indicates whether this node is a parent to other nodes.
- virtual IDOMNodePtr getParentNode () const =0

  Gets the parent node of this node.
- virtual IDOMNodePtr getFirstChild () const =0

  Gets the first child node of this node.
- virtual IDOMNodePtr getLastChild () const =0

  Gets the last child node of this node.
- virtual IDOMNodePtr getNextChild (const IDOMNodePtr &child) const =0

  Gets the child node which follows the node passed in.
- virtual IDOMNodePtr getPreviousChild (const IDOMNodePtr &child) const =0

  Gets the child node which precedes the node passed in.
- virtual IDOMNodePtr getPreviousSibling () const =0

  Retrieves the node's previous sibling node.
- virtual IDOMNodePtr getNextSibling () const =0

  Retrieves node's next sibling node.
- virtual void appendChild (const IDOMNodePtr &child)=0

  Appends a node to the end of the node's child list.

- virtual void insertChild (const IDOMNodePtr &ptrPreviousSibling, const IDOMNodePtr &child, bool bCheck↩ Complete=true)=0

  Insert a child node after ptrPreviousSibling.
- virtual IDOMNodePtr extractChild (const IDOMNodePtr &child)=0

  Extracts (that is, finds and removes) a child node from the node children. After extraction the child node is no longer a part of the DOM. If no node is specified, the first available node will be extracted from the node's children.
- virtual void replaceChild (const IDOMNodePtr &oldChild, const IDOMNodePtr &newChild)=0

  Replaces the child node with another.
- virtual bool isComplete () const =0

  Signals the completeness of the node.
  A complete node is one that has no more children to be added to it.
- virtual void **setComplete** ()=0

  Sets the node's completeness status to "true".
- virtual IDOMNodeFlags ∗ getFlags ()=0

  Retrieves the node's flags property.
- virtual void setParentNode (const IDOMNodePtr &ptrParent)=0

  Sets the parent node.
- virtual void setPreviousSibling (const IDOMNodePtr &ptrPreviousSibling)=0

  Sets the previous sibling node.
- virtual void setNextSibling (const IDOMNodePtr &ptrNextSibling)=0

  Sets the next sibling node.
- virtual bool isAncestor (const IDOMNodePtr &ptrCandidate)=0

  Function tests whether a candidate node is a descendant of the node.
- virtual FRect getBounds (bool applyTransform=true, bool applyClip=true)

  Find the conservative bounding box of the marking content of the node.
- virtual bool copyNodeData (IDOMNode ∗pSourceNode)=0

  Copy the properties collection, the flags and the DOM ID from the given source node to this one.
- virtual IDOMNodePtr cloneNode (IEDLClassFactory ∗pFactory) const =0

  Simpified node cloning. An exception of type IEDLError will be thrown on failure.
- virtual IDOMNodePtr cloneTree (IEDLClassFactory ∗pFactory) const =0

  Clone the tree of nodes beginning at this node. An exception of type IEDLError will be thrown on failure.
- virtual void cloneTreeAndAppend (IEDLClassFactory ∗pFactory, const IDOMNodePtr &dest) const =0

  Clone the tree of nodes beginning at this node, and append the result to the destination tree.
- virtual void **completeTree** ()=0

  Mark the entire tree from this point as complete. You should not ordinarily need to call this function.
- virtual void **removeCompleteFlagFromTree** ()=0

  Mark the entire tree from this point as complete.
- virtual void findChildrenOfType (eDOMNodeType type, CDOMNodeVect &nodes, bool searchForms=false)=0

  Find all children of this node with the given type, appending to the given vector. Does not descend into brushes.
- virtual void walkTree (WalkTreeFunc func, void ∗priv, bool descendIntoBrushes=false, bool descendInto↩ Forms=false)=0

  Walk through the DOM calling a given function on each node. The function is allowed to:
- virtual void notifyOnDestruct (NodeDeleteFunc func, void ∗priv)=0

  Register interest in being told when this node is about to be destroyed.
- virtual void unregisterNotify (NodeDeleteFunc func, void ∗priv)=0

  Unregister interest in being told when this node is about to be destroyed.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static EDL_API IDOMFormInstancePtr create (IEDLClassFactory *pFactory, const IDOMFormPtr &form, const FMatrix &renderTransform=FMatrix())

  Simplified creation function for IDOMFormInstance. Throws an IEDLError exception on failure.
- static const CClassID & classID ()

  Retrieves class id of IDOMForm.

Static Public Member Functions inherited from IDOMNode

- static EDL_API FMatrix effectiveTransformationOfNode (const IDOMNodePtr &node)

  Attempt to find the effective transformation matrix external to the specified node relative to either a containing page or ultimate parent.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.196.1 Detailed Description

IDOMFormInstance interface. This describes an instance of an IDOMForm in a DOM tree.

7.196.2 Member Function Documentation

classID()

```
static const CClassID & IDOMFormInstance::classID ( )  [inline], [static]
```

Retrieves class id of IDOMForm.

Returns

CClassID class id of the element

create()

```
static EDL_API IDOMFormInstancePtr IDOMFormInstance::create (
            IEDLClassFactory * pFactory,
            const IDOMFormPtr & form,
            const FMatrix & renderTransform = FMatrix() )  [static]
```

Simplified creation function for IDOMFormInstance. Throws an IEDLError exception on failure.

Parameters

pFactory	The EDL class factory.
form	The form that this instance will use.
renderTransform	The desired render transform.

Returns

IDOMFormInstancePtr A smart pointer to the IDOMForm.

getBlendMode()

```
virtual eBlendMode IDOMFormInstance::getBlendMode ( ) const  [pure virtual]
```

Get the blend mode to be used for compositing this form.

Returns

eBlendMode The blend mode for this form

getForm()

```
virtual IDOMFormPtr IDOMFormInstance::getForm ( ) const  [pure virtual]
```

Retrieves the Form associated with this instance.

Returns

IDOMFormPtr The form or NULL if no form is present in the instance.

getOpacity()

```
virtual float IDOMFormInstance::getOpacity ( ) const  [pure virtual]
```

Get the form alpha/opacity.

Returns

float The opacity.

getOpacityMask()

```
virtual IDOMBrushPtr IDOMFormInstance::getOpacityMask ( ) const  [pure virtual]
```

Retrieves smart pointer to opacity mask.

Returns

IDOMBrushPtr The opacity mask for this form or NULL if no opacity mask is present.

getRenderTransform()

```
virtual const FMatrix & IDOMFormInstance::getRenderTransform ( ) const  [pure virtual]
```

Retrieves render transform applied to this instance of the form.

Returns

FMatrix The render transform

setBlendMode()

```
virtual void IDOMFormInstance::setBlendMode (
            eBlendMode blendMode )  [pure virtual]
```

Set the blend mode to be used for compositing this form.

Parameters

blendMode	The desired blend mode

setForm()

```
virtual void IDOMFormInstance::setForm (
            const IDOMFormPtr & form )  [pure virtual]
```

Sets the Form associated with this instance.

Parameters

form	Reference to the desired form.

setOpacity()

```
virtual void IDOMFormInstance::setOpacity (
            float opacity )  [pure virtual]
```

Set the form alpha/opacity.

Parameters

opacity	The desired opacity, a value between 0.0 and 1.0

setOpacityMask()

```
virtual void IDOMFormInstance::setOpacityMask (
            const IDOMBrushPtr & ptrOpacityMask )  [pure virtual]
```

Sets the opacity mask.

Parameters

ptrOpacityMask	Smart pointer to desired opacity mask

setRenderTransform()

```
virtual void IDOMFormInstance::setRenderTransform (
            const FMatrix & renderTransform )  [pure virtual]
```

Retrieves render transform applied to this instance of the form.

Parameters

renderTransform	Reference to the desired transform

The documentation for this class was generated from the following file:

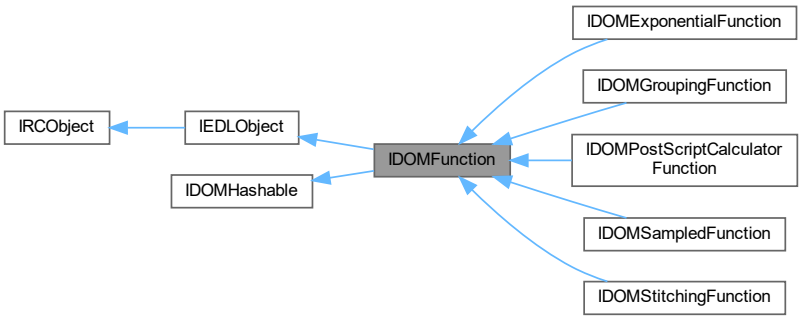
- idomform.h

7.197 IDOMFunction Class Reference

Base class for PDF/PS Style functions All function instances throw IEDLError exceptions on failure.

`#include <idomfunction.h>`

Inheritance diagram for IDOMFunction:



Public Types

- enum eFunctionType

  An enum for the various types of functions.

Public Member Functions

- virtual eFunctionType getFunctionType () const =0

  Retrieves function type.
- virtual uint32 getNumInputValues () const =0

  Get the number of input values that this function will operate on.
- virtual uint32 getNumOutputValues () const =0

  Get the number of output values that this function will produce.
- virtual void getInputDomain (uint32 inputNum, float &low, float &high) const =0

  Get the input domain for a given input to the function.
- virtual bool getOutputRange (uint32 outputNum, float &low, float &high) const =0

  Get the output range for a given input to the function.
- virtual void evaluate (const float ∗inputValues, float ∗outputValues) const =0

  Evaluate the input through the function and return the result.
- virtual void evaluate (const int32 ∗inputValues, float ∗outputValues) const =0

  Evaluate the input through the function and return the result.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams ∗pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory ∗pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.197.1 Detailed Description

Base class for PDF/PS Style functions All function instances throw IEDLError exceptions on failure.

7.197.2 Member Function Documentation

evaluate() [1/2]

```
virtual void IDOMFunction::evaluate (
            const float * inputValues,
            float * outputValues ) const  [pure virtual]
```

Evaluate the input through the function and return the result.

Parameters

inputValues	An array of floats that are input into the function. The size of the array must be the same as the required number of inputValues
outputValues	An array of floats that are the result of evaluating the input through the function. The size of the array must be the same as the required number of inputValues

evaluate() [2/2]

```
virtual void IDOMFunction::evaluate (
            const int32 * inputValues,
            float * outputValues ) const  [pure virtual]
```

Evaluate the input through the function and return the result.

Parameters

inputValues	An array of integers that are input into the function. The size of the array must be the same as the required number of inputValues
outputValues	An array of floats that are the result of evaluating the input through the function. The size of the array must be the same as the required number of inputValues

getFunctionType()

```
virtual eFunctionType IDOMFunction::getFunctionType ( ) const  [pure virtual]
```

Retrieves function type.

Returns

eFunctionType The function type

getInputDomain()

```
virtual void IDOMFunction::getInputDomain (
            uint32 inputNum,
            float & low,
            float & high ) const  [pure virtual]
```

Get the input domain for a given input to the function.

Parameters

inputNum	The 0-indexed input number
low	A reference to receive the low domain bound for the given inputNum
high	A reference to receive the high domain bound for the given inputNum

getNumInputValues()

```
virtual uint32 IDOMFunction::getNumInputValues ( ) const  [pure virtual]
```

Get the number of input values that this function will operate on.

Returns

uint32 The number of input values.

getNumOutputValues()

```
virtual uint32 IDOMFunction::getNumOutputValues ( ) const  [pure virtual]
```

Get the number of output values that this function will produce.

Returns

uint32 The number of output values.

getOutputRange()

```
virtual bool IDOMFunction::getOutputRange (
            uint32 outputNum,
            float & low,
            float & high ) const  [pure virtual]
```

Get the output range for a given input to the function.

Parameters

outputNum	The 0-indexed output number
low	A reference to receive the low range bound for the given outputNum
high	A reference to receive the high range bound for the given outputNum

Returns

bool Returns true on success. False if there is no output range (which is not required for all function types).

The documentation for this class was generated from the following file:

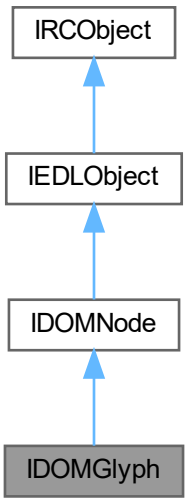
- idomfunction.h

7.198 IDOMGlyph Class Reference

The IDOMGlyph class is an abstract class modelling a single character from a font.

```
#include <idomglyph.h>
```

Inheritance diagram for IDOMGlyph:



Classes

- class Data

  Initialization data.

Public Types

- enum eGlyphIDSpecial { eGlyphIDNotdef = 0 }

  Enumeration type used to define known, special Glyph IDs.
- typedef uint16 GlyphID

  Holds the glyph ID of the glyph.

Public Member Functions

- virtual bool getHasCustomAdvance () const =0

  Determines whether or not a custom value has been set for the glyph's advance.
- virtual void setHasCustomAdvance (bool bHasCustomAdvance)=0

  Informs whether a custom value has been set for the glyph's advance.
- virtual GlyphID getGlyphID () const =0

  Retrieves the glyph id for this glyph.
- virtual bool getGlyphUnicode (uint32 &codePoint) const =0

  Retrieves the Unicode code point for the glyph, if the glyph has Unicode and the Unicode is a single code point.
- virtual const EDLU32String & getGlyphUnicode () const =0

  Retrieves the Unicode for the glyph, or an empty string.
- virtual bool getGlyphName (EDLSysString &glyphName)=0

  Retrieves the glyph name for this glyph, if present. An empty string is still a valid glyph name.

- virtual IDOMGlyphNamePtr getGlyphName () const =0

  Retrieves the glyph name for this glyph, if present, as a IDOMGlyphNamePtr. Returns NULL if the glyph has no name.
- virtual double getAdvanceX () const =0

  Retrieves the x-component of the advance of the glyph. The advance indicates the positioning of the next glyph in the sequence, relative to the origin of the current glyph. Base glyphs generally have a non-zero advance and combining glyphs have a zero advance.
- virtual double getAdvanceY () const =0

  Retrieves the y-component of the advance of the glyph. The advance indicates the positioning of the next glyph in the sequence, relative to the origin of the current glyph. Base glyphs generally have a non-zero advance and combining glyphs have a zero advance.
- virtual double getUOffset () const =0

  Retrieves the UOffset value for the glyph.
- virtual double getVOffset () const =0

  Retrieves the VOffset value for the glyph.
- virtual FRect getBounds () const =0

  Retrieves the glyph bounding box.
- virtual void setColored (bool bColored)=0

  Sets a flag whereby the glyph specifies its own colors or not.
- virtual bool getColored () const =0

  Retrieves the flag which indicates if this glyph contains color information.
- virtual double getOriginalAdvanceX () const =0

  Retrieves the x-component of the original advance of the glyph.

Public Member Functions inherited from IDOMNode

- virtual ~**IDOMNode** ()

  virtual destructor
- virtual DOMid getDOMid () const =0

  Retrieves the node ID.
- virtual void setDOMid (DOMid id)=0

  Sets the node ID.
- virtual eDOMNodeType getNodeType () const =0

  Retrieves the DOM node type.
- virtual bool getProperty (const EDLSysString &propertyName, PValue &propertyValue) const =0

  Retrieves the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.
- virtual void setProperty (const EDLSysString &propertyName, const PValue &propertyValue)=0

  Sets the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.
- virtual void removeProperty (const EDLSysString &propertyName)=0

  Removes property.
- virtual IEDLSysStringCollectionEnumPtr getPropertyCollectionEnum ()=0

  Retrieves a navigable list of the property names stored on this node.
- virtual bool hasChildNodes () const =0

  Function that indicates whether this node is a parent to other nodes.
- virtual IDOMNodePtr getParentNode () const =0

  Gets the parent node of this node.
- virtual IDOMNodePtr getFirstChild () const =0

  Gets the first child node of this node.
- virtual IDOMNodePtr getLastChild () const =0

Gets the last child node of this node.

- virtual IDOMNodePtr getNextChild (const IDOMNodePtr &child) const =0

  Gets the child node which follows the node passed in.
- virtual IDOMNodePtr getPreviousChild (const IDOMNodePtr &child) const =0

  Gets the child node which precedes the node passed in.
- virtual IDOMNodePtr getPreviousSibling () const =0

  Retrieves the node's previous sibling node.
- virtual IDOMNodePtr getNextSibling () const =0

  Retrieves node's next sibling node.
- virtual void appendChild (const IDOMNodePtr &child)=0

  Appends a node to the end of the node's child list.
- virtual void insertChild (const IDOMNodePtr &ptrPreviousSibling, const IDOMNodePtr &child, bool bCheck↩ Complete=true)=0

  Insert a child node after ptrPreviousSibling.
- virtual IDOMNodePtr extractChild (const IDOMNodePtr &child)=0

  Extracts (that is, finds and removes) a child node from the node children. After extraction the child node is no longer a part of the DOM. If no node is specified, the first available node will be extracted from the node's children.
- virtual void replaceChild (const IDOMNodePtr &oldChild, const IDOMNodePtr &newChild)=0

  Replaces the child node with another.
- virtual bool isComplete () const =0

  Signals the completeness of the node.
  A complete node is one that has no more children to be added to it.
- virtual void **setComplete** ()=0

  Sets the node's completeness status to "true".
- virtual IDOMNodeFlags ∗ getFlags ()=0

  Retrieves the node's flags property.
- virtual void setParentNode (const IDOMNodePtr &ptrParent)=0

  Sets the parent node.
- virtual void setPreviousSibling (const IDOMNodePtr &ptrPreviousSibling)=0

  Sets the previous sibling node.
- virtual void setNextSibling (const IDOMNodePtr &ptrNextSibling)=0

  Sets the next sibling node.
- virtual bool isAncestor (const IDOMNodePtr &ptrCandidate)=0

  Function tests whether a candidate node is a descendant of the node.
- virtual FRect getBounds (bool applyTransform=true, bool applyClip=true)

  Find the conservative bounding box of the marking content of the node.
- virtual bool copyNodeData (IDOMNode ∗pSourceNode)=0

  Copy the properties collection, the flags and the DOM ID from the given source node to this one.
- virtual IDOMNodePtr cloneNode (IEDLClassFactory ∗pFactory) const =0

  Simpified node cloning. An exception of type IEDLError will be thrown on failure.
- virtual IDOMNodePtr cloneTree (IEDLClassFactory ∗pFactory) const =0

  Clone the tree of nodes beginning at this node. An exception of type IEDLError will be thrown on failure.
- virtual void cloneTreeAndAppend (IEDLClassFactory ∗pFactory, const IDOMNodePtr &dest) const =0

  Clone the tree of nodes beginning at this node, and append the result to the destination tree.
- virtual void **completeTree** ()=0

  Mark the entire tree from this point as complete. You should not ordinarily need to call this function.
- virtual void **removeCompleteFlagFromTree** ()=0

  Mark the entire tree from this point as complete.
- virtual void findChildrenOfType (eDOMNodeType type, CDOMNodeVect &nodes, bool searchForms=false)=0

  Find all children of this node with the given type, appending to the given vector. Does not descend into brushes.

- virtual void walkTree (WalkTreeFunc func, void *priv, bool descendIntoBrushes=false, bool descendInto↩ Forms=false)=0

  Walk through the DOM calling a given function on each node. The function is allowed to:
- virtual void notifyOnDestruct (NodeDeleteFunc func, void *priv)=0

  Register interest in being told when this node is about to be destroyed.
- virtual void unregisterNotify (NodeDeleteFunc func, void *priv)=0

  Unregister interest in being told when this node is about to be destroyed.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves the class id for IDOMGlyph.
- static EDL_API IDOMGlyphPtr create (IEDLClassFactory *pFactory, const IDOMGlyph::Data &params)

  Simplified creator for a glyph object Throws an IEDLError on failure.
- static EDL_API IDOMGlyphPtr create (IEDLClassFactory *pFactory, bool hasCustomAdvance, double advanceX, double advanceY, double uOffset, double vOffset, const EDLU32String &unicode, IDOMGlyph::GlyphID glyphID, const IDOMGlyphNamePtr &glyphName, const FRect bounds, bool colored, double originalAdvanceX)

  Simplified creator for a glyph object Throws an IEDLError on failure.

Static Public Member Functions inherited from IDOMNode

- static EDL_API FMatrix effectiveTransformationOfNode (const IDOMNodePtr &node)

  Attempt to find the effective transformation matrix external to the specified node relative to either a containing page or ultimate parent.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()
  Virtual destructor.

7.198.1 Detailed Description

The IDOMGlyph class is an abstract class modelling a single character from a font.

All fonts contain glyphs. TrueType fonts describe each glyph as a set of paths. A path is a closed curve specified using points and particular mathematics. A lower case 'i' has two paths, one for the dot and one for the stem. The paths are filled with pixels to create the final letter form. This set of paths is called an outline.

In general, glyphs within a text run are either base glyphs or combining marks that may be attached to base glyphs. A glyph describes a single character. The IDOMGlyphs node describes a run of characters, each of the characters in the run is described by an IDOMGlyph node.

Another way that a glyph may be specified is in terms of other component glyphs. A glyph may consist of references to other glyphs which are combined to make the new compound glyph. An 'acute e'(é) could be composed of the glyphs for 'e' and 'acute'. In such compound glyphs, each component glyph has placement and optional transformation data associated with it.

All measurements are relative to the font rendering em size, which is considered to be 1.

If children are present, they are the marking nodes that represent the glyph

7.198.2 Member Function Documentation

classID()

```
static const CClassID & IDOMGlyph::classID ( )  [inline], [static]
```

Retrieves the class id for IDOMGlyph.

Returns

CClassID The class id of IDOMGlyph.

create() [1/2]

```
static EDL_API IDOMGlyphPtr IDOMGlyph::create (
            IEDLClassFactory * pFactory,
            bool hasCustomAdvance,
            double advanceX,
            double advanceY,
            double uOffset,
            double vOffset,
            const EDLU32String & unicode,
            IDOMGlyph::GlyphID glyphID,
            const IDOMGlyphNamePtr & glyphName,
            const FRect bounds,
            bool colored,
            double originalAdvanceX )  [static]
```

Simplified creator for a glyph object Throws an IEDLError on failure.

Parameters

pFactory	The factory to use.
hasCustomAdvance	Whether or not a custom value has been set for the glyph's advance.
advanceX	The x-component of the advance of the glyph.
advanceY	The y-component of the advance of the glyph.
uOffset	The UOffset value for the glyph.
vOffset	The VOffset value for the glyph.
unicode	The Unicode for the glyph, or an empty string.
glyphName	The glyph name, or an empty string.
bounds	The glyph bounding box.
colored	Indicates if this glyph contains color information.
originalAdvanceX	The x-component of the original advance of the glyph.

Returns

IDOMGlyphPtr The new glyph object.

create() [2/2]

```
static EDL_API IDOMGlyphPtr IDOMGlyph::create (
            IEDLClassFactory * pFactory,
            const IDOMGlyph::Data & params )  [static]
```

Simplified creator for a glyph object Throws an IEDLError on failure.

Parameters

pFactory	The factory to use.
params	Reference to initialization data.

Returns

IDOMGlyphPtr The new glyph object.

getAdvanceX()

```
virtual double IDOMGlyph::getAdvanceX ( ) const  [pure virtual]
```

Retrieves the x-component of the advance of the glyph. The advance indicates the positioning of the next glyph in the sequence, relative to the origin of the current glyph. Base glyphs generally have a non-zero advance and combining glyphs have a zero advance.

Advance width is measured in ems.

Returns

double The value of the x-component of the advance of the glyph

getAdvanceY()

```
virtual double IDOMGlyph::getAdvanceY ( ) const  [pure virtual]
```

Retrieves the y-component of the advance of the glyph. The advance indicates the positioning of the next glyph in the sequence, relative to the origin of the current glyph. Base glyphs generally have a non-zero advance and combining glyphs have a zero advance.

Returns

double The value of the y-component of the advance of the glyph, in ems.

getBounds()

```
virtual FRect IDOMGlyph::getBounds ( ) const  [pure virtual]
```

Retrieves the glyph bounding box.

The bounding box of a glyph is an imaginary box that encloses the glyph as tightly as possible. The bounding box is described as an FRect representing the origin of the bounding box relative to the glyph origin and the bounding box's width and height.

Note that it is possible for the values of the bounding box origin to be negative. For example, the origin of the glyph rests on the baseline for the font, so any glyph with a descender will have a negative y-component in the origin of its bounding box.

The bounding box values are not adjusted by any of the offsets that have been set.

Returns

FRect Bounding box value

getColored()

```
virtual bool IDOMGlyph::getColored ( ) const  [pure virtual]
```

Retrieves the flag which indicates if this glyph contains color information.

For most glyphs, the color is specified before the glyph proc is invoked and the current color is used in drawing the glyph. This type of glyph can be considered uncolored.

If some part of the glyph proc specifies its own colors then the glyph can be considered colored as that part of the glyph is always drawn in that color.

This is used in PostScript output to determine whether or not the glyph proc should use setcachedevice or setcharwidth in the PS proc.

Returns

bool Colored flag

getGlyphID()

```
virtual GlyphID IDOMGlyph::getGlyphID ( ) const  [pure virtual]
```

Retrieves the glyph id for this glyph.

Returns

GlyphID GlyphID value

getGlyphName() [1/2]

```
virtual IDOMGlyphNamePtr IDOMGlyph::getGlyphName ( ) const  [pure virtual]
```

Retrieves the glyph name for this glyph, if present, as a IDOMGlyphNamePtr. Returns NULL if the glyph has no name.

Returns

IDOMGlyphNamePtr The glyph name

getGlyphName() [2/2]

```
virtual bool IDOMGlyph::getGlyphName (
            EDLSysString & glyphName )  [pure virtual]
```

Retrieves the glyph name for this glyph, if present. An empty string is still a valid glyph name.

Parameters

glyphName	Reference to receive the glyph name

Returns

bool True on success, false if the call fails in any way.

getGlyphUnicode() [1/2]

```
virtual const EDLU32String & IDOMGlyph::getGlyphUnicode ( ) const  [pure virtual]
```

Retrieves the Unicode for the glyph, or an empty string.

Returns

EDLU32String The Unicode, or an empty string if no Unicode is present

getGlyphUnicode() [2/2]

```
virtual bool IDOMGlyph::getGlyphUnicode (
            uint32 & codePoint ) const  [pure virtual]
```

Retrieves the Unicode code point for the glyph, if the glyph has Unicode and the Unicode is a single code point.

Parameters

codePoint	A reference to receive the code point.

Returns

bool True on success.

getHasCustomAdvance()

```
virtual bool IDOMGlyph::getHasCustomAdvance ( ) const  [pure virtual]
```

Determines whether or not a custom value has been set for the glyph's advance.

The advance of a glyph is the distance from the glyph's origin to the origin of the next glyph along the baseline, which can be either vertical or horizontal.

A "custom" advance is one that has not been extracted from the font. A glyph has a custom advance if the advance information has been determined by the IndicesString or has been adjusted manually in some way.

Returns

bool True if the glyph is using a custom advance, and false otherwise.

getOriginalAdvanceX()

```
virtual double IDOMGlyph::getOriginalAdvanceX ( ) const  [pure virtual]
```

Retrieves the x-component of the original advance of the glyph.

The advance indicates the positioning of the next glyph in the sequence, relative to the origin of the current glyph. Base glyphs generally have a non-zero advance and combining glyphs have a zero advance. The original advance is the one specified in the font.

Advance width is measured in hundredths of the font em size.

Returns

double The value of the x-component of the original advance of the glyph

getUOffset()

```
virtual double IDOMGlyph::getUOffset ( ) const  [pure virtual]
```

Retrieves the UOffset value for the glyph.

Offsets are used to attach marks to base characters, and define how to move a glyph, relative to its origin, to place it correctly relative to the base glyph it is attached to. The offset value is added to the nominal glyph origin calculated using the advance to generate the actual origin for the glyph.

UOffset defines the x-component of the offset for a glyph; VOffset defines the y-component of the offset. UOffset and VOffset are real numbers, specified in ems.

By default, the offset values are (0.0, 0.0).

Base glyphs generally have a glyph offset of (0.0, 0.0). Combining glyphs generally have an offset that places them correctly on top of the nearest preceding base glyph. For left-to-right text, a positive UOffset value points to the right, for right-to-left text, a positive UOffset value points to the left.

Returns

double The value of UOffset.

getVOffset()

```
virtual double IDOMGlyph::getVOffset ( ) const  [pure virtual]
```

Retrieves the VOffset value for the glyph.

Offsets are used to attach marks to base characters, and define how to move a glyph, relative to its origin, to place it correctly relative to the base glyph it is attached to. The offset value is added to the nominal glyph origin calculated using the advance to generate the actual origin for the glyph.

UOffset defines the x-component of the offset for a glyph; VOffset defines the y-component of the offset. UOffset and VOffset are real numbers, specified in ems.

By default, the offset values are (0.0, 0.0).

Base glyphs generally have a glyph offset of (0.0, 0.0). Combining glyphs generally have an offset that places them correctly on top of the nearest preceding base glyph. For left-to-right text, a positive UOffset value points to the right, for right-to-left text, a positive UOffset value points to the left.

Returns

double Returns the value of VOffset.

setColored()

```
virtual void IDOMGlyph::setColored (
            bool bColored )  [pure virtual]
```

Sets a flag whereby the glyph specifies its own colors or not.

For most glyphs, the color is specified before the glyph proc is invoked and the current color is used in drawing the glyph. This type of glyph can be considered uncolored.

If some part of the glyph proc specifies its own colors then the glyph can be considered colored as that part of the glyph is always drawn in that color.

This is used in PostScript output to determine whether or not the glyph proc should use setcachedevice or setcharwidth in the PS proc

Parameters

bColored	Set to true if the glyph specifies its own colors

setHasCustomAdvance()

```
virtual void IDOMGlyph::setHasCustomAdvance (
            bool bHasCustomAdvance )  [pure virtual]
```

Informs whether a custom value has been set for the glyph's advance.

The advance of a glyph is the distance from the glyph's origin to the origin of the next glyph along the baseline, which can be either vertical or horizontal.

A "custom" advance is one that has not been extracted from the font. A glyph has a custom advance if the advance information has been determined by the IndicesString or has been adjusted manually in some way.

Parameters

bHasCustomAdvance	Set to true if the glyph has a custom advance

The documentation for this class was generated from the following file:

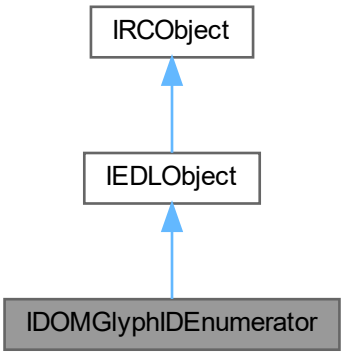
- idomglyph.h

7.199 IDOMGlyphIDEnumerator Class Reference

DOM GlyphID Enumerator.

`#include <idomglyph.h>`

Inheritance diagram for IDOMGlyphIDEnumerator:



Public Member Functions

- virtual void add (IDOMGlyph::GlyphID glyphID)=0

  Adds a glyph to the end of the list.
- virtual IDOMGlyph::GlyphID getGlyphID ()=0

  Returns the ID of the glyph at the current point in the list.
- virtual bool beginEnumeration ()=0

  Moves the current list point to the start of the list.
- virtual bool endEnumeration ()=0

  Indicates whether or not the end of the list has been reached.
- virtual bool nextEnumerationItem ()=0

  Moves the current list position on to the next item in the list, unless the end of the list has been reached.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves the class id of IDOMGlyphIDEnumerator.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.199.1 Detailed Description

DOM GlyphID Enumerator.

The IDOMGlyphIDEnumerator provides a generalized interface to a list of glyphs by ID. The list is designed to be read from beginning to end in sequence. Glyphs can only be added to the end of the list.

7.199.2 Member Function Documentation

add()

```
virtual void IDOMGlyphIDEnumerator::add (
            IDOMGlyph::GlyphID glyphID )  [pure virtual]
```

Adds a glyph to the end of the list.

Parameters

glyphID	The glyph ID to add to the end of the list.

beginEnumeration()

```
virtual bool IDOMGlyphIDEnumerator::beginEnumeration ( )  [pure virtual]
```

Moves the current list point to the start of the list.

Returns

bool True on success, false otherwise.

classID()

```
static const CClassID & IDOMGlyphIDEnumerator::classID ( )  [inline], [static]
```

Retrieves the class id of IDOMGlyphIDEnumerator.

Returns

CClassID Class id of IDOMGlyphIDEnumerator.

endEnumeration()

```
virtual bool IDOMGlyphIDEnumerator::endEnumeration ( )  [pure virtual]
```

Indicates whether or not the end of the list has been reached.

Returns

bool True if the end of the list has been reached, false otherwise.

getGlyphID()

```
virtual IDOMGlyph::GlyphID IDOMGlyphIDEnumerator::getGlyphID ( )  [pure virtual]
```

Returns the ID of the glyph at the current point in the list.

Returns

IDOMGlyph::GlyphID The ID of the glyph at the current position in the list.

nextEnumerationItem()

```
virtual bool IDOMGlyphIDEnumerator::nextEnumerationItem ( )  [pure virtual]
```

Moves the current list position on to the next item in the list, unless the end of the list has been reached.

Returns

bool True if the current list position was successfully advanced. In the case where the end of the list had already been reached prior to the call, the current list position remains on the last item and the function returns false.

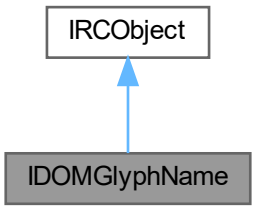
The documentation for this class was generated from the following file:

- idomglyph.h

7.200 IDOMGlyphName Class Reference

```
#include <idomglyph.h>
```

Inheritance diagram for IDOMGlyphName:



Additional Inherited Members

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.200.1 Detailed Description

Reference-counted name for a string. Used to reduce duplication of glyph names.

The documentation for this class was generated from the following file:

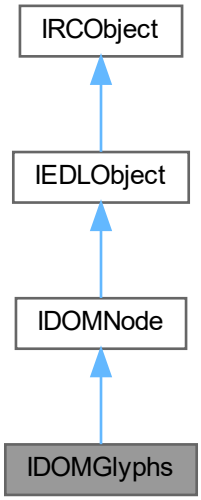
- idomglyph.h

7.201 IDOMGlyphs Class Reference

An abstract class providing an interface to a "Glyphs" node. Glyphs nodes are used to represent a run of uniformly formatted text from a single font. Text runs are broken by line advances and formatting changes. When a text run is broken, a new Glyphs node will be created to describe the text from the change point onwards.

`#include <idomglyphs.h>`

Inheritance diagram for IDOMGlyphs:



Classes

- class Data

  Initialization data.

Public Types

- enum eStyleSimulations

  Specifies a style simulation for a glyphs node.

Public Member Functions

- virtual uint8 getBidiLevel () const =0

  Retrieves the bidiLevel of the text run.
- virtual void setBidiLevel (uint8 bidiLevel)=0

  Sets the bidiLevel for this run of text.
- virtual const EDLSysString & getCaretStops () const =0

  Retrieves the XPS-specific caret stop specification for a run of glyphs.
- virtual void setCaretStops (const EDLSysString &caretStops)=0

  Sets the XPS-specific caret stop specification for a run of glyphs.
- virtual const EDLString & getDeviceFontName () const =0

  Retrieves the XPS-specific device font name. The device font name, which uniquely identifies a specific device font. The identifier is typically defined by a hardware vendor or font vendor.
- virtual void setDeviceFontName (const EDLString &name)=0

  Sets the the XPS-specific device font name The device font name, which uniquely identifies a specific device font. The identifier is typically defined by a hardware vendor or font vendor.
- virtual double getFontRenderingEmSize () const =0

  Retrieves the font rendering em size in drawing surface units. The font rendering em size is expressed as a floating point value in units of the effective coordinate space. A value of 0 results in no visible text.
- virtual void setFontRenderingEmSize (double emSize)=0

  Sets the font rendering em size in drawing surface units. The font rendering em size is expressed as a floating point value in units of the effective coordinate space. A value of 0 results in no visible text.
- virtual IDOMFontPtr getFont () const =0

  Retrieves a pointer to the font object describing the font in which the text is displayed.
- virtual void setFont (const IDOMFontPtr &font)=0

  Sets the font object to be used to display the text.
- virtual uint32 getFontIndex () const =0

  Retrieves the font index.
- virtual void setFontIndex (uint32 index)=0

  Sets the font index.
- virtual double getOriginX () const =0

  Retrieves the x-coordinate of the first glyph in the run, in units of the effective coordinate space. The first glyph is placed so that the leading edge of its advance vector and its baseline intersect with the point defined by the originX and originY coordinates.
- virtual void setOriginX (double originX)=0

  Sets the x-coordinate of the first glyph in the run, in units of the effective coordinate space. The first glyph is placed so that the leading edge of its advance vector and its baseline intersect with the point defined by the originX and originY coordinates.
- virtual double getOriginY () const =0

  Retrieves the y-coordinate of the first glyph in the run, in units of the effective coordinate space. The first glyph is placed so that the leading edge of its advance vector and its baseline intersect with the point defined by the originX and originY coordinates.
- virtual void setOriginY (double originY)=0

  Sets the y-coordinate of the first glyph in the run, in units of the effective coordinate space. The first glyph is placed so that the leading edge of its advance vector and its baseline intersect with the point defined by the originX and originY coordinates.
- virtual bool getIsSideways () const =0

  Indicates that a glyph is turned on its side, with the origin being defined as the top center of the unturned glyph.

- virtual void setIsSideways (bool isSideways)=0

  Sets the IsSideways value. See getIsSideways for a description of the effects of IsSideways.
- virtual const EDLSysString & getIndices () const =0

  Retrieves a Unicode string that represents the text of the run of glyphs.
- virtual void setIndices (const EDLSysString &indices)=0

  Sets the glyphs indices.
- virtual const EDLString & getUnicodeString () const =0

  Gets the Unicode string that represents the text of the run of glyphs.
- virtual void setUnicodeString (const EDLString &unicodeString)=0

  Sets the Unicode string that represents the text of the run of glyphs.
- virtual IGlyphsClustersPtr getClusters () const =0

  Unpack the UnicodeString and Indices strings into a set of clusters representing the smallest units of information in the text. This form combines the unicode and indices information such that they are more easily edited together. The returned IGlyphsCluster instance can be edited freely; a new instance is returned on every call. However, for the changes to have effect, setClusters() must be called to repopulate the UnicodeString and Indices.
- virtual void setClusters (const IGlyphsClustersPtr &clusters)=0

  Populate the glyphs node UnicodeString and Indices from the given set of glyphs clusters.
- virtual eStyleSimulations getStyleSimulations () const =0

  Retrieves the style simulation value.
- virtual void setStyleSimulations (eStyleSimulations styleSimulations)=0

  Retrieves the style simulation value.
- virtual const FMatrix & getRenderTransform () const =0

  Retrieves the render transform matrix of the Glyphs node and its children. The render transform establishes a new coordinate frame for the child and descendants of the Glyphs node. The render transform also affects clip and opacity masks, fill, x-origin, y-origin, the actual shape of the individual glyphs and the advance widths. The render transform also affects the font size and values specified by the indices setting.
- virtual void setRenderTransform (const FMatrix &matrix)=0

  Sets the render transform matrix of the Glyphs node and its children. The render transform establishes a new coordinate frame for the child and descendants of the Glyphs node. The render transform also affects clip and opacity masks, fill, x-origin, y-origin, the actual shape of the individual glyphs and the advance widths. The render transform also affects the font size and values specified by the indices setting.
- virtual float getOpacity () const =0

  Retrieves the opacity value of the Glyphs node. The opacity value defines the uniform transparency of the canvas. The opacity value is a number between 0 (fully transparent) and 1 (fully opaque). Values outside this range are invalid. The default value is 1.0.
- virtual void setOpacity (float opc)=0

  Sets the opacity value of the Glyphs node. The opacity value defines the uniform transparency of the canvas. The opacity value is a number between 0 (fully transparent) and 1 (fully opaque). Values outside this range are invalid. The default value is 1.0. Will return 0.0 if the node is intended to be inivisible (see #setIsInivisible()).
- virtual eBlendMode getBlendMode () const =0

  Gets the blend mode to be used for rendering this node.
- virtual void setBlendMode (eBlendMode blendMode)=0

  Sets the blend mode to be used for rendering this node. Note: modes other than Normal are not directly representable in XPS.
- virtual IDOMTargetPtr getNavigateLink () const =0

  Retrieves the target of a hyperlink.
- virtual void setNavigateLink (const IDOMTargetPtr &target)=0

  Sets the target of a hyperlink.
- virtual const EDLString & getLanguage () const =0

  Retrieves the default language of the Glyphs element and any of its children.
- virtual void setLanguage (const EDLString &lang)=0

  Sets the default language of the Glyphs element and any of its children.
- virtual IDOMBrushPtr getOpacityMask () const =0

Retrieves the opacity mask for the node. The opacity mask specifies a mask of alpha values that is applied to the glyphs in the same fashion as the opacity setting, but allowing different alpha values for different areas of the canvas.

- virtual void setOpacityMask (const IDOMBrushPtr &ptrOpacityMask)=0

  Sets the opacity mask for the node. The opacity mask specifies a mask of alpha values that is applied to the glyphs in the same fashion as the opacity setting, but allowing different alpha values for different areas of the canvas.
- virtual IDOMPathGeometryPtr getClip () const =0

  Retrieves the clip for the Glyphs node. The clip limits the rendered region of the glyph.
- virtual void setClip (const IDOMPathGeometryPtr &ptrClip)=0

  Sets the clip for the Glyphs node. The clip limits the rendered region of the glyph.
- virtual IDOMBrushPtr getFill () const =0

  Retrieves the fill brush used to fill the shape of the rendered glyphs.
- virtual void setFill (const IDOMBrushPtr &ptrFill)=0

  Sets the fill brush used to fill the shape of the rendered glyphs.
- virtual IDOMGlyph::CDOMGlyphDataVect getFlattenedGlyphInfo () const =0

  Retrieves the flattened glyph information, indicating every glyph that results from painting this glyphs nodes. The results are provided as a simple vector of IDOMGlyph::Data objects, laid out according to the glyphs node with offsets, glyph positioning, advance, bounds, unicode (if present) and glyph ID information computed.
- virtual bool getIsCharPath () const =0

  Retrieves a Boolean value which indicates if this Glyphs node is intended to draw a path.
- virtual void setIsCharPath (bool isCharPath)=0

  Sets the flag to indicate if this glyph is intended to draw a path.
- virtual bool getIsInvisible () const =0

  Retrieves a Boolean value which indicates if this Glyphs node is intended to be invisible.
- virtual void setIsInvisible (bool isInvisible)=0

  Sets the flag to indicate if this glyph is intended to be invisible.
- virtual IDOMPathNodePtr getEquivalentPath () const =0

  Get a graphically equivalent path node for the glyphs node, with all glyphs converted to path geometry. Not possible for glyphs nodes using Type 3 fonts, and will result in an exception.
- virtual IDOMNodePtr split () const =0

  Split the glyphs node into a series of glyph nodes each consisting of the smallest possible unit.
- virtual IDOMNodePtr hardSplit () const =0

  Split the glyphs node into a series of glyph nodes each consisting of single glyphs.
- virtual bool validateInstructions () const =0

  Validate glyph instructions.

Public Member Functions inherited from IDOMNode

- virtual ~**IDOMNode** ()

  virtual destructor
- virtual DOMid getDOMid () const =0

  Retrieves the node ID.
- virtual void setDOMid (DOMid id)=0

  Sets the node ID.
- virtual eDOMNodeType getNodeType () const =0

  Retrieves the DOM node type.
- virtual bool getProperty (const EDLSysString &propertyName, PValue &propertyValue) const =0

  Retrieves the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.
- virtual void setProperty (const EDLSysString &propertyName, const PValue &propertyValue)=0

Sets the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.

- virtual void removeProperty (const EDLSysString &propertyName)=0

  Removes property.
- virtual IEDLSysStringCollectionEnumPtr getPropertyCollectionEnum ()=0

  Retrieves a navigable list of the property names stored on this node.
- virtual bool hasChildNodes () const =0

  Function that indicates whether this node is a parent to other nodes.
- virtual IDOMNodePtr getParentNode () const =0

  Gets the parent node of this node.
- virtual IDOMNodePtr getFirstChild () const =0

  Gets the first child node of this node.
- virtual IDOMNodePtr getLastChild () const =0

  Gets the last child node of this node.
- virtual IDOMNodePtr getNextChild (const IDOMNodePtr &child) const =0

  Gets the child node which follows the node passed in.
- virtual IDOMNodePtr getPreviousChild (const IDOMNodePtr &child) const =0

  Gets the child node which precedes the node passed in.
- virtual IDOMNodePtr getPreviousSibling () const =0

  Retrieves the node's previous sibling node.
- virtual IDOMNodePtr getNextSibling () const =0

  Retrieves node's next sibling node.
- virtual void appendChild (const IDOMNodePtr &child)=0

  Appends a node to the end of the node's child list.
- virtual void insertChild (const IDOMNodePtr &ptrPreviousSibling, const IDOMNodePtr &child, bool bCheck↩ Complete=true)=0

  Insert a child node after ptrPreviousSibling.
- virtual IDOMNodePtr extractChild (const IDOMNodePtr &child)=0

  Extracts (that is, finds and removes) a child node from the node children. After extraction the child node is no longer a part of the DOM. If no node is specified, the first available node will be extracted from the node's children.
- virtual void replaceChild (const IDOMNodePtr &oldChild, const IDOMNodePtr &newChild)=0

  Replaces the child node with another.
- virtual bool isComplete () const =0

  Signals the completeness of the node.
  A complete node is one that has no more children to be added to it.
- virtual void **setComplete** ()=0

  Sets the node's completeness status to "true".
- virtual IDOMNodeFlags * getFlags ()=0

  Retrieves the node's flags property.
- virtual void setParentNode (const IDOMNodePtr &ptrParent)=0

  Sets the parent node.
- virtual void setPreviousSibling (const IDOMNodePtr &ptrPreviousSibling)=0

  Sets the previous sibling node.
- virtual void setNextSibling (const IDOMNodePtr &ptrNextSibling)=0

  Sets the next sibling node.
- virtual bool isAncestor (const IDOMNodePtr &ptrCandidate)=0

  Function tests whether a candidate node is a descendant of the node.
- virtual FRect getBounds (bool applyTransform=true, bool applyClip=true)

  Find the conservative bounding box of the marking content of the node.
- virtual bool copyNodeData (IDOMNode *pSourceNode)=0

  Copy the properties collection, the flags and the DOM ID from the given source node to this one.

- virtual IDOMNodePtr cloneNode (IEDLClassFactory *pFactory) const =0

  Simpified node cloning. An exception of type IEDLError will be thrown on failure.
- virtual IDOMNodePtr cloneTree (IEDLClassFactory *pFactory) const =0

  Clone the tree of nodes beginning at this node. An exception of type IEDLError will be thrown on failure.
- virtual void cloneTreeAndAppend (IEDLClassFactory *pFactory, const IDOMNodePtr &dest) const =0

  Clone the tree of nodes beginning at this node, and append the result to the destination tree.
- virtual void **completeTree** ()=0

  Mark the entire tree from this point as complete. You should not ordinarily need to call this function.
- virtual void **removeCompleteFlagFromTree** ()=0

  Mark the entire tree from this point as complete.
- virtual void findChildrenOfType (eDOMNodeType type, CDOMNodeVect &nodes, bool searchForms=false)=0

  Find all children of this node with the given type, appending to the given vector. Does not descend into brushes.
- virtual void walkTree (WalkTreeFunc func, void *priv, bool descendIntoBrushes=false, bool descendInto↩ Forms=false)=0

  Walk through the DOM calling a given function on each node. The function is allowed to:
- virtual void notifyOnDestruct (NodeDeleteFunc func, void *priv)=0

  Register interest in being told when this node is about to be destroyed.
- virtual void unregisterNotify (NodeDeleteFunc func, void *priv)=0

  Unregister interest in being told when this node is about to be destroyed.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static EDL_API IDOMGlyphsPtr create (IEDLClassFactory *pFactory, const EDLString &unicodeString, double size, const FPoint &position, const IDOMBrushPtr &brush, const IDOMFontPtr &font, uint32 fontIndex=0, eStyleSimulations styleSimulations=eSSNone, const FMatrix renderTransform=FMatrix(), const IDOMPath↩ GeometryPtr &clip=IDOMPathGeometryPtr(), float opacity=1.0f)

  Simplified creator for a glyphs object Throws an IEDLError on failure.
- static const CClassID & classID ()

  Retrieves class id of IDOM.

Static Public Member Functions inherited from IDOMNode

- static EDL_API FMatrix effectiveTransformationOfNode (const IDOMNodePtr &node)

  Attempt to find the effective transformation matrix external to the specified node relative to either a containing page or ultimate parent.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.201.1 Detailed Description

An abstract class providing an interface to a "Glyphs" node. Glyphs nodes are used to represent a run of uniformly formatted text from a single font. Text runs are broken by line advances and formatting changes. When a text run is broken, a new Glyphs node will be created to describe the text from the change point onwards.

A Glyphs node may describe an extended run of text or a single character, depending on the document formatting. The set of properties of a Glyphs node allows for a complete description of the glyph characteristics, such as the fill and opacity of the text, as well as clipping information.

A Glyphs node also allows the specification of a Unicode string and supports bidirectional and vertical text.

Some properties of the Glyphs node are composable, meaning that the markings rendered to the page are determined by a combination of the property and all the like-named properties of the Glyphs node's parent.

Errors are via exceptions of type IEDLError.

7.201.2 Member Function Documentation

classID()

```
static const CClassID & IDOMGlyphs::classID ( )  [inline], [static]
```

Retrieves class id of IDOM.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMGlyphsPtr IDOMGlyphs::create (
            IEDLClassFactory * pFactory,
            const EDLString & unicodeString,
            double size,
            const FPoint & position,
            const IDOMBrushPtr & brush,
            const IDOMFontPtr & font,
            uint32 fontIndex = 0,
            eStyleSimulations styleSimulations = eSSNone,
            const FMatrix renderTransform = FMatrix(),
            const IDOMPathGeometryPtr & clip = IDOMPathGeometryPtr(),
            float opacity = 1.0f )  [static]
```

Simplified creator for a glyphs object Throws an IEDLError on failure.

Parameters

pFactory	The factory to use.
position	The desired position for the glyphs.
unicodeString	The text to be used.
size	The font size to use, in ems.
brush	The fill brush to use.
font	The font to use.
fontIndex	The index of the font to use, if the font is within a truetype collection.
styleSimulations	The desired style simulations effect.
renderTransform	The render transform to use.
clip	The geometry to use for clipping. NULL if no clip.
opacity	The opacity to use.

Returns

IDOMGlyphsPtr The new glyphs object.

getBidiLevel()

```
virtual uint8 IDOMGlyphs::getBidiLevel ( ) const  [pure virtual]
```

Retrieves the bidiLevel of the text run.

The bidi (bi-directional) level specifies the Unicode bi-directional nesting level. A value of 0 implies left-to-right text, while a value of 1 implies right-to-left text. Right-to-left text places the run origin at the right side of the first glyph, with positive advance widths representing advances to the left, so that subsequent glyphs are placed to the left of the previous glyph.

Returns

uint8 The bidiLevel value.

getBlendMode()

```
virtual eBlendMode IDOMGlyphs::getBlendMode ( ) const  [pure virtual]
```

Gets the blend mode to be used for rendering this node.

Returns

eBlendMode The current blend mode.

getCaretStops()

```
virtual const EDLSysString & IDOMGlyphs::getCaretStops ( ) const  [pure virtual]
```

Retrieves the XPS-specific caret stop specification for a run of glyphs.

Caret stops identify the positions within the sequence of Unicode characters at which a text-selection tool may place a text-editing caret. Potential caret-stop positions are identified by their indices into the UTF-16 code units represented by the unicodeString attribute value. When this value is missing, the test in the UnicodeString attribute must be interpreted as having a caret stop between every Unicode UTF-16 code unit and at the beginning and end of the text.

Because it is legal to place the caret before any of the UTF-16 code units in the Unicode string if the caretStops attribute is omitted, omitting the caretStops attribute is equivalent to specifying a string that has all the bits set to 1.

If there are insufficient flags in the caretStops string to correspond to all the UTF-16 code units in the Unicode string, all remaining UTF-16 code units in the Unicode string MUST be considered valid caret stops.

Returns

EDLSysString The caret stops specification string, or an empty string if not present. See setCaretStops() for a description of the format of the specification string.

getClip()

```
virtual IDOMPathGeometryPtr IDOMGlyphs::getClip ( ) const  [pure virtual]
```

Retrieves the clip for the Glyphs node. The clip limits the rendered region of the glyph.

Returns

IDOMPathGeometryPtr The clip, or NULL if there is no clip set.

getClusters()

```
virtual IGlyphsClustersPtr IDOMGlyphs::getClusters ( ) const  [pure virtual]
```

Unpack the UnicodeString and Indices strings into a set of clusters representing the smallest units of information in the text. This form combines the unicode and indices information such that they are more easily edited together. The returned IGlyphsCluster instance can be edited freely; a new instance is returned on every call. However, for the changes to have effect, setClusters() must be called to repopulate the UnicodeString and Indices.

Returns

IGlyphsClustersPtr The glyphs clusters.

getDeviceFontName()

```
virtual const EDLString & IDOMGlyphs::getDeviceFontName ( ) const  [pure virtual]
```

Retrieves the XPS-specific device font name. The device font name, which uniquely identifies a specific device font. The identifier is typically defined by a hardware vendor or font vendor.

Returns

EDLString The device font name, or an empty string if there is no device font name.

getEquivalentPath()

```
virtual IDOMPathNodePtr IDOMGlyphs::getEquivalentPath ( ) const  [pure virtual]
```

Get a graphically equivalent path node for the glyphs node, with all glyphs converted to path geometry. Not possible for glyphs nodes using Type 3 fonts, and will result in an exception.

Returns

IDOMPathNodePtr The equivalent path.

getFill()

```
virtual IDOMBrushPtr IDOMGlyphs::getFill ( ) const  [pure virtual]
```

Retrieves the fill brush used to fill the shape of the rendered glyphs.

Returns

IDOMBrushPtr The brush, or NULL if there is no fill brush set.

getFlattenedGlyphInfo()

```
virtual IDOMGlyph::CDOMGlyphDataVect IDOMGlyphs::getFlattenedGlyphInfo ( ) const  [pure virtual]
```

Retrieves the flattened glyph information, indicating every glyph that results from painting this glyphs nodes. The results are provided as a simple vector of IDOMGlyph::Data objects, laid out according to the glyphs node with offsets, glyph positioning, advance, bounds, unicode (if present) and glyph ID information computed.

Returns

IDOMGlyph::CDOMGlyphDataVect The flattened glyph information.

getFont()

```
virtual IDOMFontPtr IDOMGlyphs::getFont ( ) const  [pure virtual]
```

Retrieves a pointer to the font object describing the font in which the text is displayed.

Returns

IDOMFontPtr A smart pointer to the text's font object. Will not be NULL.

getFontIndex()

```
virtual uint32 IDOMGlyphs::getFontIndex ( ) const  [pure virtual]
```

Retrieves the font index.

Returns

uint32 The font index.

getFontRenderingEmSize()

```
virtual double IDOMGlyphs::getFontRenderingEmSize ( ) const  [pure virtual]
```

Retrieves the font rendering em size in drawing surface units. The font rendering em size is expressed as a floating point value in units of the effective coordinate space. A value of 0 results in no visible text.

Returns

double The font rendering em size.

getIndices()

```
virtual const EDLSysString & IDOMGlyphs::getIndices ( ) const  [pure virtual]
```

Retrieves a Unicode string that represents the text of the run of glyphs.

Returns

EDLSysString The indices, or an empty string if the indices string is not present or empty.

getIsCharPath()

```
virtual bool IDOMGlyphs::getIsCharPath ( ) const  [pure virtual]
```

Retrieves a Boolean value which indicates if this Glyphs node is intended to draw a path.

Returns

bool True if this node is a charpath , false otherwise.

getIsInvisible()

```
virtual bool IDOMGlyphs::getIsInvisible ( ) const  [pure virtual]
```

Retrieves a Boolean value which indicates if this Glyphs node is intended to be invisible.

Returns

bool True if this node is intended to be invisible, false otherwise.

getIsSideways()

```
virtual bool IDOMGlyphs::getIsSideways ( ) const  [pure virtual]
```

Indicates that a glyph is turned on its side, with the origin being defined as the top center of the unturned glyph.

Glyphs for text in vertical writing are normally represented by rotating the co-ordinate system and using the isSideways attribute. Glyph runs with isSideways set to true will be rotated 90 degrees counter-clockwise and placed so that the sideways baseline origin is coincident with the normal origin of the character, as modified by the offset vector in the indices attribute. The advance setting places the nominal origin of the next character a distance along the direction of the progression of the run. The direction of the advance setting is unaffected by isSideways, however the method by which the size of the advance vector is chosen is different.

For example, to represent a run of characters top to bottom of a page, a render transform can be used to rotate the coordinate system 90 degrees clockwise. originX and originY can be used to specify a position at the top of the column of text. Text from a vertical writing system can then be written using a glyphs run with isSideways set to true. The individual glyphs appear in the normal orientation because the rotation is effected by the isSideways attribute undoes the effect of the render transform.

Returns

bool The IsSideways value

getLanguage()

```
virtual const EDLString & IDOMGlyphs::getLanguage ( ) const  [pure virtual]
```

Retrieves the default language of the Glyphs element and any of its children.

```
The default language is specified according to RFC 3066. English is defined as en_GB and American
English as en_US. There is no default setting. If the language is not known it is
set to und (undetermined). For further information see
http://www.w3.org/International/articles/language-tags/
```

Returns

EDLString The language.

getNavigateLink()

```
virtual IDOMTargetPtr IDOMGlyphs::getNavigateLink ( ) const  [pure virtual]
```

Retrieves the target of a hyperlink.

Returns

IDOMTargetPtr the target, or NULL if there is no target.

getOpacity()

```
virtual float IDOMGlyphs::getOpacity ( ) const  [pure virtual]
```

Retrieves the opacity value of the Glyphs node. The opacity value defines the uniform transparency of the canvas. The opacity value is a number between 0 (fully transparent) and 1 (fully opaque). Values outside this range are invalid. The default value is 1.0.

Returns

float Returns the opacity value.

getOpacityMask()

```
virtual IDOMBrushPtr IDOMGlyphs::getOpacityMask ( ) const  [pure virtual]
```

Retrieves the opacity mask for the node. The opacity mask specifies a mask of alpha values that is applied to the glyphs in the same fashion as the opacity setting, but allowing different alpha values for different areas of the canvas.

Returns

IDOMBrushPtr The opacity mask, or NULL if there is no opacity mask set.

getOriginX()

```
virtual double IDOMGlyphs::getOriginX ( ) const  [pure virtual]
```

Retrieves the x-coordinate of the first glyph in the run, in units of the effective coordinate space. The first glyph is placed so that the leading edge of its advance vector and its baseline intersect with the point defined by the originX and originY coordinates.

Returns

double The first glyph's x-coordinate.

getOriginY()

```
virtual double IDOMGlyphs::getOriginY ( ) const  [pure virtual]
```

Retrieves the y-coordinate of the first glyph in the run, in units of the effective coordinate space. The first glyph is placed so that the leading edge of its advance vector and its baseline intersect with the point defined by the originX and originY coordinates.

Returns

double The first glyph's y-coordinate.

getRenderTransform()

```
virtual const FMatrix & IDOMGlyphs::getRenderTransform ( ) const  [pure virtual]
```

Retrieves the render transform matrix of the Glyphs node and its children. The render transform establishes a new coordinate frame for the child and descendants of the Glyphs node. The render transform also affects clip and opacity masks, fill, x-origin, y-origin, the actual shape of the individual glyphs and the advance widths. The render transform also affects the font size and values specified by the indices setting.

Returns

FMatrix The transformation matrix.

getStyleSimulations()

```
virtual eStyleSimulations IDOMGlyphs::getStyleSimulations ( ) const  [pure virtual]
```

Retrieves the style simulation value.

Synthetic style simulations can be applied to the shape of the glyphs by using the Style Simulations function. Style simulations can be applied to the designed style of a font. The default value for Style Simulations is eSSNone, that is, the shapes of the glyphs are not modified. When eSSBold is set, the glyphs are geometrically widened by 1% of the em size, so that the centres of strokes remain at the same position. This leaves the baseline origin unmodified. The bounding box grows 1% all around for a total of 2% horizontal and 2% vertical. As a result the character height, and the advance width of each glyph are increased by 2% of the em size.

When eSSItalic is set, italicization is applied to glyphs with an IsSideways value of false. Italicization is applied by skewing the top edge of the bounding box by 20 degrees to the right, relative to the baseline of the character. Glyphs with an IsSideways value of true are italicized by skewing the right edge of the bounding box by 20 degrees down, relative to the baseline origin of the glyph. The character height and advance width are not modified.

It is possible to apply both bold and italic simulations to the glyphs by setting eSSBoldItalic.

Returns

eStyleSimulations The current style simulation setting.

getUnicodeString()

```
virtual const EDLString & IDOMGlyphs::getUnicodeString ( ) const  [pure virtual]
```

Gets the Unicode string that represents the text of the run of glyphs.

Returns

EDLString The unicode string.

hardSplit()

```
virtual IDOMNodePtr IDOMGlyphs::hardSplit ( ) const  [pure virtual]
```

Split the glyphs node into a series of glyph nodes each consisting of single glyphs.

As per split above, however here we allow unicode information to be lost in order to ensure that each glyphs node in the split result consists of a single glyph only.

The behaviour for hardSplit() only differs from split() if there is a case where one or more unicode characters is mapped to multiple glyphs in a single cluster. In this case, the cluster will be broken apart, and the unicode is lost.

Useful where it is imperative that each glyph node consists of a single glyph.

Returns

IDOMNodePtr The result of splitting the glyphs node.

setBidiLevel()

```
virtual void IDOMGlyphs::setBidiLevel (
            uint8 bidiLevel )  [pure virtual]
```

Sets the bidiLevel for this run of text.

The bidi (bi-directional) level specifies the Unicode bi-directional nesting level. A value of 0 implies left-to-right text, while a value of 1 implies right-to-left text. Right-to-left text places the run origin at the right side of the first glyph, with positive advance widths representing advances to the left, so that subsequent glyphs are placed to the left of the previous glyph.

Parameters

bidiLevel	The new bidiLevel value.

setBlendMode()

```
virtual void IDOMGlyphs::setBlendMode (
            eBlendMode blendMode )  [pure virtual]
```

Sets the blend mode to be used for rendering this node. Note: modes other than Normal are not directly representable in XPS.

Parameters

blendMode	The new blend mode.

setCaretStops()

```
virtual void IDOMGlyphs::setCaretStops (
            const EDLSysString & caretStops )  [pure virtual]
```

Sets the XPS-specific caret stop specification for a run of glyphs.

Caret stops identify the positions within the sequence of Unicode characters at which a text-selection tool may place a text-editing caret. Potential caret-stop positions are identified by their indices into the UTF-16 code units represented by the unicodeString attribute value. When this value is missing, the test in the UnicodeString attribute must be interpreted as having a caret stop between every Unicode UTF-16 code unit and at the beginning and end of the text.

Because it is legal to place the caret before any of the UTF-16 code units in the Unicode string if the caretStops attribute is omitted, omitting the caretStops attribute is equivalent to specifying a string that has all the bits set to 1.

If there are insufficient flags in the caretStops string to correspond to all the UTF-16 code units in the Unicode string, all remaining UTF-16 code units in the Unicode string MUST be considered valid caret stops.

Parameters

caretStops	The caret stops specification expressed as a string.

The caretStops attribute contains an array of boolean bit-flags, which is represented as a string of hexadecimal characters. The caretStops attribute includes a final flag for placement of the caret following the final UTF-16 code unit in the Unicode string. Each hexadecimal character in the caretStops value represents the flags for four UTF-16 code units in the Unicode string, with the highest-order bit representing the first UTF-16 code unit. Any unused bits in the last UTF-16 code unit must be 0. For example if caretStops is set to "f7" (that is, 1111 0111), in reference to the run of glyphs, "Not here.", it means that it is valid to place the caret stop in any position in the run of text except for directly before the "h" of "here".

setClip()

```
virtual void IDOMGlyphs::setClip (
            const IDOMPathGeometryPtr & ptrClip )  [pure virtual]
```

Sets the clip for the Glyphs node. The clip limits the rendered region of the glyph.

Parameters

ptrClip	Smart pointer to the new clip information, or NULL to clear.

setClusters()

```
virtual void IDOMGlyphs::setClusters (
            const IGlyphsClustersPtr & clusters )  [pure virtual]
```

Populate the glyphs node UnicodeString and Indices from the given set of glyphs clusters.

Parameters

clusters	The clusters to use

setDeviceFontName()

```
virtual void IDOMGlyphs::setDeviceFontName (
            const EDLString & name )  [pure virtual]
```

Sets the the XPS-specific device font name The device font name, which uniquely identifies a specific device font. The identifier is typically defined by a hardware vendor or font vendor.

Parameters

name	The new device font name, or an empty string to clear.

setFill()

```
virtual void IDOMGlyphs::setFill (
            const IDOMBrushPtr & ptrFill )  [pure virtual]
```

Sets the fill brush used to fill the shape of the rendered glyphs.

Parameters

ptrFill	Smart pointer to the new fill brush.

setFont()

```
virtual void IDOMGlyphs::setFont (
            const IDOMFontPtr & font )  [pure virtual]
```

Sets the font object to be used to display the text.

Parameters

font	Smart pointer to the new font object to use. Must not be NULL.

setFontIndex()

```
virtual void IDOMGlyphs::setFontIndex (
            uint32 index )  [pure virtual]
```

Sets the font index.

Parameters

index	The new font index value.

setFontRenderingEmSize()

```
virtual void IDOMGlyphs::setFontRenderingEmSize (
            double emSize )  [pure virtual]
```

Sets the font rendering em size in drawing surface units. The font rendering em size is expressed as a floating point value in units of the effective coordinate space. A value of 0 results in no visible text.

Parameters

emSize	The new font rendering em size value.

setIndices()

```
virtual void IDOMGlyphs::setIndices (
            const EDLSysString & indices )  [pure virtual]
```

Sets the glyphs indices.

Parameters

indices	The string specifying the indices. Within the string, each glyph specification is separated by a semicolon.

setIsCharPath()

```
virtual void IDOMGlyphs::setIsCharPath (
            bool isCharPath )  [pure virtual]
```

Sets the flag to indicate if this glyph is intended to draw a path.

Parameters

isCharPath	True if this node is a charpath, false otherwise.

setIsInvisible()

```
virtual void IDOMGlyphs::setIsInvisible (
            bool isInvisible )  [pure virtual]
```

Sets the flag to indicate if this glyph is intended to be invisible.

Parameters

isInvisible	True if this node is to be considered invisible, false otherwise.

setIsSideways()

```
virtual void IDOMGlyphs::setIsSideways (
            bool isSideways )  [pure virtual]
```

Sets the IsSideways value. See getIsSideways for a description of the effects of IsSideways.

Parameters

isSideways	The new IsSideways value.

setLanguage()

```
virtual void IDOMGlyphs::setLanguage (
            const EDLString & lang )  [pure virtual]
```

Sets the default language of the Glyphs element and any of its children.

```
The default language is specified according to RFC 3066. English is defined as en_GB and American
English as en_US. There is no default setting. If the language is not known it is
set to und (undetermined). For further information see
http://www.w3.org/International/articles/language-tags/
@param    lang The new default language string.
```

setNavigateLink()

```
virtual void IDOMGlyphs::setNavigateLink (
            const IDOMTargetPtr & target )  [pure virtual]
```

Sets the target of a hyperlink.

Parameters

target	The target to associate with the node, or NULL to clear.

setOpacity()

```
virtual void IDOMGlyphs::setOpacity (
            float opc )  [pure virtual]
```

Sets the opacity value of the Glyphs node. The opacity value defines the uniform transparency of the canvas. The opacity value is a number between 0 (fully transparent) and 1 (fully opaque). Values outside this range are invalid. The default value is 1.0. Will return 0.0 if the node is intended to be inivisible (see #setIsInivisible()).

Parameters

opc	The new opacity value.

setOpacityMask()

```
virtual void IDOMGlyphs::setOpacityMask (
            const IDOMBrushPtr & ptrOpacityMask )  [pure virtual]
```

Sets the opacity mask for the node. The opacity mask specifies a mask of alpha values that is applied to the glyphs in the same fashion as the opacity setting, but allowing different alpha values for different areas of the canvas.

Parameters

ptrOpacityMask	Smart pointer to a brush representing the new opacity mask for the node, or null to clear.

setOriginX()

```
virtual void IDOMGlyphs::setOriginX (
            double originX )  [pure virtual]
```

Sets the x-coordinate of the first glyph in the run, in units of the effective coordinate space. The first glyph is placed so that the leading edge of its advance vector and its baseline intersect with the point defined by the originX and originY coordinates.

Parameters

originX	The new x-coordinate for the first glyph in the text run.

setOriginY()

```
virtual void IDOMGlyphs::setOriginY (
            double originY )  [pure virtual]
```

Sets the y-coordinate of the first glyph in the run, in units of the effective coordinate space. The first glyph is placed so that the leading edge of its advance vector and its baseline intersect with the point defined by the originX and originY coordinates.

Parameters

originY	The new y-coordinate for the first glyph in the text run.

setRenderTransform()

```
virtual void IDOMGlyphs::setRenderTransform (
            const FMatrix & matrix )  [pure virtual]
```

Sets the render transform matrix of the Glyphs node and its children. The render transform establishes a new coordinate frame for the child and descendants of the Glyphs node. The render transform also affects clip and opacity masks, fill, x-origin, y-origin, the actual shape of the individual glyphs and the advance widths. The render transform also affects the font size and values specified by the indices setting.

Parameters

matrix	the new render transform matrix.

setStyleSimulations()

```
virtual void IDOMGlyphs::setStyleSimulations (
            eStyleSimulations styleSimulations )  [pure virtual]
```

Retrieves the style simulation value.

Synthetic style simulations can be applied to the shape of the glyphs by using the Style Simulations function. Style simulations can be applied to the designed style of a font. The default value for Style Simulations is e↩ StyleSimulationsNone, that is, the shapes of the glyphs are not modified. When eSSBold is set, the glyphs are geometrically widened by 1% of the em size, so that the centres of strokes remain at the same position. This leaves the baseline origin unmodified. The bounding box grows 1% all around for a total of 2% horizontal and 2% vertical. As a result the character height, and the advance width of each glyph are increased by 2% of the em size.

When eSSItalic is set, italicization is applied to glyphs with an IsSideways value of false. Italicization is applied by skewing the top edge of the bounding box by 20 degrees to the right, relative to the baseline of the character. Glyphs with an IsSideways value of true are italicized by skewing the right edge of the bounding box by 20 degrees down, relative to the baseline origin of the glyph. The character height and advance width are not modified.

It is possible to apply both bold and italic simulations to the glyphs by setting eSSBoldItalic.

Parameters

styleSimulations	The new style simulation setting.

setUnicodeString()

```
virtual void IDOMGlyphs::setUnicodeString (
            const EDLString & unicodeString )  [pure virtual]
```

Sets the Unicode string that represents the text of the run of glyphs.

Parameters

unicodeString	The Unicode string to set.

split()

```
virtual IDOMNodePtr IDOMGlyphs::split ( ) const  [pure virtual]
```

Split the glyphs node into a series of glyph nodes each consisting of the smallest possible unit.

The resulting glyphs nodes may still consist of multiple unicode codepoints per node if the combinations of code points results in a single glyph (such as for ligatures). Similarly the resulting glyphs nodes may still consist of multiple individual glyphs with a single code point (such as for combining characters). Regardless, the resulting glyphs nodes are split to the smallest form that can be interpreted on their own.

If the glyphs node cannot be further split (for example if it is already in it's simplest form), then the returned result will be this glyphs node.

If the glyphs node can be split, the returned node will be an IDOMGroupNode or one of it's subclasses, with the split glyphs present as children of that group.

Depending on the brushes or transparency that may be used with the glyphs node, the group may be a simple IDOMGroup, an IDOMCanvas or alternatively an IDOMTransparency group. In any case, the simplest form that can be used will be used. The transform for the glyphs node will be moved to the group.

Returns

IDOMNodePtr The result of splitting the glyphs node.

validateInstructions()

```
virtual bool IDOMGlyphs::validateInstructions ( ) const  [pure virtual]
```

Validate glyph instructions.

Applies only to TrueType glyphs. Not possible for glyphs nodes using Type 3 fonts, and will result in an exception. This will attempt to detect some cases of broken instructions (hints) by rendering each glyph in the glyphs node and return a boolean value to indicate if any issues were found.

Note that while this function may be useful to detect general issues with fonts/glyph instructions it cannot detect issues seen only when rendering at a specific resolution and point size.

Returns

bool true on success, or false if invalid instructions were found.

The documentation for this class was generated from the following file:

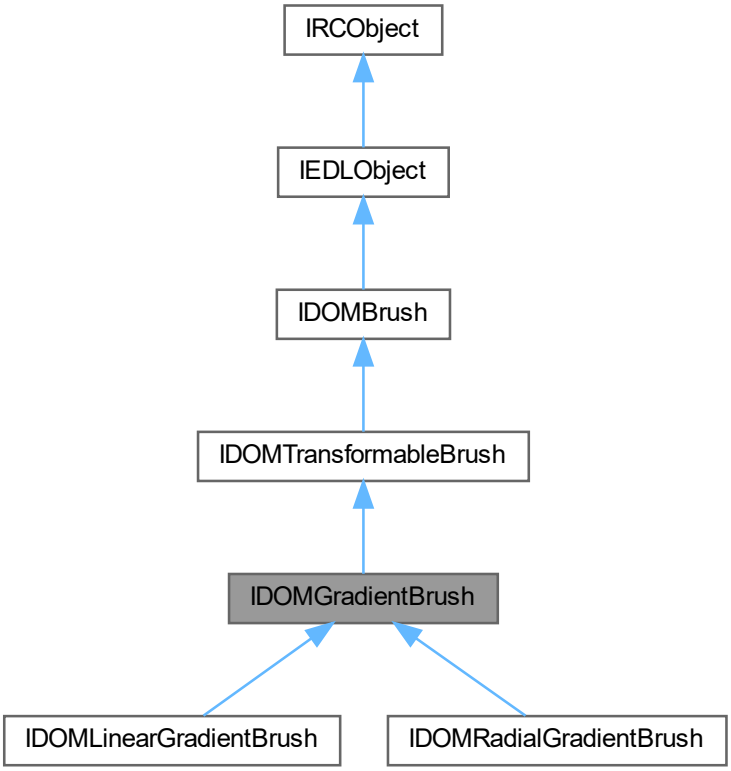
- idomglyphs.h

7.202 IDOMGradientBrush Class Reference

A common interface for both IDOMLinearGradient and IDOMRadialGradient. Provides straightforward access to common attributes.

`#include <idombrush.h>`

Inheritance diagram for IDOMGradientBrush:



Public Member Functions

- virtual eColorInterpolationMode getColorInterpolationMode () const =0

  Retrieves the color interpolation mode value of the radial gradient brush.
- virtual void setColorInterpolationMode (eColorInterpolationMode cim)=0

  Sets the color interpolation mode value of the radial gradient brush.
- virtual eSpreadMethod getSpreadMethod () const =0

  Retrieves the spread method value of the RadialGradientBrush element.
- virtual void setSpreadMethod (eSpreadMethod sm)=0

  Sets spread method value of the RadialGradientBrush element.
- virtual void setGradientStops (const CDOMGradientStopVect &stops)=0

  Set the vector of stops in this gradient. Must not be empty.
- virtual const CDOMGradientStopVect & getGradientStops () const =0

  Retrieves the vector of stops in this gradient. An exception will be thrown if the gradient has no stops.

- virtual void addGradientStop (const IDOMGradientStopPtr &ptrGradientStop)=0

  Append a gradient stop.
- virtual void normalizeStops (IEDLClassFactory *pFactory, const IDOMColorSpacePtr &ptrColorSpace=IDOMColorSpacePtr(), eRenderingIntent intent=eRelativeColorimetric, eBlackPointCompensation bpc=eBPCDefault)=0

  Normalize the gradient stops to a single color space, sort them and ensure there are stops at 0.0 and 1.0.

Public Member Functions inherited from IDOMTransformableBrush

- virtual const FMatrix & getRenderTransform () const =0

  Retrieves the render transform matrix.
- virtual void setRenderTransform (const FMatrix &matrix)=0

  Sets the render transform matrix.

Public Member Functions inherited from IDOMBrush

- virtual eBrushType getBrushType () const =0

  Retrieves the type of the brush.
- virtual float getOpacity () const =0

  Retrieves the opacity value of the brush element.
- virtual void setOpacity (float opc)=0

  Sets the opacity value of a brush element.
- virtual IDOMBrushPtr getAdjustedForUseInTransformedNode (IEDLClassFactory *pFactory, const FMatrix &nodeTransform)

  Get a version of this brush adjusted for use inside a node with the given transform.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Public Types inherited from IDOMBrush

- enum eBrushType {
  eSolidColor , eLinearGradient , eRadialGradient , eImage ,
  eMasked , eVisual , eSoftMask , eTilingPattern ,
  eType1ShadingPattern , eType2ShadingPattern , eType3ShadingPattern , eType4567ShadingPattern ,
  eNull }

  Brush type enumeration.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.202.1 Detailed Description

A common interface for both IDOMLinearGradient and IDOMRadialGradient. Provides straightforward access to common attributes.

7.202.2 Member Function Documentation

addGradientStop()

```
virtual void IDOMGradientBrush::addGradientStop (
            const IDOMGradientStopPtr & ptrGradientStop )  [pure virtual]
```

Append a gradient stop.

Parameters

ptrGradientStop	The smart pointer to the gradient stop interface

See also

IDOMGradientStop

getColorInterpolationMode()

```
virtual eColorInterpolationMode IDOMGradientBrush::getColorInterpolationMode ( ) const  [pure
virtual]
```

Retrieves the color interpolation mode value of the radial gradient brush.

This is the gamma function for color interpolation for sRGB colors. The gamma adjustment should not be applied to the alpha component, if specified.

See also

eColorInterpolationMode

Returns

eColorInterpolationMode The color interpolation mode.

getGradientStops()

```
virtual const CDOMGradientStopVect & IDOMGradientBrush::getGradientStops ( ) const  [pure
virtual]
```

Retrieves the vector of stops in this gradient. An exception will be thrown if the gradient has no stops.

See also

IDOMGradientStop

Returns

CDOMGradientStopVect The gradient stops

getSpreadMethod()

```
virtual eSpreadMethod IDOMGradientBrush::getSpreadMethod ( ) const  [pure virtual]
```

Retrieves the spread method value of the RadialGradientBrush element.

The spread method describes how the brush should fill the content area outside of the primary gradient area.

See also

eSpreadMethod

Returns

SpreadMethod The spread method value.

normalizeStops()

```
virtual void IDOMGradientBrush::normalizeStops (
            IEDLClassFactory * pFactory,
            const IDOMColorSpacePtr & ptrColorSpace = IDOMColorSpacePtr(),
            eRenderingIntent intent = eRelativeColorimetric,
            eBlackPointCompensation bpc = eBPCDefault )  [pure virtual]
```

Normalize the gradient stops to a single color space, sort them and ensure there are stops at 0.0 and 1.0.

Parameters

pFactory	The EDL class factory
ptrColorSpace	The desired colors pace. Pass NULL to use the color space of the first brush.
intent	The desired conversion intent if color conversion is required.
bpc	The desired black point compensation treatment if color conversion is required.

setColorInterpolationMode()

```
virtual void IDOMGradientBrush::setColorInterpolationMode (
            eColorInterpolationMode cim )  [pure virtual]
```

Sets the color interpolation mode value of the radial gradient brush.

This is the gamma function for color interpolation for sRGB colors. The gamma adjustment should not be applied to the alpha component, if specified.

Parameters

cim	The color interpolation mode value

See also

eColorInterpolationMode

setGradientStops()

```
virtual void IDOMGradientBrush::setGradientStops (
            const CDOMGradientStopVect & stops )  [pure virtual]
```

Set the vector of stops in this gradient. Must not be empty.

See also

IDOMGradientStop

Parameters

stops	The gradient stops.

setSpreadMethod()

```
virtual void IDOMGradientBrush::setSpreadMethod (
            eSpreadMethod sm )  [pure virtual]
```

Sets spread method value of the RadialGradientBrush element.

The spread method describes how the brush should fill the content area outside of the primary gradient area.

Parameters

sm	The new spread method value.

The documentation for this class was generated from the following file:

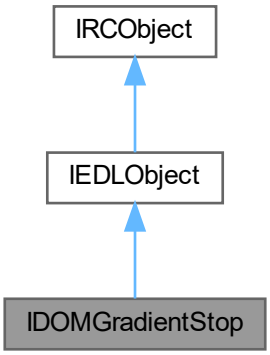
- idombrush.h

7.203 IDOMGradientStop Class Reference

IDOMGradientStop defines the ramp of colors to use on a gradient.

`#include <idombrush.h>`

Inheritance diagram for IDOMGradientStop:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual IDOMColorPtr getColor () const =0

  Retrieves the color value of the gradient stop. Never returns NULL.
- virtual void setColor (const IDOMColorPtr &color)=0

  Sets the color value of the gradient stop.
- virtual double getOffset () const =0

  Retrieves the offset value of the gradient stop.
- virtual void setOffset (double offset)=0

  Sets the offset value of the gradient stop.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static EDL_API IDOMGradientStopPtr create (IEDLClassFactory *pFactory, const IDOMColorPtr &color, double offset)

  Simplified gradient stop creation. Throws an IEDLError on failure.
- static const CClassID & classID ()

  Retrieves class id of IDOMGradientStop.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.203.1 Detailed Description

IDOMGradientStop defines the ramp of colors to use on a gradient.

A gradient is a smooth transition from one color to the next. The color range for a gradient can be composed of two or more colors. Each color used in a gradient requires at least one gradient stop. A gradient stop has two attributes: color and offset. The offset defines a line along which all points have the same color value; it determines the distance from the base line (for linear gradients) or the radial distance from the starting point of the gradient (for radial gradients), relative to the total distance the gradient covers. The color value determines the color of all points on the line specified by the offset. Between two adjacent gradient stops, the color shades smoothly from the color specified at the first offset to that specified at the second.

7.203.2 Member Function Documentation

classID()

```
static const CClassID & IDOMGradientStop::classID ( )  [inline], [static]
```

Retrieves class id of IDOMGradientStop.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMGradientStopPtr IDOMGradientStop::create (
            IEDLClassFactory * pFactory,
            const IDOMColorPtr & color,
            double offset )  [static]
```

Simplified gradient stop creation. Throws an IEDLError on failure.

Parameters

pFactory	The EDL Class factory to use.
color	The color to use.
offset	The stop offset

Returns

IDOMGradientStopPtr The new gradient stop

getColor()

```
virtual IDOMColorPtr IDOMGradientStop::getColor ( ) const  [pure virtual]
```

Retrieves the color value of the gradient stop. Never returns NULL.

Returns

IDOMGradientStop the color of the stop

getOffset()

```
virtual double IDOMGradientStop::getOffset ( ) const  [pure virtual]
```

Retrieves the offset value of the gradient stop.

Returns

double Returns the offset value.

setColor()

```
virtual void IDOMGradientStop::setColor (
            const IDOMColorPtr & color )  [pure virtual]
```

Sets the color value of the gradient stop.

Parameters

color	The new color value.

setOffset()

```
virtual void IDOMGradientStop::setOffset (
            double offset )  [pure virtual]
```

Sets the offset value of the gradient stop.

Parameters

offset	The new offset value.

The documentation for this class was generated from the following file:

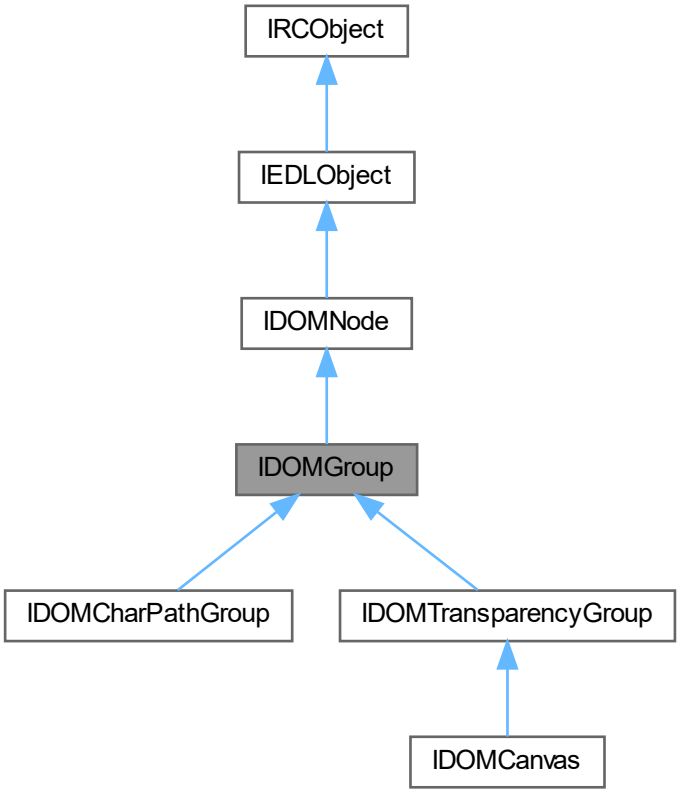
- idombrush.h

7.204 IDOMGroup Class Reference

Interface to DOM node representing Group Elements.

```
#include <idomgroup.h>
```

Inheritance diagram for IDOMGroup:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual const FMatrix & getRenderTransform () const =0

  Retrieves render transform matrix of the Group and its children.
- virtual void setRenderTransform (const FMatrix &matrix)=0

  Sets render transform matrix of the Group and its children.
- virtual IDOMPathGeometryPtr getClip () const =0

  Retrieves smart pointer to the clip.
- virtual void setClip (const IDOMPathGeometryPtr &ptrClip)=0

  Sets clip.
- virtual JawsMako::IOptionalContentDetailsPtr getOptionalContentDetails () const =0

  Get the JawsMako Optional Content details, or NULL if the group is not subject to optional content.
- virtual void setOptionalContentDetails (const JawsMako::IOptionalContentDetailsPtr &details)=0

Set the JawsMako Optional Content details for the group, or NULL to remove. Note that this is only allowed for bare IDOMGroup objects and not for subclasses. Any attempt to set optional content details on any object that is not an IDOMGroup (getNodeType() is eDOMGroupNode) will result in an exception. Note that an IDOMGroup may have optional content details or marked content details, but not both.

- virtual JawsMako::IMarkedContentDetailsPtr getMarkedContentDetails () const =0

  Get the JawsMako Marked Content details for this group, or NULL if the group is not marked.

- virtual void setMarkedContentDetails (const JawsMako::IMarkedContentDetailsPtr &details)=0

  Set the JawsMako Marked Content details for this group, or NULL to remove. Note that this is only allowed for bare IDOMGroup objects and not for subclasses. Any attempt to set marked content details on any object that is not an IDOMGroup (getNodeType() is eDOMGroupNode) will result in an exception. Note that an IDOMGroup may have optional content details or marked content details, but not both.

Public Member Functions inherited from IDOMNode

- virtual ~**IDOMNode** ()

  virtual destructor

- virtual DOMid getDOMid () const =0

  Retrieves the node ID.

- virtual void setDOMid (DOMid id)=0

  Sets the node ID.

- virtual eDOMNodeType getNodeType () const =0

  Retrieves the DOM node type.

- virtual bool getProperty (const EDLSysString &propertyName, PValue &propertyValue) const =0

  Retrieves the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.

- virtual void setProperty (const EDLSysString &propertyName, const PValue &propertyValue)=0

  Sets the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.

- virtual void removeProperty (const EDLSysString &propertyName)=0

  Removes property.

- virtual IEDLSysStringCollectionEnumPtr getPropertyCollectionEnum ()=0

  Retrieves a navigable list of the property names stored on this node.

- virtual bool hasChildNodes () const =0

  Function that indicates whether this node is a parent to other nodes.

- virtual IDOMNodePtr getParentNode () const =0

  Gets the parent node of this node.

- virtual IDOMNodePtr getFirstChild () const =0

  Gets the first child node of this node.

- virtual IDOMNodePtr getLastChild () const =0

  Gets the last child node of this node.

- virtual IDOMNodePtr getNextChild (const IDOMNodePtr &child) const =0

  Gets the child node which follows the node passed in.

- virtual IDOMNodePtr getPreviousChild (const IDOMNodePtr &child) const =0

  Gets the child node which precedes the node passed in.

- virtual IDOMNodePtr getPreviousSibling () const =0

  Retrieves the node's previous sibling node.

- virtual IDOMNodePtr getNextSibling () const =0

  Retrieves node's next sibling node.

- virtual void appendChild (const IDOMNodePtr &child)=0

  Appends a node to the end of the node's child list.

- virtual void insertChild (const IDOMNodePtr &ptrPreviousSibling, const IDOMNodePtr &child, bool bCheck↩ Complete=true)=0

  Insert a child node after ptrPreviousSibling.
- virtual IDOMNodePtr extractChild (const IDOMNodePtr &child)=0

  Extracts (that is, finds and removes) a child node from the node children. After extraction the child node is no longer a part of the DOM. If no node is specified, the first available node will be extracted from the node's children.
- virtual void replaceChild (const IDOMNodePtr &oldChild, const IDOMNodePtr &newChild)=0

  Replaces the child node with another.
- virtual bool isComplete () const =0

  Signals the completeness of the node.
  A complete node is one that has no more children to be added to it.
- virtual void **setComplete** ()=0

  Sets the node's completeness status to "true".
- virtual IDOMNodeFlags ∗ getFlags ()=0

  Retrieves the node's flags property.
- virtual void setParentNode (const IDOMNodePtr &ptrParent)=0

  Sets the parent node.
- virtual void setPreviousSibling (const IDOMNodePtr &ptrPreviousSibling)=0

  Sets the previous sibling node.
- virtual void setNextSibling (const IDOMNodePtr &ptrNextSibling)=0

  Sets the next sibling node.
- virtual bool isAncestor (const IDOMNodePtr &ptrCandidate)=0

  Function tests whether a candidate node is a descendant of the node.
- virtual FRect getBounds (bool applyTransform=true, bool applyClip=true)

  Find the conservative bounding box of the marking content of the node.
- virtual bool copyNodeData (IDOMNode ∗pSourceNode)=0

  Copy the properties collection, the flags and the DOM ID from the given source node to this one.
- virtual IDOMNodePtr cloneNode (IEDLClassFactory ∗pFactory) const =0

  Simpified node cloning. An exception of type IEDLError will be thrown on failure.
- virtual IDOMNodePtr cloneTree (IEDLClassFactory ∗pFactory) const =0

  Clone the tree of nodes beginning at this node. An exception of type IEDLError will be thrown on failure.
- virtual void cloneTreeAndAppend (IEDLClassFactory ∗pFactory, const IDOMNodePtr &dest) const =0

  Clone the tree of nodes beginning at this node, and append the result to the destination tree.
- virtual void **completeTree** ()=0

  Mark the entire tree from this point as complete. You should not ordinarily need to call this function.
- virtual void **removeCompleteFlagFromTree** ()=0

  Mark the entire tree from this point as complete.
- virtual void findChildrenOfType (eDOMNodeType type, CDOMNodeVect &nodes, bool searchForms=false)=0

  Find all children of this node with the given type, appending to the given vector. Does not descend into brushes.
- virtual void walkTree (WalkTreeFunc func, void ∗priv, bool descendIntoBrushes=false, bool descendInto↩ Forms=false)=0

  Walk through the DOM calling a given function on each node. The function is allowed to:
- virtual void notifyOnDestruct (NodeDeleteFunc func, void ∗priv)=0

  Register interest in being told when this node is about to be destroyed.
- virtual void unregisterNotify (NodeDeleteFunc func, void ∗priv)=0

  Unregister interest in being told when this node is about to be destroyed.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static EDL_API IDOMGroupPtr create (IEDLClassFactory *pFactory, const FMatrix &transform=FMatrix(), const IDOMPathGeometryPtr &clip=IDOMPathGeometryPtr())

  Simplified creation function for IDOMGroup. Throws an IEDLError exception on failure.
- static const CClassID & classID ()

  Retrieves class id of IDOMGroup.

Static Public Member Functions inherited from IDOMNode

- static EDL_API FMatrix effectiveTransformationOfNode (const IDOMNodePtr &node)

  Attempt to find the effective transformation matrix external to the specified node relative to either a containing page or ultimate parent.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.204.1 Detailed Description

Interface to DOM node representing Group Elements.

7.204.2 Member Function Documentation

classID()

```
static const CClassID & IDOMGroup::classID ( )  [inline], [static]
```

Retrieves class id of IDOMGroup.

Returns

CClassID class id of the element

create()

```
static EDL_API IDOMGroupPtr IDOMGroup::create (
            IEDLClassFactory * pFactory,
            const FMatrix & transform = FMatrix(),
            const IDOMPathGeometryPtr & clip = IDOMPathGeometryPtr() )  [static]
```

Simplified creation function for IDOMGroup. Throws an IEDLError exception on failure.

Parameters

pFactory	The EDL class factory
transform	The desired render transform
clip	The desired clip

Returns

IDOMGroupPtr The IDOMGroup

getClip()

```
virtual IDOMPathGeometryPtr IDOMGroup::getClip ( ) const  [pure virtual]
```

Retrieves smart pointer to the clip.

Returns

IDOMPathGeometryPtr Smart pointer to the clip

getMarkedContentDetails()

```
virtual JawsMako::IMarkedContentDetailsPtr IDOMGroup::getMarkedContentDetails ( ) const  [pure
virtual]
```

Get the JawsMako Marked Content details for this group, or NULL if the group is not marked.

Returns

JawsMako::IMarkedContentDetailsPtr A smart pointer to the Marked Content details

getOptionalContentDetails()

```
virtual JawsMako::IOptionalContentDetailsPtr IDOMGroup::getOptionalContentDetails ( ) const
[pure virtual]
```

Get the JawsMako Optional Content details, or NULL if the group is not subject to optional content.

Returns

JawsMako::IOptionalContentDetailsPtr A smart pointer to the optional content details

getRenderTransform()

```
virtual const FMatrix & IDOMGroup::getRenderTransform ( ) const  [pure virtual]
```

Retrieves render transform matrix of the Group and its children.

Returns

FMatrix The render transform matrix

setClip()

```
virtual void IDOMGroup::setClip (
            const IDOMPathGeometryPtr & ptrClip )  [pure virtual]
```

Sets clip.

Parameters

ptrClip	Smart pointer to clip

setMarkedContentDetails()

```
virtual void IDOMGroup::setMarkedContentDetails (
            const JawsMako::IMarkedContentDetailsPtr & details )  [pure virtual]
```

Set the JawsMako Marked Content details for this group, or NULL to remove. Note that this is only allowed for bare IDOMGroup objects and *not* for subclasses. Any attempt to set marked content details on any object that is not an IDOMGroup (getNodeType() is eDOMGroupNode) will result in an exception. Note that an IDOMGroup may have optional content details or marked content details, but not both.

Parameters

details	The Marked Content details

setOptionalContentDetails()

```
virtual void IDOMGroup::setOptionalContentDetails (
            const JawsMako::IOptionalContentDetailsPtr & details )  [pure virtual]
```

Set the JawsMako Optional Content details for the group, or NULL to remove. Note that this is only allowed for bare IDOMGroup objects and *not* for subclasses. Any attempt to set optional content details on any object that is not an IDOMGroup (getNodeType() is eDOMGroupNode) will result in an exception. Note that an IDOMGroup may have optional content details or marked content details, but not both.

Parameters

details	The Optional Content details

setRenderTransform()

```
virtual void IDOMGroup::setRenderTransform (
            const FMatrix & matrix )  [pure virtual]
```

Sets render transform matrix of the Group and its children.

Parameters

matrix	Render transform matrix

The documentation for this class was generated from the following file:

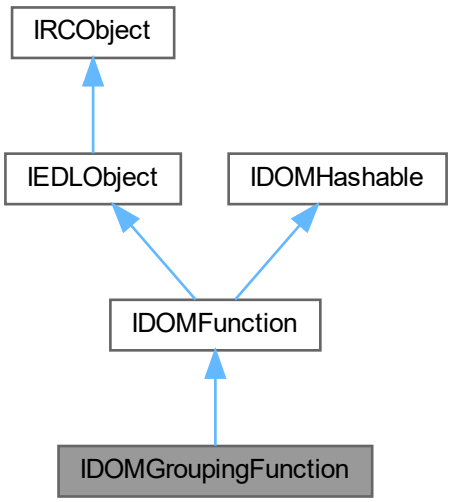
- idomgroup.h

7.205 IDOMGroupingFunction Class Reference

Interface to encapsulate an array of x-input-1-output functions.

```
#include <idomfunction.h>
```

Inheritance diagram for IDOMGroupingFunction:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual uint32 getNumFunctions () const =0

  Get the number of functions in the group.
- virtual IDOMFunctionPtr getFunctionAtIndex (uint32 index) const =0

  Get the function for a given function index.

Public Member Functions inherited from IDOMFunction

- virtual eFunctionType getFunctionType () const =0

  Retrieves function type.
- virtual uint32 getNumInputValues () const =0

  Get the number of input values that this function will operate on.
- virtual uint32 getNumOutputValues () const =0

  Get the number of output values that this function will produce.
- virtual void getInputDomain (uint32 inputNum, float &low, float &high) const =0

  Get the input domain for a given input to the function.
- virtual bool getOutputRange (uint32 outputNum, float &low, float &high) const =0

  Get the output range for a given input to the function.
- virtual void evaluate (const float *inputValues, float *outputValues) const =0

  Evaluate the input through the function and return the result.
- virtual void evaluate (const int32 *inputValues, float *outputValues) const =0

  Evaluate the input through the function and return the result.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves class id of IDOM.

Additional Inherited Members

Public Types inherited from IDOMFunction

- enum eFunctionType

  An enum for the various types of functions.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.205.1 Detailed Description

Interface to encapsulate an array of x-input-1-output functions.

```
This allows an array of \e n x-input-1-output functions to be treated as a
single x-input-n-output function. This is used in particular to represent
arrays of functions present in certain shading patterns.
```

7.205.2 Member Function Documentation

classID()

```
static const CClassID & IDOMGroupingFunction::classID ( )  [inline], [static]
```

Retrieves class id of IDOM.

Returns

CClassID class id of the element

getFunctionAtIndex()

```
virtual IDOMFunctionPtr IDOMGroupingFunction::getFunctionAtIndex (
            uint32 index ) const  [pure virtual]
```

Get the function for a given function index.

Parameters

index	A zero based index for the desired function.

Returns

IDOMFunctionPtr The function

getNumFunctions()

```
virtual uint32 IDOMGroupingFunction::getNumFunctions ( ) const  [pure virtual]
```

Get the number of functions in the group.

Returns

uint32 The number of functions

The documentation for this class was generated from the following file:

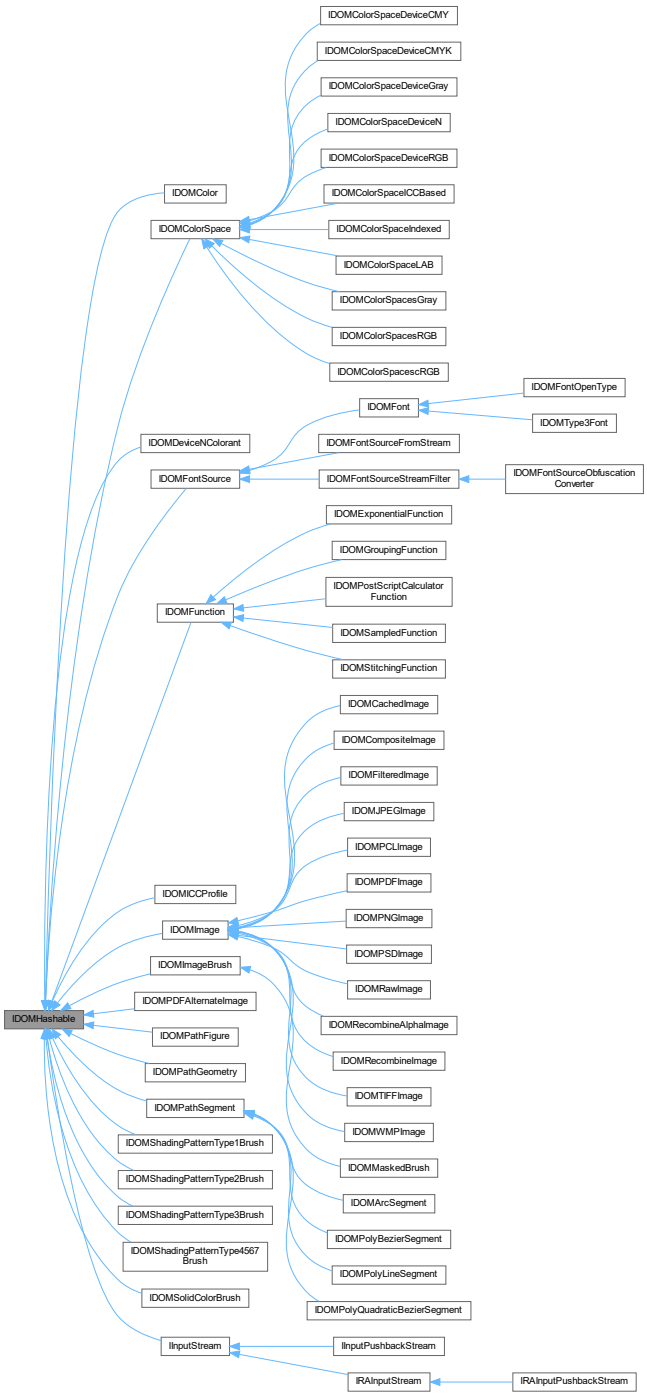
- idomfunction.h

7.206 IDOMHashable Class Reference

Abstract interface for objects that can be hashed.

`#include <idomhashable.h>`

Inheritance diagram for IDOMHashable:



Public Member Functions

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

7.206.1 Detailed Description

Abstract interface for objects that can be hashed.

Abstract interface for EDL objects that may be hashed.

Useful for determining a fast, 64-bit hash of objects that support this interface. Not guaranteed to be completely unique, but collisions are unlikely.

7.206.2 Member Function Documentation

hash()

```
virtual bool IDOMHashable::hash (
            uint64 & hash )  [pure virtual]
```

Retrieve a hash for this object.

Parameters

hash	A reference to receive the hash.

Returns

bool True on success, false if the call fails.

Implemented in IInputStream.

hashE()

```
virtual uint64 IDOMHashable::hashE ( )  [inline], [virtual]
```

As hash(), but throws an exception if the operation fails.

Returns

uint64 The hash.

The documentation for this class was generated from the following file:

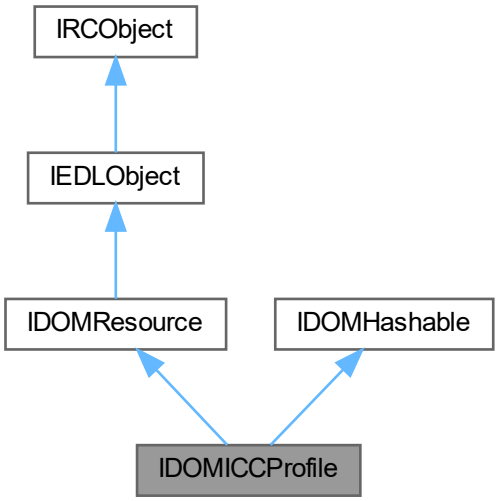
- idomhashable.h

7.207 IDOMICCProfile Class Reference

IDOMICCProfile interface.

`#include <idomresources.h>`

Inheritance diagram for IDOMICCProfile:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual void getProfileVersion (uint8 &majorVersion, uint8 &minorVersion)=0

  Get the ICC profile version. Throws an exception of type IEDLError on failure.
- virtual uint32 getProfileColorSpace ()=0

  Get the color space type for this ICC profile. Throws an exception of type IEDLError on failure.
- virtual EDLSysString getProfileName ()=0

  Get the name of this ICC profile. Throws an exception of type IEDLError on failure.

Public Member Functions inherited from IDOMResource

- virtual IInputStreamPtr getStream () const =0

  Retrieves the resource stream.
- virtual void setStream (const IInputStreamPtr &stream)=0

  Sets the resource stream for the node.
- virtual uint64 getStreamLength () const =0

  Retrieves the stream length, if it is available.
- virtual const EDLSysString & getUri () const =0

  Retrieves the resource URI.
- virtual void setUri (const EDLSysString &uri)=0

  Sets the resource URI.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static EDL_API IDOMICCProfilePtr create (IEDLClassFactory *pFactory, const IInputStreamPtr &stream, uint32 length=0, const EDLSysString &uri=EDLSysString())

  Creation function for an IDOMICCProfile Throws an IEDLError exception on failure.
- static const CClassID & classID ()

  Retrieves class id of IDOM.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.207.1 Detailed Description

IDOMICCProfile interface.

7.207.2 Member Function Documentation

classID()

```
static const CClassID & IDOMICCProfile::classID ( )  [inline], [static]
```

Retrieves class id of IDOM.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMICCProfilePtr IDOMICCProfile::create (
            IEDLClassFactory * pFactory,
            const IInputStreamPtr & stream,
            uint32 length = 0,
            const EDLSysString & uri = EDLSysString() )  [static]
```

Creation function for an IDOMICCProfile Throws an IEDLError exception on failure.

Parameters

pFactory	The class factory.
stream	The ICC profile stream.
length	The length of the profile stream. If 0, an attempt will be made to determine the length of the stream.
uri	The URI that should be associated with this profile, if required.

Returns

IDOMICCProfilePtr The new profile.

getProfileColorSpace()

```
virtual uint32 IDOMICCProfile::getProfileColorSpace ( )  [pure virtual]
```

Get the color space type for this ICC profile. Throws an exception of type IEDLError on failure.

Returns

uint32 The ICC color space signature, using the enumeration as per the ICC specification

getProfileName()

```
virtual EDLSysString IDOMICCProfile::getProfileName ( )  [pure virtual]
```

Get the name of this ICC profile. Throws an exception of type IEDLError on failure.

Returns

EDLSyString The profile name

getProfileVersion()

```
virtual void IDOMICCProfile::getProfileVersion (
            uint8 & majorVersion,
            uint8 & minorVersion )  [pure virtual]
```

Get the ICC profile version. Throws an exception of type IEDLError on failure.

Parameters

majorVersion	Reference to receive the major version.
minorVersion	Reference to receive the minor version.

The documentation for this class was generated from the following file:

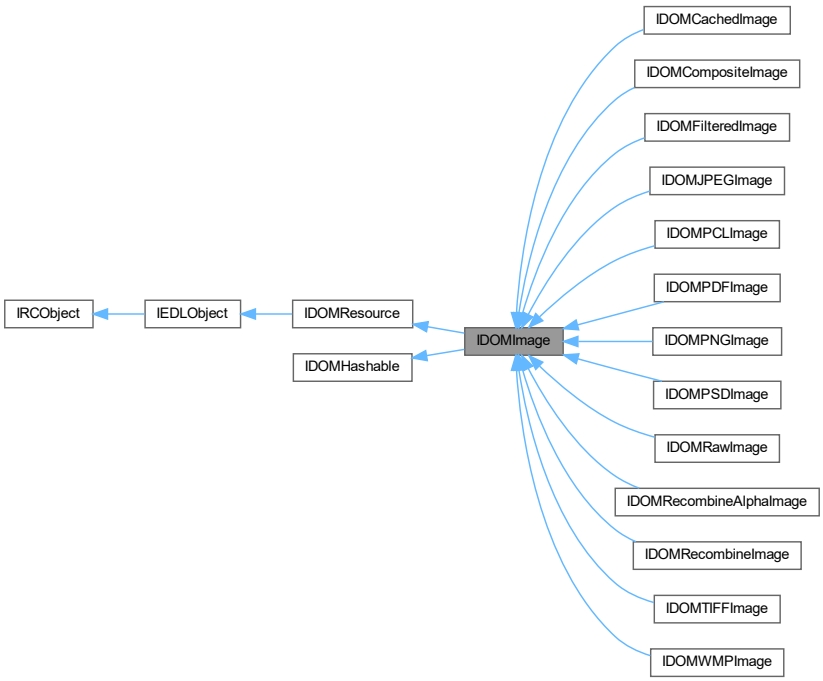
- idomresources.h

7.208 IDOMImage Class Reference

The base class describing an image. This class is subclassed to create a number of more specific image types. Instances of these objects may throw IEDLError exceptions on failure.

```
#include <idomimageresource.h>
```

Inheritance diagram for IDOMImage:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual IImageDecoderPtr createImageDecoder (IEDLClassFactory *factory, const IDOMImagePropertiesPtr &imageProperties)=0

  Creates a properly initialized image decoder object that reads from an inputstream that is specific to that image format.
- virtual IImageFramePtr getImageFrame (IEDLClassFactory *factory)

  Fetch the image frame; convenience.
- virtual IImageEncoderPtr createImageEncoder (const ISessionPtr &session, const IOutputStreamPtr &imageDest, const IDOMImagePropertiesPtr &imageProperties)=0

  Creates a properly initialized image encoder object that writes to an outputstream that is specific to that image format.
- virtual IDOMImagePropertiesPtr getImageProperties ()=0

  Returns an object that stores the properties for this image object. The properties can then be inspected (or more added) by clients that need to manipulate the image resource.
- virtual eDOMImageType getImageType ()=0

  Retrieves the image type.
- virtual bool getIsRendered ()=0

  Determine if the image is as a result of rendering. This is indicated if the image type is eDITRendered or if the image explicitly notes this is the case (such as for IDOMPDFImage).
- virtual IDOMImagePtr getImageWithSubstitutedColorSpace (IEDLClassFactory *factory, const IDOMColor↩SpacePtr &colorSpace)

  Obtain an image that is the same as this image, but with the colorspace substituted for another.

Public Member Functions inherited from IDOMResource

- virtual IInputStreamPtr getStream () const =0

  Retrieves the resource stream.
- virtual void setStream (const IInputStreamPtr &stream)=0

  Sets the resource stream for the node.
- virtual uint64 getStreamLength () const =0

  Retrieves the stream length, if it is available.
- virtual const EDLSysString & getUri () const =0

  Retrieves the resource URI.
- virtual void setUri (const EDLSysString &uri)=0

  Sets the resource URI.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.208.1 Detailed Description

The base class describing an image. This class is subclassed to create a number of more specific image types. Instances of these objects may throw IEDLError exceptions on failure.

7.208.2 Member Function Documentation

createImageDecoder()

```
virtual IImageDecoderPtr IDOMImage::createImageDecoder (
            IEDLClassFactory * factory,
            const IDOMImagePropertiesPtr & imageProperties )  [pure virtual]
```

Creates a properly initialized image decoder object that reads from an inputstream that is specific to that image format.

Parameters

factory	Pointer to the EDL class factory
imageProperties	Smart pointer to the DOM image properties interface, or NULL if there are no relevant properties to provide

Returns

IImageDecoderPtr The image decoder

createImageEncoder()

```
virtual IImageEncoderPtr IDOMImage::createImageEncoder (
            const ISessionPtr & session,
            const IOutputStreamPtr & imageDest,
            const IDOMImagePropertiesPtr & imageProperties )  [pure virtual]
```

Creates a properly initialized image encoder object that writes to an outputstream that is specific to that image format.

Parameters

session	Pointer to the EDL session
imageDest	Smart pointer to the output stream
imageProperties	Smart pointer to the DOM image properties interface, or NULL if there are no relevant properties to provide

Returns

IImageEncoderPtr The encoder

getImageFrame()

```
virtual IImageFramePtr IDOMImage::getImageFrame (
            IEDLClassFactory * factory )  [virtual]
```

Fetch the image frame; convenience.

Parameters

factory	Pointer to the EDL class factory

Returns

IImageFramePtr The frame.

getImageProperties()

```
virtual IDOMImagePropertiesPtr IDOMImage::getImageProperties ( )  [pure virtual]
```

Returns an object that stores the properties for this image object. The properties can then be inspected (or more added) by clients that need to manipulate the image resource.

Returns

IDOMImagePropertiesPtr The image properties

getImageType()

```
virtual eDOMImageType IDOMImage::getImageType ( )  [pure virtual]
```

Retrieves the image type.

Returns

eDOMImageType The image type

getImageWithSubstitutedColorSpace()

```
virtual IDOMImagePtr IDOMImage::getImageWithSubstitutedColorSpace (
            IEDLClassFactory * factory,
            const IDOMColorSpacePtr & colorSpace )  [virtual]
```

Obtain an image that is the same as this image, but with the colorspace substituted for another.

While this can be achieved by overlaying an IDOMFilteredImage with IDOMImageColorSpaceSubstitutionFilter, for some image types this can be done in a simpler manner by overriding the existing colour space. internally.

If the operation would make no changes, then the existing image is returned. Otherwise a modified clone will be made.

Parameters

factory	Pointer to the EDL class factory
colorSpace	The colorspace to substitute

Returns

IDOMImagePtr The new image with substituted space, or the existing image if no changes were required.

getIsRendered()

```
virtual bool IDOMImage::getIsRendered ( )  [pure virtual]
```

Determine if the image is as a result of rendering. This is indicated if the image type is eDITRendered or if the image explicitly notes this is the case (such as for IDOMPDFImage).

Returns

bool Whether or not this is a rendered image

The documentation for this class was generated from the following file:

- idomimageresource.h

7.209 IDOMImageBitScalerFilter Class Reference

An image filter that presents an image as an image with a different bits per sample.

`#include <idomimageresource.h>`

Inherits IDOMImageFilter.

Classes

- class Data
  Initialization data.

Static Public Member Functions

- static EDL_API IDOMImageBitScalerFilterPtr create (IEDLClassFactory *pFactory, uint8 bps)
  Simplified creator for a bit scaling filter.
- static const CClassID & classID ()
  Retrieves class id of IDOMImageBitScalerFilterClassID.

7.209.1 Detailed Description

An image filter that presents an image as an image with a different bits per sample.

7.209.2 Member Function Documentation

classID()

```
static const CClassID & IDOMImageBitScalerFilter::classID ( )  [inline], [static]
```

Retrieves class id of IDOMImageBitScalerFilterClassID.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMImageBitScalerFilterPtr IDOMImageBitScalerFilter::create (
            IEDLClassFactory * pFactory,
            uint8 bps )  [static]
```

Simplified creator for a bit scaling filter.

Parameters

pFactory	The EDL Class Factory
bps	The desired bits per sample (1, 3, 4, 8, 12 or 16 bit)

Returns

IDOMBitScalerFilterPtr The new filter.

The documentation for this class was generated from the following file:

- idomimageresource.h

7.210 IDOMImageBleederFilter Class Reference

An image filter that presents an image with the edge pixels repeated. Useful for cases where consumers may interpolate pixels at the edge, creating unwanted artifacts.

`#include <idomimageresource.h>`

Inherits IDOMImageFilter.

Classes

- class Data
  Initialization data.

Static Public Member Functions

- static EDL_API IDOMImageBleederFilterPtr create (IEDLClassFactory *pFactory, uint16 bleedPixels=1)
  Simplified creator for an image bleeder.
- static const CClassID & classID ()
  Retrieves class id of IDOMImageBleederFilter.

7.210.1 Detailed Description

An image filter that presents an image with the edge pixels repeated. Useful for cases where consumers may interpolate pixels at the edge, creating unwanted artifacts.

7.210.2 Member Function Documentation

classID()

```
static const CClassID & IDOMImageBleederFilter::classID ( )  [inline], [static]
```

Retrieves class id of IDOMImageBleederFilter.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMImageBleederFilterPtr IDOMImageBleederFilter::create (
            IEDLClassFactory * pFactory,
            uint16 bleedPixels = 1 )  [static]
```

Simplified creator for an image bleeder.

Parameters

pFactory	The EDL Class Factory
bleedPixels	The number of pixels to bleed the image.

Returns

IDOMImageBleederFilterPtr The new filter.

The documentation for this class was generated from the following file:

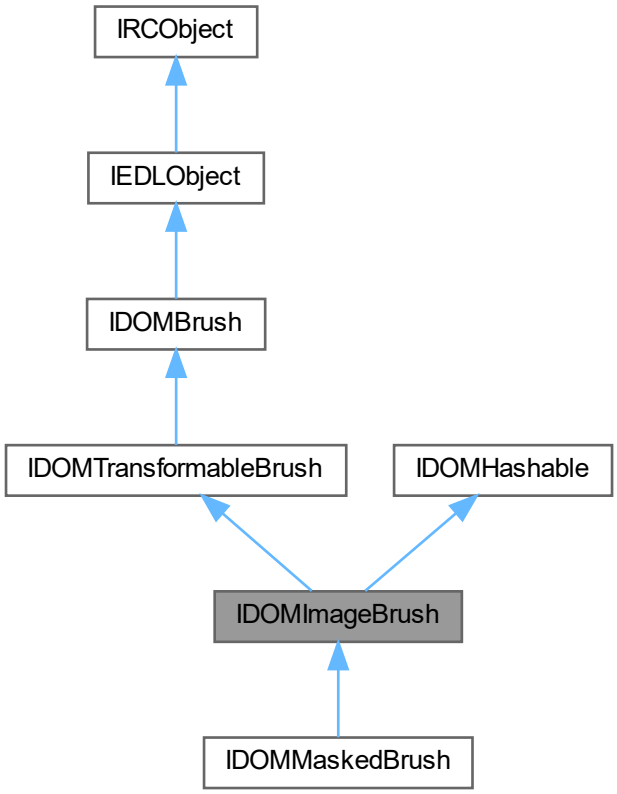
- idomimageresource.h

7.211 IDOMImageBrush Class Reference

Provides an interface to a DOM image brush object.

`#include <idombrush.h>`

Inheritance diagram for IDOMImageBrush:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual eTilingMode getTileMode () const =0

  Retrieves the tiling mode value of the image brush.
- virtual void setTileMode (eTilingMode tm)=0

  Sets the tiling mode of the image brush.
- virtual eViewUnits getViewBoxUnits () const =0

  Retrieves the viewbox units used by the image brush. Currently, only absolute units are supported.
- virtual void setViewBoxUnits (eViewUnits vu)=0

  Sets the viewbox units value of the image brush. Currently, only absolute units are supported.
- virtual eViewUnits getViewPortUnits () const =0

  Retrieves the viewport units value of the image brush. Currently, only absolute units are supported.
- virtual void setViewPortUnits (eViewUnits vu)=0

  Sets the viewport units used for the image brush. Currently, only absolute units are supported.
- virtual const FRect & getViewBox () const =0

  Retrieves the viewbox rectangle.
- virtual void setViewBox (const FRect &vb)=0

  Sets the viewbox rectangle.
- virtual const FRect & getViewPort () const =0

  Retrieves the viewport rectangle.
- virtual void setViewPort (const FRect &vp)=0

  Sets the viewport rectangle.
- virtual IDOMICCProfilePtr getICCProfile () const =0

  Retrieves the external ICC profile of the brush if present.
- virtual void setICCProfile (const IDOMICCProfilePtr &icc)=0

  Retrieves the external ICC profile of the brush if present.
- virtual IDOMImagePtr getImageSource () const =0

  Retrieves a smart pointer to the image resource.
- virtual void setImageSource (const IDOMImagePtr &ptrImageSource)=0

  Sets the image resource for the brush.
- virtual CDOMAlternateImageVect getAlternateImages () const =0

  Retrieves any alternate images associated with this brush.
- virtual void setAlternateImages (const CDOMAlternateImageVect &alternates)=0

  Set the alternate images associated with this brush.
- virtual JawsMako::IOptionalContentDetailsPtr getOptionalContentDetails () const =0

  Returns any optional content information that applies to this brush.
- virtual void setOptionalContentDetails (const JawsMako::IOptionalContentDetailsPtr &details)=0

  Set the optional content details that apply to this brush, or NULL to remove.
- virtual JawsMako::IPDFDictionaryPtr getPdfPropertiesDictionary () const =0

  Get the dictionary containing PDF properties attached to the image object. This dictionary will be as per the XObject dictionary in the original PDF, but with entries handled by existing DOM features stripped. In particular, the following entries described in the PDF specification will not be present in the dictionary:
- virtual void setPdfPropertiesDictionary (const JawsMako::IPDFDictionaryPtr &propertiesDictionary)=0

  Set the dictionary containing PDF properties attached to the image object. Please see getPdfPropertiesDictionary() above for information on the form of this dictionary.
- virtual IDOMTilingPatternBrushPtr getEquivalentTilingBrush (IEDLClassFactory *pFactory)=0

  Gets an equivalent IDOMTilingPattern brush. If the receiver has a tile mode of eNoTile, this call will fail.
- virtual IDOMVisualBrushPtr getEquivalentVisualBrush (IEDLClassFactory *pFactory)=0

  For tiled images, returns an equivalent visual brush containing the image without tiling. Will throw an exception if the image is not tiled. Not available for masked brushes as tiling is not supported for those brushes.

Public Member Functions inherited from IDOMTransformableBrush

- virtual const FMatrix & getRenderTransform () const =0

  Retrieves the render transform matrix.
- virtual void setRenderTransform (const FMatrix &matrix)=0

  Sets the render transform matrix.

Public Member Functions inherited from IDOMBrush

- virtual eBrushType getBrushType () const =0

  Retrieves the type of the brush.
- virtual float getOpacity () const =0

  Retrieves the opacity value of the brush element.
- virtual void setOpacity (float opc)=0

  Sets the opacity value of a brush element.
- virtual IDOMBrushPtr getAdjustedForUseInTransformedNode (IEDLClassFactory *pFactory, const FMatrix &nodeTransform)

  Get a version of this brush adjusted for use inside a node with the given transform.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static EDL_API IDOMImageBrushPtr create (IEDLClassFactory *pFactory, const IDOMImagePtr &image, const FRect &viewBox, const FRect &viewPort, const FMatrix &renderTransform=FMatrix(), float opacity=1.0f, eTilingMode tileMode=eNoTile, const CDOMAlternateImageVect &alternateImages=CDOMAlternateImageVect(), const JawsMako::IOptionalContentDetailsPtr &optionalContentDetails=JawsMako::IOptionalContentDetailsPtr(), const JawsMako::IPDFDictionaryPtr &propertiesDictionary=JawsMako::IPDFDictionaryPtr())

  Simplified creator for an image brush. Throws an IEDLError on failure.
- static const CClassID & classID ()

  Retrieves class id of IDOM.

Additional Inherited Members

Public Types inherited from IDOMBrush

- enum eBrushType {
  eSolidColor , eLinearGradient , eRadialGradient , eImage ,
  eMasked , eVisual , eSoftMask , eTilingPattern ,
  eType1ShadingPattern , eType2ShadingPattern , eType3ShadingPattern , eType4567ShadingPattern ,
  eNull }

  Brush type enumeration.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.211.1 Detailed Description

Provides an interface to a DOM image brush object.

An image brush is used to fill a region with an image. The image is defined in a coordinate space specified by the resolution of the image.

7.211.2 Member Function Documentation

classID()

```
static const CClassID & IDOMImageBrush::classID ( )  [inline], [static]
```

Retrieves class id of IDOM.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMImageBrushPtr IDOMImageBrush::create (
            IEDLClassFactory * pFactory,
            const IDOMImagePtr & image,
            const FRect & viewBox,
            const FRect & viewPort,
            const FMatrix & renderTransform = FMatrix(),
            float opacity = 1.0f,
            eTilingMode tileMode = eNoTile,
            const CDOMAlternateImageVect & alternateImages = CDOMAlternateImageVect(),
            const JawsMako::IOptionalContentDetailsPtr & optionalContentDetails = JawsMako:↩
:IOptionalContentDetailsPtr(),
            const JawsMako::IPDFDictionaryPtr & propertiesDictionary = JawsMako::IPDFDictionaryPtr()
)  [static]
```

Simplified creator for an image brush. Throws an IEDLError on failure.

Parameters

pFactory	The factory to use.
image	The image to use.
viewBox	The desired view box. If empty, the viewPort will be set to the full area of the image.
viewPort	The desired view port.
renderTransform	The desired render transform.
opacity	The desired brush opacity.
tileMode	The desired tile mode.
alternateImages	The PDF Alternate image brushes, if present.
optionalContentDetails	The optional content details. Ignored for tiling images
propertiesDictionary	Dictionary containing any additional PDF properties

Returns

IDOMImageBrushPtr The new brush.

getAlternateImages()

```
virtual CDOMAlternateImageVect IDOMImageBrush::getAlternateImages ( ) const  [pure virtual]
```

Retrieves any alternate images associated with this brush.

Returns

CDOMAlternateImageVect The alternate images, or an empty vector if there are no alternates.

getEquivalentTilingBrush()

```
virtual IDOMTilingPatternBrushPtr IDOMImageBrush::getEquivalentTilingBrush (
            IEDLClassFactory * pFactory )  [pure virtual]
```

Gets an equivalent IDOMTilingPattern brush. If the receiver has a tile mode of eNoTile, this call will fail.

Parameters

pFactory	A pointer to an EDL class factory

Returns

IDOMTilingPatternBrushPtr The resulting tiling brush. This brush will be cached, so do not edit; make a clone if the returned brush needs to be changed.

getEquivalentVisualBrush()

```
virtual IDOMVisualBrushPtr IDOMImageBrush::getEquivalentVisualBrush (
            IEDLClassFactory * pFactory )  [pure virtual]
```

For tiled images, returns an equivalent visual brush containing the image without tiling. Will throw an exception if the image is not tiled. Not available for masked brushes as tiling is not supported for those brushes.

Parameters

pFactory	A pointer to an EDL class factory

Returns

IDOMVisualBrushPtr The created visual brush. This brush will be cached, so do not edit; make a clone if the returned brush needs to be changed.

getICCProfile()

```
virtual IDOMICCProfilePtr IDOMImageBrush::getICCProfile ( ) const  [pure virtual]
```

Retrieves the external ICC profile of the brush if present.

Returns

IDOMICCProfilePtr The profile of the image, or NULL if no profile provided.

getImageSource()

```
virtual IDOMImagePtr IDOMImageBrush::getImageSource ( ) const  [pure virtual]
```

Retrieves a smart pointer to the image resource.

Returns

IDOMImagePtr the image resource

getOptionalContentDetails()

```
virtual JawsMako::IOptionalContentDetailsPtr IDOMImageBrush::getOptionalContentDetails ( )
const  [pure virtual]
```

Returns any optional content information that applies to this brush.

Returns

IOptionalContentDetailsPtr The optional content details, or NULL if optional content details do not apply

getPdfPropertiesDictionary()

```
virtual JawsMako::IPDFDictionaryPtr IDOMImageBrush::getPdfPropertiesDictionary ( ) const  [pure
virtual]
```

Get the dictionary containing PDF properties attached to the image object. This dictionary will be as per the XObject dictionary in the original PDF, but with entries handled by existing DOM features stripped. In particular, the following entries described in the PDF specification will not be present in the dictionary:

- Type
- Subtype
- Width
- Height
- ColorSpace
- BitsPerComponent
- Intent
- ImageMask
- Mask
- Decode
- Interpolate
- Alternates
- SMask
- SMaskInData
- Name
- OC

Returns

IPDFDictionaryPtr The properties dictionary, or NULL if not present. If non-null, the dictionary may be edited freely.

getTileMode()

```
virtual eTilingMode IDOMImageBrush::getTileMode ( ) const  [pure virtual]
```

Retrieves the tiling mode value of the image brush.

See also

eTilingMode

Returns

eTilingMode The image tiling mode.

getViewBox()

```
virtual const FRect & IDOMImageBrush::getViewBox ( ) const  [pure virtual]
```

Retrieves the viewbox rectangle.

The viewbox specifies the portion of a source image or visual to be rendered to the page as a tile, whose size and location are determined by the image brush's viewport. The tile is then used to fill the geometry specified by the parent element according to the image brush's tile mode. The ViewBox can specify a region larger than the image itself, including negative values. The view box specifies the position and dimension of the brush's source content. It is specified by four comma-separated real numbers (x, y, width, height) where width and height are non-negative.

Returns

FRect the viewbox rectangle

getViewBoxUnits()

```
virtual eViewUnits IDOMImageBrush::getViewBoxUnits ( ) const  [pure virtual]
```

Retrieves the viewbox units used by the image brush. Currently, only absolute units are supported.

See also

getViewBox()

setViewBox()

Returns

eViewUnits The viewbox units.

getViewPort()

```
virtual const FRect & IDOMImageBrush::getViewPort ( ) const  [pure virtual]
```

Retrieves the viewport rectangle.

The viewport specifies the dimensions and location of the initial tile that will be filled with the specified image or visual fragment. It is defined in the current effective coordinate space. It is specified by four comma separated real numbers (x, y, width, height) where width and height are non-negative.

Returns

FRect the viewport rectangle

getViewPortUnits()

```
virtual eViewUnits IDOMImageBrush::getViewPortUnits ( ) const  [pure virtual]
```

Retrieves the viewport units value of the image brush. Currently, only absolute units are supported.

See also

getViewPort()

setViewPort()

Returns

eViewUnits The viewport units.

setAlternateImages()

```
virtual void IDOMImageBrush::setAlternateImages (
            const CDOMAlternateImageVect & alternates )  [pure virtual]
```

Set the alternate images associated with this brush.

Parameters

alternates	The alternate images.

setICCProfile()

```
virtual void IDOMImageBrush::setICCProfile (
            const IDOMICCProfilePtr & icc )  [pure virtual]
```

Retrieves the external ICC profile of the brush if present.

Parameters

icc	The profile of the image, or NULL to clear

setImageSource()

```
virtual void IDOMImageBrush::setImageSource (
            const IDOMImagePtr & ptrImageSource )  [pure virtual]
```

Sets the image resource for the brush.

Parameters

ptrImageSource	Smart pointer to the new image resource.

setOptionalContentDetails()

```
virtual void IDOMImageBrush::setOptionalContentDetails (
            const JawsMako::IOptionalContentDetailsPtr & details )  [pure virtual]
```

Set the optional content details that apply to this brush, or NULL to remove.

Parameters

b details The optional content details, or NULL to remove

setPdfPropertiesDictionary()

```
virtual void IDOMImageBrush::setPdfPropertiesDictionary (
            const JawsMako::IPDFDictionaryPtr & propertiesDictionary )  [pure virtual]
```

Set the dictionary containing PDF properties attached to the image object. Please see getPdfPropertiesDictionary() above for information on the form of this dictionary.

Parameters

propertiesDictionary	The properties dictionary to set, or NULL to clear.

setTileMode()

```
virtual void IDOMImageBrush::setTileMode (
            eTilingMode tm )  [pure virtual]
```

Sets the tiling mode of the image brush.

Parameters

tm	The tiling mode value

setViewBox()

```
virtual void IDOMImageBrush::setViewBox (
            const FRect & vb )  [pure virtual]
```

Sets the viewbox rectangle.

The viewbox specifies the portion of a source image or visual to be rendered to the page as a tile. The tile is then used to fill the geometry specified by the parent element according to the TileMode() function. The viewbox can specify a region larger than the image itself, including negative values. The viewbox specifies the position and dimension of the brush's source content. It is specified by four comma-separated real numbers (x, y, width, height) where width and height are non-negative.

Parameters

vb	The viewbox rectangle

setViewBoxUnits()

```
virtual void IDOMImageBrush::setViewBoxUnits (
            eViewUnits vu )  [pure virtual]
```

Sets the viewbox units value of the image brush. Currently, only absolute units are supported.

See also

getViewBox()

setViewBox()

Parameters

vu	The viewbox units value

setViewPort()

```
virtual void IDOMImageBrush::setViewPort (
            const FRect & vp )  [pure virtual]
```

Sets the viewport rectangle.

The viewport specifies the dimensions and location of the initial tile that will be filled with the specified image or visual fragment. It is defined in the current effective coordinate space. It is specified by four comma separated real numbers (x, y, width, height) where width and height are non-negative.

Parameters

vp	The viewport rectangle

setViewPortUnits()

```
virtual void IDOMImageBrush::setViewPortUnits (
            eViewUnits vu )  [pure virtual]
```

Sets the viewport units used for the image brush. Currently, only absolute units are supported.

See also

getViewPort()
setViewPort()

Parameters

vu	The new viewport units value

The documentation for this class was generated from the following file:

- idombrush.h

7.212 IDOMImageChannelSelectorFilter Class Reference

An image filter that presents optionally an image stripped of alpha, or alternatively a Gray image representing the extra channel (ie Alpha or Mask) or a device color space channel.

```
#include <idomimageresource.h>
```

Inherits IDOMImageFilter.

Classes

- class Data
  Initialization data.

Public Types

- enum eSelection
  Enum for choosing which channel(s) to select during filtering.

Static Public Member Functions

- static EDL_API IDOMImageChannelSelectorFilterPtr create (IEDLClassFactory *pFactory, eSelection selection, uint8 channel=0)

  Simplified creator for a color converter filter.
- static const CClassID & classID ()

  Retrieves class id of IDOMImageChannelSelectorFilter.

7.212.1 Detailed Description

An image filter that presents optionally an image stripped of alpha, or alternatively a Gray image representing the extra channel (ie Alpha or Mask) or a device color space channel.

7.212.2 Member Function Documentation

classID()

```
static const CClassID & IDOMImageChannelSelectorFilter::classID ( )  [inline], [static]
```

Retrieves class id of IDOMImageChannelSelectorFilter.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMImageChannelSelectorFilterPtr IDOMImageChannelSelectorFilter::create (
            IEDLClassFactory * pFactory,
            eSelection selection,
            uint8 channel = 0 )  [static]
```

Simplified creator for a color converter filter.

Parameters

pFactory	The EDL Class Factory
selection	Which channel(s) to retain during filtering.
channel	The channel to select when using eSelectChannel.

Returns

IDOMImageChannelSelectorFilterPtr The new filter.

The documentation for this class was generated from the following file:

- idomimageresource.h

7.213 IDOMImageColorConverterFilter Class Reference

An image filter that presents a colour converted version of an image.

`#include <idomimageresource.h>`

Inherits IDOMImageFilter.

Classes

- class Data

  Initialization data.

Static Public Member Functions

- static EDL_API IDOMImageColorConverterFilterPtr create (IEDLClassFactory *pFactory, const IDOMColor↩ SpacePtr &space, eRenderingIntent intent, eBlackPointCompensation bpc)

  Simplified creator for a color converter filter.
- static const CClassID & classID ()

  Retrieves class id of IDOMImageColorConverterFilter.

7.213.1 Detailed Description

An image filter that presents a colour converted version of an image.

7.213.2 Member Function Documentation

classID()

```
static const CClassID & IDOMImageColorConverterFilter::classID ( )  [inline], [static]
```

Retrieves class id of IDOMImageColorConverterFilter.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMImageColorConverterFilterPtr IDOMImageColorConverterFilter::create (
            IEDLClassFactory * pFactory,
            const IDOMColorSpacePtr & space,
            eRenderingIntent intent,
            eBlackPointCompensation bpc )  [static]
```

Simplified creator for a color converter filter.

Parameters

pFactory	The EDL Class Factory
space	The target color space. Must be a base space; ie no DeviceN or Indexed spaces.
intent	The rendering intent to use
bpc	The desired black point compensation treatment. If in doubt, use eBPCDefault.

Returns

IDOMImageColorConverterFilterPtr The new filter.

The documentation for this class was generated from the following file:

- idomimageresource.h

7.214 IDOMImageColorKeyFilter Class Reference

An image filter that presents a masked image where colours within a given range are masked out, analogous to a green screen. The source image must not have a mask or alpha channel.

`#include <idomimageresource.h>`

Inherits IDOMImageFilter.

Classes

- class Data

  Initialization data.

Static Public Member Functions

- static EDL_API IDOMImageColorKeyFilterPtr create (IEDLClassFactory *pFactory, const CEDLVector< uint16 > &key)

  Simplified creator for a color key scaling filter.
- static const CClassID & classID ()

  Retrieves class id of IDOMImageColorKeyFilter.

7.214.1 Detailed Description

An image filter that presents a masked image where colours within a given range are masked out, analogous to a green screen. The source image must not have a mask or alpha channel.

7.214.2 Member Function Documentation

classID()

```
static const CClassID & IDOMImageColorKeyFilter::classID ( )  [inline], [static]
```

Retrieves class id of IDOMImageColorKeyFilter.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMImageColorKeyFilterPtr IDOMImageColorKeyFilter::create (
            IEDLClassFactory * pFactory,
            const CEDLVector< uint16 > & key )  [static]
```

Simplified creator for a color key scaling filter.

Parameters

pFactory	The EDL Class Factory
key	A vector that describes the range of colours to be masked out. In the form low1, hi1, low2, hi2, ... lowN, hiN where N is the number of color components in the source image. The values are expressed in the range of the samples in the image. For example, if the image is 4 bits per component then the values in this vector need to be between 0 and 15 inclusive.

Returns

IDOMImageColorKeyFilterPtr The new filter.

The documentation for this class was generated from the following file:

- idomimageresource.h

7.215 IDOMImageColorSpaceSubstitutionFilter Class Reference

An image filter that presents an identical image, just with the colourspace substituted.

```
#include <idomimageresource.h>
```

Inherits IDOMImageFilter.

Classes

- class Data

  Initialization data.

Static Public Member Functions

- static EDL_API IDOMImageColorSpaceSubstitutionFilterPtr create (IEDLClassFactory *pFactory, const IDOMColorSpacePtr &colorSpace)

  Simplified creator for a color space substitutor filter.
- static const CClassID & classID ()

  Retrieves class id of IDOMImageColorSpaceSubstitutionFilterClassID.

7.215.1 Detailed Description

An image filter that presents an identical image, just with the colourspace substituted.

7.215.2 Member Function Documentation

classID()

```
static const CClassID & IDOMImageColorSpaceSubstitutionFilter::classID ( )  [inline], [static]
```

Retrieves class id of IDOMImageColorSpaceSubstitutionFilterClassID.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMImageColorSpaceSubstitutionFilterPtr IDOMImageColorSpaceSubstitution↩
Filter::create (
            IEDLClassFactory * pFactory,
            const IDOMColorSpacePtr & colorSpace )  [static]
```

Simplified creator for a color space substitutor filter.

Parameters

pFactory	The EDL Class Factory
colorSpace	The color space to substitute.

Returns

IDOMImageColorSpaceSubstitutionFilterPtr The new filter.

The documentation for this class was generated from the following file:

- idomimageresource.h

7.216 IDOMImageDecodeFilter Class Reference

An image filter that applies a PDF/PS style Decode array to the image contents. For details on decode arrays, please see "Decode Arrays" on page 344 of the PDF Reference, version 1.7. The bit depth of the result may be promoted to eight or 16 bits per component depending on the situation.

`#include <idomimageresource.h>`

Inherits IDOMImageFilter.

Classes

- class Data

  Initialization data.

Static Public Member Functions

- static EDL_API IDOMImageDecodeFilterPtr create (IEDLClassFactory *pFactory, const CEDLVector< float > &decode)

  Simplified creator for a decode filter.
- static const CClassID & classID ()

  Retrieves class id of IDOMImageDecodeFilter.

7.216.1 Detailed Description

An image filter that applies a PDF/PS style Decode array to the image contents. For details on decode arrays, please see "Decode Arrays" on page 344 of the PDF Reference, version 1.7. The bit depth of the result may be promoted to eight or 16 bits per component depending on the situation.

7.216.2 Member Function Documentation

classID()

```
static const CClassID & IDOMImageDecodeFilter::classID ( )  [inline], [static]
```

Retrieves class id of IDOMImageDecodeFilter.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMImageDecodeFilterPtr IDOMImageDecodeFilter::create (
            IEDLClassFactory * pFactory,
            const CEDLVector< float > & decode )  [static]
```

Simplified creator for a decode filter.

Parameters

pFactory	The EDL Class Factory
decode	The decode vector.

Returns

IDOMImageDecodeFilterPtr The new filter.

The documentation for this class was generated from the following file:

- idomimageresource.h

7.217 IDOMImageDeindexerFilter Class Reference

An image filter that presents an image with an Indexed colour space as a simple eight bit image.

`#include <idomimageresource.h>`

Inherits IDOMImageFilter.

Classes

- class Data
  Initialization data.

Static Public Member Functions

- static EDL_API IDOMImageDeindexerFilterPtr create (IEDLClassFactory *pFactory)
  Simplified creator for a deindexer filter.
- static const CClassID & classID ()
  Retrieves class id of IDOMImageDeindexerFilter.

7.217.1 Detailed Description

An image filter that presents an image with an Indexed colour space as a simple eight bit image.

7.217.2 Member Function Documentation

classID()

```
static const CClassID & IDOMImageDeindexerFilter::classID ( )  [inline], [static]
```

Retrieves class id of IDOMImageDeindexerFilter.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMImageDeindexerFilterPtr IDOMImageDeindexerFilter::create (
            IEDLClassFactory * pFactory )  [static]
```

Simplified creator for a deindexer filter.

Parameters

pFactory	The EDL Class Factory

Returns

IDOMImageDeindexerFilterPtr The new filter.

The documentation for this class was generated from the following file:

- idomimageresource.h

7.218 IDOMImageDeviceNToBaseFilter Class Reference

An image filter that presents an image with a DeviceN colour space as a simple image in the alternate space.

`#include <idomimageresource.h>`

Inherits IDOMImageFilter.

Classes

- class Data
  Initialization data.

Static Public Member Functions

- static EDL_API IDOMImageDeviceNToBaseFilterPtr create (IEDLClassFactory *pFactory)
  Simplified creator for this filter.
- static const CClassID & classID ()
  Retrieves class id of IDOMImageDeindexerFilter.

7.218.1 Detailed Description

An image filter that presents an image with a DeviceN colour space as a simple image in the alternate space.

7.218.2 Member Function Documentation

classID()

```
static const CClassID & IDOMImageDeviceNToBaseFilter::classID ( )  [inline], [static]
```

Retrieves class id of IDOMImageDeindexerFilter.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMImageDeviceNToBaseFilterPtr IDOMImageDeviceNToBaseFilter::create (
            IEDLClassFactory * pFactory )  [static]
```

Simplified creator for this filter.

Parameters

pFactory	The EDL Class Factory

Returns

IDOMImageDeviceNToBaseFilterPtr The new filter.

The documentation for this class was generated from the following file:

- idomimageresource.h

7.219 IDOMImageDownsamplerFilter Class Reference

An image filter that presents a downsampled version of an image.

`#include <idomimageresource.h>`

Inherits IDOMImageFilter.

Classes

- class Data

  Initialization data.

Public Types

- enum eDownsamplingMethod

  The type of downsampling to be performed.

Static Public Member Functions

- static EDL_API IDOMImageDownsamplerFilterPtr create (IEDLClassFactory *pFactory, double xRes, double yRes=0.0, eDownsamplingMethod method=eSubsample)

  Simplified creator for an image downsampler filter.
- static const CClassID & classID ()

  Retrieves class id of IDOMImageDownsamplerFilter.

7.219.1 Detailed Description

An image filter that presents a downsampled version of an image.

7.219.2 Member Function Documentation

classID()

```
static const CClassID & IDOMImageDownsamplerFilter::classID ( )  [inline], [static]
```

Retrieves class id of IDOMImageDownsamplerFilter.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMImageDownsamplerFilterPtr IDOMImageDownsamplerFilter::create (
            IEDLClassFactory * pFactory,
            double xRes,
            double yRes = 0.0,
            eDownsamplingMethod method = eSubsample )  [static]
```

Simplified creator for an image downsampler filter.

Parameters

pFactory	The EDL Class Factory
xRes	The desired x resolution of the result.
yRes	The desired y resolution of the result. If 0, the x resolution will be used.
method	The desired downsampling method to use.

Returns

IDOMImageDownsamplerFilterPtr The new filter.

The documentation for this class was generated from the following file:

- idomimageresource.h

7.220 IDOMImageInverterFilter Class Reference

An image filter that presents a bitwise inverted form of the source image.

```
#include <idomimageresource.h>
```

Inherits IDOMImageFilter.

Classes

- class Data

  Initialization data.

Static Public Member Functions

- static EDL_API IDOMImageInverterFilterPtr create (IEDLClassFactory *pFactory)

  Simplified creator for this filter.
- static const CClassID & classID ()

  Retrieves class id of IDOMImageInverterFilter.

7.220.1 Detailed Description

An image filter that presents a bitwise inverted form of the source image.

7.220.2 Member Function Documentation

classID()

```
static const CClassID & IDOMImageInverterFilter::classID ( )  [inline], [static]
```

Retrieves class id of IDOMImageInverterFilter.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMImageInverterFilterPtr IDOMImageInverterFilter::create (
            IEDLClassFactory * pFactory )  [static]
```

Simplified creator for this filter.

Parameters

pFactory	The EDL Class Factory

Returns

IDOMImageInverterFilterPtr The new filter.

The documentation for this class was generated from the following file:

- idomimageresource.h

7.221 IDOMImageMaskExpanderFilter Class Reference

An image filter that presents a image source and color combination as a plain image with an alpha channel, with all pixels colored with the given color. Useful for simplifying an **IDOMMaskedBrush** where the brush masked by the image is a solid color.

`#include <idomimageresource.h>`

Inherits IDOMImageFilter.

Classes

- class Data

  Initialization data.

Static Public Member Functions

- static EDL_API IDOMImageMaskExpanderFilterPtr create (IEDLClassFactory *pFactory, const IDOMColor↩ Ptr &color)

  Simplified creator for a mask expander filter.
- static const CClassID & classID ()

  Retrieves class id of IDOMImageMaskExpanderFilter.

7.221.1 Detailed Description

An image filter that presents a image source and color combination as a plain image with an alpha channel, with all pixels colored with the given color. Useful for simplifying an **IDOMMaskedBrush** where the brush masked by the image is a solid color.

7.221.2 Member Function Documentation

classID()

```
static const CClassID & IDOMImageMaskExpanderFilter::classID ( )  [inline], [static]
```

Retrieves class id of IDOMImageMaskExpanderFilter.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMImageMaskExpanderFilterPtr IDOMImageMaskExpanderFilter::create (
            IEDLClassFactory * pFactory,
            const IDOMColorPtr & color )  [static]
```

Simplified creator for a mask expander filter.

Parameters

pFactory	The EDL Class Factory
color	The color to use.

Returns

IDOMImageMaskExpanderFilterPtr The new filter.

The documentation for this class was generated from the following file:

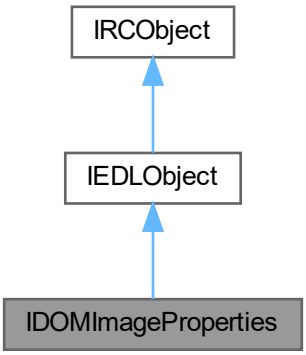
- idomimageresource.h

7.222 IDOMImageProperties Class Reference

The IDOMImageProperties interface provides access to an underlying implementation which stores miscellaneous information about the associated image.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMImageProperties:



Public Member Functions

- virtual void setProperty (const EDLString &name, const EDLString &value)=0
  Registers the name of a property and a string value for the property.
- virtual bool getProperty (const EDLString &name, EDLString &value)=0
  Retrieves the value of a named property as a string value.
- virtual void setBoolProperty (const EDLString &name, bool value)=0
  Registers the name of a property and a Boolean value for the property.
- virtual bool getBoolProperty (const EDLString &name, bool &value)=0
  Retrieves the value of a named property as a Boolean value.
- virtual void setIntProperty (const EDLString &name, int value)=0
  Registers the name of a property and an integer value for the property.
- virtual bool getIntProperty (const EDLString &name, int &value)=0
  Retrieves the value of a named property as an integer value.
- virtual void setObjectProperty (const EDLString &name, const IRCObjectPtr &object)=0
  Registers the name of a property and an IRCObject value for the property.
- virtual bool getObjectProperty (const EDLString &name, IRCObjectPtr &value)=0
  Retrieves the value of a named property as an IRCObject value.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves class id of IDOMImageProperties.
- static EDL_API IDOMImagePropertiesPtr create ()

  Create an empty image properties.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.222.1 Detailed Description

The IDOMImageProperties interface provides access to an underlying implementation which stores miscellaneous information about the associated image.

IDOMImageProperties holds a collection of name-value pairs which can be used to store arbitrary information about the image. The name must be unique within the collection, as it provides a key into it.

Methods are provided to allow the storage of integer, Boolean and string values.

A typical use would be to store information which would be expensive to compute-for example, whether or not the image is a noisy mask.

Instances of these objects may throw IEDLError exceptions on failure.

7.222.2 Member Function Documentation

classID()

```
static const CClassID & IDOMImageProperties::classID ( )  [inline], [static]
```

Retrieves class id of IDOMImageProperties.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMImagePropertiesPtr IDOMImageProperties::create ( )  [static]
```

Create an empty image properties.

Returns

IDOMImagePropertiesPtr The properties

getBoolProperty()

```
virtual bool IDOMImageProperties::getBoolProperty (
            const EDLString & name,
            bool & value )  [pure virtual]
```

Retrieves the value of a named property as a Boolean value.

Parameters

name	Property name.
value	Reference parameter to receive the property value.

Returns

bool True on success, false if property does not exist or is a different type. The value parameter will contain the property value.

getIntProperty()

```
virtual bool IDOMImageProperties::getIntProperty (
            const EDLString & name,
            int & value )  [pure virtual]
```

Retrieves the value of a named property as an integer value.

Parameters

name	Property name.
value	Reference parameter to receive the property value.

Returns

bool True on success, false if property does not exist or is a different type. The value parameter will contain the property value.

getObjectProperty()

```
virtual bool IDOMImageProperties::getObjectProperty (
            const EDLString & name,
            IRCObjectPtr & value )  [pure virtual]
```

Retrieves the value of a named property as an IRCObject value.

Parameters

name	Property name.
value	An IRCObject reference to receive the property value.

Returns

bool True on success, false if property does not exist or is a different type. The value parameter will contain the property value.

getProperty()

```
virtual bool IDOMImageProperties::getProperty (
            const EDLString & name,
            EDLString & value )  [pure virtual]
```

Retrieves the value of a named property as a string value.

Parameters

name	Property name.
value	Reference parameter to receive the property value.

Returns

bool True on success, false if property does not exist. The value parameter will contain the property value.

setBoolProperty()

```
virtual void IDOMImageProperties::setBoolProperty (
```

```
            const EDLString & name,
            bool value )  [pure virtual]
```

Registers the name of a property and a Boolean value for the property.

Parameters

name	Property name.
value	The Boolean value for the property.

setIntProperty()

```
virtual void IDOMImageProperties::setIntProperty (
            const EDLString & name,
            int value )  [pure virtual]
```

Registers the name of a property and an integer value for the property.

Parameters

name	Property name.
value	The integer value for the property.

setObjectProperty()

```
virtual void IDOMImageProperties::setObjectProperty (
            const EDLString & name,
            const IRCObjectPtr & object )  [pure virtual]
```

Registers the name of a property and an IRCObject value for the property.

Parameters

name	Property name.
object	An IRCObject reference to the value for the property.

setProperty()

```
virtual void IDOMImageProperties::setProperty (
            const EDLString & name,
            const EDLString & value )  [pure virtual]
```

Registers the name of a property and a string value for the property.

Parameters

name	Property name.
value	The string value for the property.

The documentation for this class was generated from the following file:

- idomimageresource.h

7.223 IDOMImageSpotMergerFilter Class Reference

An image filter that merges selected spot components into the process components of an image. This is conceptually similar to IDOMImageDeviceNToBaseFilter, but:

`#include <idomimageresource.h>`

Inherits IDOMImageFilter.

Classes

- class Data

  Initialization data.

Static Public Member Functions

- static EDL_API IDOMImageSpotMergerFilterPtr create (IEDLClassFactory *pFactory, const IDOMColorSpacePtr &inProcessSpace=IDOMColorSpacePtr(), const IDOMColorSpacePtr &outProcessSpace=IDOMColorSpacePtr(), bool mergeSpots=true, const IDOMColorSpaceDeviceN::CColorantInfoVect &spotsToMerge=IDOMColorSpaceDeviceN::CColorantInfoVect(), const JawsMako::CSpotColorNames &spotsToRetain=JawsMako::CSpotColorNames(), const JawsMako::CSpotColorNames &spotsToDrop=JawsMako::CSpotColorNames(), eRenderingIntent intent=eRelativeColorimetric, eBlackPointCompensation bpc=eBPCDefault)

  Simplified creator for a spot merging filter. Currently the result is always an 8bpc image.
- static EDL_API IDOMImageSpotMergerFilterPtr create (IEDLClassFactory *pFactory, const IDOMColorSpacePtr &inProcessSpace, bool mergeSpots, const IDOMColorSpaceDeviceN::CColorantInfoVect &spotsToMerge, const JawsMako::CSpotColorNames &spotsToRetain, const JawsMako::CSpotColorNames &spotsToDrop, const CProcessConversionVect &processConversions=CProcessConversionVect())

  Simplified creator for a spot merging filter, allowing multiple process color transformations to be performed. Currently the result is always an 8bpc image.
- static const CClassID & classID ()

  Retrieves class id of IDOMImageSpotMergerFilter.

7.223.1 Detailed Description

An image filter that merges selected spot components into the process components of an image. This is conceptually similar to IDOMImageDeviceNToBaseFilter, but:

- It allows only some components to be merged into the process components, and does *not* apply a tint transform. Instead, the spot colorants are merged into the process based on the color information provided via a kind of transparency blending.
- It allows some spot colorants to be dropped entirely.
- It provides a convenient method for dropping a single colorant while preserving all other spot components. While this is not merging per se, it can be useful.
- It optionally will convert process components to another color space.

7.223.2 Member Function Documentation

classID()

```
static const CClassID & IDOMImageSpotMergerFilter::classID ( )  [inline], [static]
```

Retrieves class id of IDOMImageSpotMergerFilter.

Returns

CClassID Class id of the element

create() [1/2]

```
static EDL_API IDOMImageSpotMergerFilterPtr IDOMImageSpotMergerFilter::create (
            IEDLClassFactory * pFactory,
            const IDOMColorSpacePtr & inProcessSpace,
            bool mergeSpots,
            const IDOMColorSpaceDeviceN::CColorantInfoVect & spotsToMerge,
            const JawsMako::CSpotColorNames & spotsToRetain,
            const JawsMako::CSpotColorNames & spotsToDrop,
            const CProcessConversionVect & processConversions = CProcessConversionVect() )
[static]
```

Simplified creator for a spot merging filter, allowing multiple process color transformations to be performed. Currently the result is always an 8bpc image.

Parameters

pFactory	The EDL Class Factory.
inProcessSpace	The process color space to be used when interpreting any process components in the source color space. If not provided, any process color information in the DeviceN space will be used (see IDOMColorSpaceDeviceN::getProcessColorSpace()) or failing that, DviceCMYK will be used. Please note that at this time only CMYK process components will be consulted, and as such, this must be a CMYK color space.
mergeSpots	If false, no spots will be merged (and #spotsToMerge is ignored).
spotsToMerge	The spots to merge. Consulted only if #mergeSpots is true. If empty, all spots will be merged, and the color of each component will be determined using a best effort method from the DeviceN color space of the source image. If non-empty, then the given spots are merged into the process components. The components in the info structures must use the same color space as the final output color space.
spotsToRetain	The names of colorants to retain, even if subject to mergeSpots or spotsToMerge. Useful in cases where it is known up front which spots to retain, but it is not known which spots to merge in order to achieve that aim. Otherwise, this can be left empty.
spotsToDrop	The names of colorants to drop entirely. If not provided, no spot components will be dropped. Please note that if all spots are dropped, and there are no process components, an error will result when the resulting frame is accessed.
processConversions	The series of conversions that will be performed on any process components in the image. This allows for process components to be converted through color spaces other than CMYK. However, the last outProcessSpace must be a CMYK color space. If there is more than one process conversion, then all must specify an outProcessSpace.

Returns

IDOMImageSpotMergerFilterPtr The new filter.

create() [2/2]

```
static EDL_API IDOMImageSpotMergerFilterPtr IDOMImageSpotMergerFilter::create (
            IEDLClassFactory * pFactory,
            const IDOMColorSpacePtr & inProcessSpace = IDOMColorSpacePtr(),
            const IDOMColorSpacePtr & outProcessSpace = IDOMColorSpacePtr(),
            bool mergeSpots = true,
            const IDOMColorSpaceDeviceN::CColorantInfoVect & spotsToMerge = IDOMColorSpaceDeviceN↩
::CColorantInfoVect(),
            const JawsMako::CSpotColorNames & spotsToRetain = JawsMako::CSpotColorNames(),
            const JawsMako::CSpotColorNames & spotsToDrop = JawsMako::CSpotColorNames(),
            eRenderingIntent intent = eRelativeColorimetric,
            eBlackPointCompensation bpc = eBPCDefault )  [static]
```

Simplified creator for a spot merging filter. Currently the result is always an 8bpc image.

Parameters

pFactory	The EDL Class Factory.
inProcessSpace	The process color space to be used when interpreting any process components in the source color space. If not provided, any process color information in the DeviceN space will be used (see IDOMColorSpaceDeviceN::getProcessColorSpace()) or failing that, DviceCMYK will be used. Please note that at this time only CMYK process components will be consulted, and as such, this must be a CMYK color space.
outProcessSpace	The process color space to generate. If not provided, the alternate color space of the image will be used. Currently, this color space must be subtractive - that is, it must be a CMYK color space. If the inProcessSpace does not match outProcessSpace, then process components will be converted from the inProcessSpace to the outProcessSpace before any spot merging takes place.
mergeSpots	If false, no spots will be merged (and #spotsToMerge is ignored).
spotsToMerge	The spots to merge. Consulted only if #mergeSpots is true. If empty, all spots will be merged, and the color of each component will be determined using a best effort method from the DeviceN color space of the source image. If non-empty, then the given spots are merged into the process components. The components in the info structures must use the same color space as the outProcessSpace.
spotsToRetain	The names of colorants to retain, even if subject to mergeSpots or spotsToMerge. Useful in cases where it is known up front which spots to retain, but it is not known which spots to merge in order to achieve that aim. Otherwise, this can be left empty.
spotsToDrop	The names of colorants to drop entirely. If not provided, no spot components will be dropped. Please note that if all spots are dropped, and there are no process components, an error will result when the resulting frame is accessed.
intent	The rendering intent to use if process components require color conversion.
bpc	The black-point compensation method to be used if process components require color conversion.

Returns

IDOMImageSpotMergerFilterPtr The new filter.

The documentation for this class was generated from the following file:

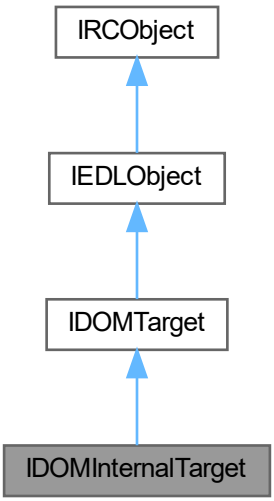
- idomimageresource.h

7.224 IDOMInternalTarget Class Reference

The IDOMInternalTarget interface describes the targets of hyperlinks that are in the same document but not on the current page.

`#include <idomtarget.h>`

Inheritance diagram for IDOMInternalTarget:



Public Member Functions

- virtual DOMid getTargetId () const =0

  Retrieves the internal target id.
- virtual void setTargetId (DOMid domId)=0

  Sets the internal target id.
- virtual eTargetType getTargetType () const

  Implementation of getTargetType for IDOMInternalTarget.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Public Types inherited from IDOMTarget

- enum eTargetType {
  eExternal , eInternal , ePage , ePageRect ,
  eActionGoToR , eActionGoToE , eActionLaunch , eActionThread ,
  eActionSound , eActionMovie , eActionHide , eActionNamed ,
  eActionSubmitForm , eActionResetForm , eActionImportData , eActionJavaScript ,
  eActionSetOCGState , eActionRendition , eActionTrans , eActionGoTo3DView ,
  eActionArray }

  An enumeration of target types.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.224.1 Detailed Description

The IDOMInternalTarget interface describes the targets of hyperlinks that are in the same document but not on the current page.

7.224.2 Member Function Documentation

getTargetId()

```
virtual DOMid IDOMInternalTarget::getTargetId ( ) const  [pure virtual]
```

Retrieves the internal target id.

Returns

DOMid. Returns the internal target id

getTargetType()

```
virtual eTargetType IDOMInternalTarget::getTargetType ( ) const  [inline], [virtual]
```

Implementation of getTargetType for IDOMInternalTarget.

Returns

eTargetType Returns eInternal;

Implements IDOMTarget.

setTargetId()

```
virtual void IDOMInternalTarget::setTargetId (
            DOMid domId )  [pure virtual]
```

Sets the internal target id.

Parameters

domId	The internal target id.

The documentation for this class was generated from the following file:

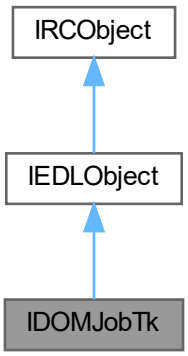
- idomtarget.h

7.225 IDOMJobTk Class Reference

Represents an EDL JobTicket.

`#include <idomjobtk.h>`

Inheritance diagram for IDOMJobTk:



Public Member Functions

- virtual void setOwner (IDOMJobTkOwnerPtr &ptrOwner)=0

  Sets the owner node, that is, the level of the DOMtree with which the JobTicket is associated.
- virtual IDOMJobTkOwnerPtr getOwner () const =0

  Retrieves the owner node, that is, the level of the DOMtree with which the JobTicket is associated.
- virtual void setContent (const IDOMJobTkContentPtr &ptrContent)=0

  Sets the JobTicket content node.
- virtual IDOMJobTkContentPtr getContent () const =0

  Retrieves the JobTicket content node.
- virtual IDOMJobTkNodePtr findChild (IDOMJobTkNode::eDOMJobTkNodeType nodeType, const EDLString &name, const EDLSysString &nmspace)=0

  Searches through the node's direct child set for the node type, name and namespace matching the information provided.
- virtual IDOMJobTkNodePtr findChild (IDOMJobTkNode::eDOMJobTkNodeType nodeType, const EDLQName &qname)=0

  Searches through the node's direct child set for the node type and qname matching the information provided.
- virtual bool getCombinedContent (IDOMJobTkContentPtr &ptrCombinedContent, bool addDefaultJob↩ Tk=false)=0

  Generate combined JobTicket content. Combined content is generated by combining the current JobTicket with all upper level JobTickets.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams ∗pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory ∗pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves the class id of IDOMJobTk.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()
  Virtual destructor.

7.225.1 Detailed Description

Represents an EDL JobTicket.

EDL JobTickets provide user intent and device configuration information to printers. JobTickets can be attached only to DocumentSequence, FixedDocument and FixedPage nodes. JobTickets can provide override settings to be used when printing the node to which they are attached.

7.225.2 Member Function Documentation

classID()

```
static const CClassID & IDOMJobTk::classID ( )  [inline], [static]
```

Retrieves the class id of IDOMJobTk.

Returns

CClassID. Class id of the element.

findChild() [1/2]

```
virtual IDOMJobTkNodePtr IDOMJobTk::findChild (
            IDOMJobTkNode::eDOMJobTkNodeType nodeType,
            const EDLQName & qname )  [pure virtual]
```

Searches through the node's direct child set for the node type and qname matching the information provided.

Parameters

nodeType	Node type of the the node's child to search for.
qname	Qualified name of the the node's child to search for.

Returns

IDOMJobTkNodePtr Smart pointer to the found child node.

findChild() [2/2]

```
virtual IDOMJobTkNodePtr IDOMJobTk::findChild (
            IDOMJobTkNode::eDOMJobTkNodeType nodeType,
```

```
            const EDLString & name,
            const EDLSysString & nmspace )  [pure virtual]
```

Searches through the node's direct child set for the node type, name and namespace matching the information provided.

Parameters

nodeType	Type of the node's child to search for.
name	Name of the node's child to search for.
nmspace	Namespace of the node's child to search for.

Returns

IDOMJobTkNodePtr Smart pointer to the found child node.

getCombinedContent()

```
virtual bool IDOMJobTk::getCombinedContent (
            IDOMJobTkContentPtr & ptrCombinedContent,
            bool addDefaultJobTk = false )  [pure virtual]
```

Generate combined JobTicket content. Combined content is generated by combining the current JobTicket with all upper level JobTickets.

Parameters

ptrCombinedContent	A smart pointer to receive the resulting combined content.
addDefaultJobTk	If this parameter is set to "true" then combine the default level of JobTicket as well as document, job and page levels.

Returns

bool. Returns true on success, false if the call fails.

getContent()

```
virtual IDOMJobTkContentPtr IDOMJobTk::getContent ( ) const  [pure virtual]
```

Retrieves the JobTicket content node.

Returns

IDOMJobTkContentPtr Returns smart pointer to the JobTicket content node

getOwner()

```
virtual IDOMJobTkOwnerPtr IDOMJobTk::getOwner ( ) const  [pure virtual]
```

Retrieves the owner node, that is, the level of the DOMtree with which the JobTicket is associated.

Returns

IDOMNodePtr. Returns a smart pointer to the owner node.

setContent()

```
virtual void IDOMJobTk::setContent (
            const IDOMJobTkContentPtr & ptrContent )  [pure virtual]
```

Sets the JobTicket content node.

Parameters

ptrContent	Smart pointer to the new JobTicket content node.

setOwner()

```
virtual void IDOMJobTk::setOwner (
            IDOMJobTkOwnerPtr & ptrOwner )  [pure virtual]
```

Sets the owner node, that is, the level of the DOMtree with which the JobTicket is associated.

Parameters

ptrOwner	Smart pointer to the new owner node.

The documentation for this class was generated from the following file:

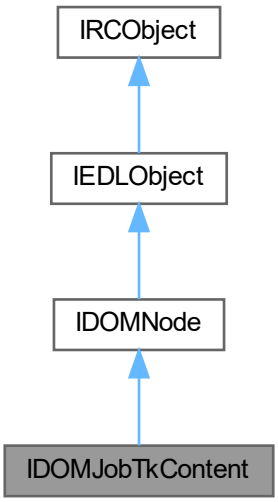
- idomjobtk.h

7.226 IDOMJobTkContent Class Reference

Represents the content element of the JobTicket.

```
#include <idomjobtk.h>
```

Inheritance diagram for IDOMJobTkContent:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual bool isValid () const =0

  Returns an indicator of the validity of the JobTicket content-that is, whether or not the JobTicket content has been initialised.
- virtual void setLevel (eDOMJobTkLevel level)=0

  Sets the level of the JobTicket content, corresponding to DocumentSequence, FixedDocument or FixedPage (see eDOMJobTkLevel). The default level is eDOMJobTkLevelDefault. The uninitialised value of the level is eDOMJob↩ TkLevelNotValid.
- virtual eDOMJobTkLevel getLevel () const =0

  Retrieves the level of the JobTicket content, corresponding to DocumentSequence, FixedDocument or FixedPage (see eDOMJobTkLevel). The default level is eDOMJobTkLevelDefault. The uninitialised value of the level is e↩ DOMJobTkLevelNotValid.
- virtual void setVersion (double version)=0

  Sets the version of the JobTicket content.
- virtual double getVersion () const =0

  Retrieves the version of the JobTicket content.
- virtual void setModified (bool modified)=0

  Sets the "modified" flag.
- virtual bool getModified () const =0

  Retrieves the value of the "modified" flag.
- virtual IDOMJobTkNodePtr getRootNode ()=0

  Retrieves the root node of the JobTicket content.

- virtual IEDLNamespaceCollectionEnumPtr getNamespaceCollectionEnum ()=0

  Retrieves the enumerator of the namespace collection.
- virtual uint32 getNamespacesCount ()=0

  Retrieves the number of namespaces in the namespace collection.
- virtual void **clearNamespaceCollection** ()=0

  Clears the collection of namespaces.
- virtual void addNamespace (const IEDLNamespacePtr &ptrNamespace)=0

  Appends a namespace to the collection of namespaces. Duplicate namespaces are allowed, providing that they at least have a different prefix. However, in this case only the existing namespace is registered; the new namespace prefix is merely aliased to it, and calling findNamespaceByPrefix() will result in the existing namespace being returned. Completely identical namespaces (prefix and namespace name) will result in an exception.
- virtual IEDLNamespacePtr addNamespace (const EDLSysString &prefix, const EDLSysString &name)=0

  Appends a namespace to the collection of namespaces. Duplicate namespaces are allowed, providing that they at least have a different prefix. However, in this case only the existing namespace is registered; the new namespace prefix is merely aliased to it, and here the existing namespace will be returned. Completely identical namespaces (prefix and namespace name) will result in a null pointer being returned.
- virtual IEDLNamespacePtr findNamespaceByName (const EDLSysString &nmspace)=0

  Finds a namespace in the namespace collection by name.
- virtual IEDLNamespacePtr findNamespaceByPrefix (const EDLSysString &prefix)=0

  Finds a namepace in the namespace collection by prefix.
- virtual void **addStandardNamespaces** ()=0

  Appends standard print ticket namespaces to the JobTicket content The following standard namespaces are appended: psk="http://schemas.microsoft.com/windows/2003/08/printing/printschemakeywords" xsd="http://www.↩ w3.org/2001/XMLSchema" xsi="http://www.w3.org/2001/XMLSchema-instance" psf="http://schemas.microsoft.↩ com/windows/2003/08/printing/printschemaframework".
- virtual EDLQName convertToQName (const EDLString &name, const EDLSysString &nmspace)=0

  Constructs a qualified name based on a name and namespace provided. The namespace must be present in the namespace collection.
- virtual IDOMJobTkNodePtr findChild (IDOMJobTkNode::eDOMJobTkNodeType nodeType, const EDLString &name, const EDLSysString &nmspace)=0

  Searches through the node's direct child set for a node matching the provided node type, name and namespace.
- virtual IDOMJobTkNodePtr findChild (IDOMJobTkNode::eDOMJobTkNodeType nodeType, const EDLQName &qname)=0

  Searches through the node's direct child set for a node matching provided the node type and qualified name.
- virtual bool loadFromFile (const EDLSysString &filename)=0

  Loads a JobTicket from a named XML file.
- virtual bool loadFromStream (const IRAInputStreamPtr &xmlStream)=0

  Loads a JobTicket from an XML stream.
- virtual bool loadFromInitString (const EDLSysString &sInit)=0

  Fills the JobTicket using input from the initialisation string.
- virtual void addParameterInit (const IEDLNamespacePtr &parameterNamespace, const EDLSysString &parameterName, const PValue &value)=0

  Adds a <psf:ParameterInit> element to the JobTicket.
- virtual void addFeature (const IEDLNamespacePtr &featureNamespace, const EDLSysString &featureName, const EDLQName &option)=0

  Adds Feature/Option elements to the JobTicket in the format:

Public Member Functions inherited from IDOMNode

- virtual ~**IDOMNode** ()

  virtual destructor
- virtual DOMid getDOMid () const =0

  Retrieves the node ID.

- virtual void setDOMid (DOMid id)=0

  Sets the node ID.
- virtual eDOMNodeType getNodeType () const =0

  Retrieves the DOM node type.
- virtual bool getProperty (const EDLSysString &propertyName, PValue &propertyValue) const =0

  Retrieves the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.
- virtual void setProperty (const EDLSysString &propertyName, const PValue &propertyValue)=0

  Sets the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.
- virtual void removeProperty (const EDLSysString &propertyName)=0

  Removes property.
- virtual IEDLSysStringCollectionEnumPtr getPropertyCollectionEnum ()=0

  Retrieves a navigable list of the property names stored on this node.
- virtual bool hasChildNodes () const =0

  Function that indicates whether this node is a parent to other nodes.
- virtual IDOMNodePtr getParentNode () const =0

  Gets the parent node of this node.
- virtual IDOMNodePtr getFirstChild () const =0

  Gets the first child node of this node.
- virtual IDOMNodePtr getLastChild () const =0

  Gets the last child node of this node.
- virtual IDOMNodePtr getNextChild (const IDOMNodePtr &child) const =0

  Gets the child node which follows the node passed in.
- virtual IDOMNodePtr getPreviousChild (const IDOMNodePtr &child) const =0

  Gets the child node which precedes the node passed in.
- virtual IDOMNodePtr getPreviousSibling () const =0

  Retrieves the node's previous sibling node.
- virtual IDOMNodePtr getNextSibling () const =0

  Retrieves node's next sibling node.
- virtual void appendChild (const IDOMNodePtr &child)=0

  Appends a node to the end of the node's child list.
- virtual void insertChild (const IDOMNodePtr &ptrPreviousSibling, const IDOMNodePtr &child, bool bCheck↩ Complete=true)=0

  Insert a child node after ptrPreviousSibling.
- virtual IDOMNodePtr extractChild (const IDOMNodePtr &child)=0

  Extracts (that is, finds and removes) a child node from the node children. After extraction the child node is no longer a part of the DOM. If no node is specified, the first available node will be extracted from the node's children.
- virtual void replaceChild (const IDOMNodePtr &oldChild, const IDOMNodePtr &newChild)=0

  Replaces the child node with another.
- virtual bool isComplete () const =0

  Signals the completeness of the node.
  A complete node is one that has no more children to be added to it.
- virtual void **setComplete** ()=0

  Sets the node's completeness status to "true".
- virtual IDOMNodeFlags * getFlags ()=0

  Retrieves the node's flags property.
- virtual void setParentNode (const IDOMNodePtr &ptrParent)=0

  Sets the parent node.
- virtual void setPreviousSibling (const IDOMNodePtr &ptrPreviousSibling)=0

Sets the previous sibling node.

- virtual void setNextSibling (const IDOMNodePtr &ptrNextSibling)=0

  Sets the next sibling node.
- virtual bool isAncestor (const IDOMNodePtr &ptrCandidate)=0

  Function tests whether a candidate node is a descendant of the node.
- virtual FRect getBounds (bool applyTransform=true, bool applyClip=true)

  Find the conservative bounding box of the marking content of the node.
- virtual bool copyNodeData (IDOMNode *pSourceNode)=0

  Copy the properties collection, the flags and the DOM ID from the given source node to this one.
- virtual IDOMNodePtr cloneNode (IEDLClassFactory *pFactory) const =0

  Simpified node cloning. An exception of type IEDLError will be thrown on failure.
- virtual IDOMNodePtr cloneTree (IEDLClassFactory *pFactory) const =0

  Clone the tree of nodes beginning at this node. An exception of type IEDLError will be thrown on failure.
- virtual void cloneTreeAndAppend (IEDLClassFactory *pFactory, const IDOMNodePtr &dest) const =0

  Clone the tree of nodes beginning at this node, and append the result to the destination tree.
- virtual void **completeTree** ()=0

  Mark the entire tree from this point as complete. You should not ordinarily need to call this function.
- virtual void **removeCompleteFlagFromTree** ()=0

  Mark the entire tree from this point as complete.
- virtual void findChildrenOfType (eDOMNodeType type, CDOMNodeVect &nodes, bool searchForms=false)=0

  Find all children of this node with the given type, appending to the given vector. Does not descend into brushes.
- virtual void walkTree (WalkTreeFunc func, void *priv, bool descendIntoBrushes=false, bool descendInto↩ Forms=false)=0

  Walk through the DOM calling a given function on each node. The function is allowed to:
- virtual void notifyOnDestruct (NodeDeleteFunc func, void *priv)=0

  Register interest in being told when this node is about to be destroyed.
- virtual void unregisterNotify (NodeDeleteFunc func, void *priv)=0

  Unregister interest in being told when this node is about to be destroyed.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves the class id of IDOMJobTkContent.
- static EDL_API EDLSysString addToInitString (const EDLSysString &paramName, const EDLSysString &initString, const EDLSysString &paramValue="", bool overwrite=true)

  Static helper to add a key/value pair to an initialisation string. Takes care of escaping as required.

Static Public Member Functions inherited from IDOMNode

- static EDL_API FMatrix effectiveTransformationOfNode (const IDOMNodePtr &node)

  Attempt to find the effective transformation matrix external to the specified node relative to either a containing page or ultimate parent.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.226.1 Detailed Description

Represents the content element of the JobTicket.

7.226.2 Member Function Documentation

addFeature()

```
virtual void IDOMJobTkContent::addFeature (
            const IEDLNamespacePtr & featureNamespace,
            const EDLSysString & featureName,
            const EDLQName & option )  [pure virtual]
```

Adds Feature/Option elements to the JobTicket in the format:

```
<psf:Feature name=parameterName>
    <psf:Option name=value/>
</psf:Feature>
```

Parameters

featureNamespace	Smart pointer to the feature name namespace.
featureName	Name of feature.
option	Qualified name of feature option.

addNamespace() [1/2]

```
virtual IEDLNamespacePtr IDOMJobTkContent::addNamespace (
            const EDLSysString & prefix,
            const EDLSysString & name )  [pure virtual]
```

Appends a namespace to the collection of namespaces. Duplicate namespaces are allowed, providing that they at least have a different prefix. However, in this case only the existing namespace is registered; the new namespace prefix is merely aliased to it, and here the existing namespace will be returned. Completely identical namespaces (prefix and namespace name) will result in a null pointer being returned.

Parameters

prefix	Value of the new namespace prefix
name	Value of the new namespace name

Returns

IEDLNamespacePtr. Smart pointer to the new namespace which has been added, or null on error.

addNamespace() [2/2]

```
virtual void IDOMJobTkContent::addNamespace (
            const IEDLNamespacePtr & ptrNamespace )  [pure virtual]
```

Appends a namespace to the collection of namespaces. Duplicate namespaces are allowed, providing that they at least have a different prefix. However, in this case only the existing namespace is registered; the new namespace prefix is merely aliased to it, and calling findNamespaceByPrefix() will result in the existing namespace being returned. Completely identical namespaces (prefix and namespace name) will result in an exception.

Parameters

ptrNamespace	Smart pointer to the new namespace.

addParameterInit()

```
virtual void IDOMJobTkContent::addParameterInit (
            const IEDLNamespacePtr & parameterNamespace,
            const EDLSysString & parameterName,
            const PValue & value )  [pure virtual]
```

Adds a <psf:ParameterInit> element to the JobTicket.

Parameters

parameterNamespace	Smart pointer to the parameter namespace.
parameterName	Name of parameter.
value	Value of parameter.

addToInitString()

```
static EDL_API EDLSysString IDOMJobTkContent::addToInitString (
            const EDLSysString & paramName,
            const EDLSysString & initString,
            const EDLSysString & paramValue = "",
            bool overwrite = true )  [static]
```

Static helper to add a key/value pair to an initialisation string. Takes care of escaping as required.

Parameters

paramName	The parameter name.
initString	The init string to add to
paramValue	The optional parameter value.
overwrite	Determines the behaviour if the parameter is already present in the initString. If true, the existing value will be replaced. If false the value will not be inserted.

Returns

EDLSysString Returns the new init string.

classID()

```
static const CClassID & IDOMJobTkContent::classID ( )  [inline], [static]
```

Retrieves the class id of IDOMJobTkContent.

Returns

CClassID. Returns the class id of the element.

convertToQName()

```
virtual EDLQName IDOMJobTkContent::convertToQName (
            const EDLString & name,
            const EDLSysString & nmspace )  [pure virtual]
```

Constructs a qualified name based on a name and namespace provided. The namespace must be present in the namespace collection.

Parameters

name	Name of JobTicket content element
nmspace	Namespace of JobTicket content element

Returns

EDLQName The constructed qualified name.

findChild() [1/2]

```
virtual IDOMJobTkNodePtr IDOMJobTkContent::findChild (
            IDOMJobTkNode::eDOMJobTkNodeType nodeType,
            const EDLQName & qname )  [pure virtual]
```

Searches through the node's direct child set for a node matching provided the node type and qualified name.

Parameters

nodeType	Job ticket node type.
qname	Qualified name of JobTicket node.

Returns

IDOMJobTkNodePtr. Returns smart pointer to the retrieved JobTicket node.

findChild() [2/2]

```
virtual IDOMJobTkNodePtr IDOMJobTkContent::findChild (
            IDOMJobTkNode::eDOMJobTkNodeType nodeType,
            const EDLString & name,
            const EDLSysString & nmspace )  [pure virtual]
```

Searches through the node's direct child set for a node matching the provided node type, name and namespace.

Parameters

nodeType	Job ticket node type.
name	Name of JobTicket node.
nmspace	Namespace of JobTicket node.

Returns

IDOMJobTkNodePtr. Returns smart pointer to the retrieved JobTicket node.

findNamespaceByName()

```
virtual IEDLNamespacePtr IDOMJobTkContent::findNamespaceByName (
            const EDLSysString & nmspace )  [pure virtual]
```

Finds a namespace in the namespace collection by name.

Parameters

nmspace	Value of namespace name.

Returns

IEDLNamespacePtr A smart pointer to the the namespace that was found.

findNamespaceByPrefix()

```
virtual IEDLNamespacePtr IDOMJobTkContent::findNamespaceByPrefix (
            const EDLSysString & prefix )  [pure virtual]
```

Finds a namepace in the namespace collection by prefix.

Parameters

prefix	Value of namespace prefix.

Returns

IEDLNamespacePtr Smart pointer to the namespace that was found

getLevel()

```
virtual eDOMJobTkLevel IDOMJobTkContent::getLevel ( ) const  [pure virtual]
```

Retrieves the level of the JobTicket content, corresponding to DocumentSequence, FixedDocument or FixedPage (see eDOMJobTkLevel). The default level is eDOMJobTkLevelDefault. The uninitialised value of the level is eDOMJobTkLevelNotValid.

Returns

eDOMJobTkLevel. Returns the level of the JobTicket content.

getModified()

```
virtual bool IDOMJobTkContent::getModified ( ) const  [pure virtual]
```

Retrieves the value of the "modified" flag.

Returns

bool. Value of the "modified" flag.

getNamespaceCollectionEnum()

```
virtual IEDLNamespaceCollectionEnumPtr IDOMJobTkContent::getNamespaceCollectionEnum ( )  [pure
virtual]
```

Retrieves the enumerator of the namespace collection.

Returns

IEDLNamespaceCollectionEnumPtr. Returns a pointer to the enumerator.

getNamespacesCount()

```
virtual uint32 IDOMJobTkContent::getNamespacesCount ( )  [pure virtual]
```

Retrieves the number of namespaces in the namespace collection.

Returns

uint32 Returns the number of namespaces in the collection.

getRootNode()

```
virtual IDOMJobTkNodePtr IDOMJobTkContent::getRootNode ( )  [pure virtual]
```

Retrieves the root node of the JobTicket content.

Returns

IDOMJobTkNodePtr The root node.

getVersion()

```
virtual double IDOMJobTkContent::getVersion ( ) const  [pure virtual]
```

Retrieves the version of the JobTicket content.

Returns

double. Returns the version of the JobTicket content.

isValid()

```
virtual bool IDOMJobTkContent::isValid ( ) const  [pure virtual]
```

Returns an indicator of the validity of the JobTicket content-that is, whether or not the JobTicket content has been initialised.

Returns

bool Returns true if the content has been initialised, false otherwise.

loadFromFile()

```
virtual bool IDOMJobTkContent::loadFromFile (
            const EDLSysString & filename )  [pure virtual]
```

Loads a JobTicket from a named XML file.

Parameters

filename	Path to and name of the JobTicket file to load

Returns

bool. Returns true on success, or false if the call fails.

loadFromInitString()

```
virtual bool IDOMJobTkContent::loadFromInitString (
            const EDLSysString & sInit )  [pure virtual]
```

Fills the JobTicket using input from the initialisation string.

Parameters

sInit	Initialisation string.

Returns

bool. Returns true on success, or false if the call fails.

loadFromStream()

```
virtual bool IDOMJobTkContent::loadFromStream (
            const IRAInputStreamPtr & xmlStream )  [pure virtual]
```

Loads a JobTicket from an XML stream.

Parameters

xmlStream	Stream for xml data

Returns

bool. Returns true on success, or false if the call fails.

setLevel()

```
virtual void IDOMJobTkContent::setLevel (
            eDOMJobTkLevel level )  [pure virtual]
```

Sets the level of the JobTicket content, corresponding to DocumentSequence, FixedDocument or FixedPage (see eDOMJobTkLevel). The default level is eDOMJobTkLevelDefault. The uninitialised value of the level is eDOMJob↩TkLevelNotValid.

Parameters

level	Job ticket level.

setModified()

```
virtual void IDOMJobTkContent::setModified (
            bool modified )  [pure virtual]
```

Sets the "modified" flag.

Parameters

modified	New value of the "modified" flag.

setVersion()

```
virtual void IDOMJobTkContent::setVersion (
            double version )  [pure virtual]
```

Sets the version of the JobTicket content.

Parameters

version	Job ticket version.

The documentation for this class was generated from the following file:

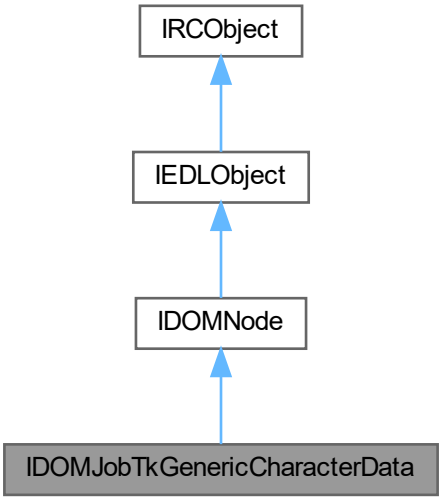
- idomjobtk.h

7.227 IDOMJobTkGenericCharacterData Class Reference

Interface to the IDOMJobTkGenericCharacterData node.

```
#include <idomjobtk.h>
```

Inheritance diagram for IDOMJobTkGenericCharacterData:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual EDLString getCharacterData () const =0

  Get the node's CDATA.
- virtual void setCharacterData (const EDLString &charData)=0

  Set the node's CDATA.
- virtual void appendCharacterData (const EDLString &charData)=0

  Append to the node's CDATA.

Public Member Functions inherited from IDOMNode

- virtual ~**IDOMNode** ()

  virtual destructor
- virtual DOMid getDOMid () const =0

  Retrieves the node ID.
- virtual void setDOMid (DOMid id)=0

  Sets the node ID.
- virtual eDOMNodeType getNodeType () const =0

  Retrieves the DOM node type.
- virtual bool getProperty (const EDLSysString &propertyName, PValue &propertyValue) const =0

Retrieves the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.

- virtual void setProperty (const EDLSysString &propertyName, const PValue &propertyValue)=0

  Sets the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.
- virtual void removeProperty (const EDLSysString &propertyName)=0

  Removes property.
- virtual IEDLSysStringCollectionEnumPtr getPropertyCollectionEnum ()=0

  Retrieves a navigable list of the property names stored on this node.
- virtual bool hasChildNodes () const =0

  Function that indicates whether this node is a parent to other nodes.
- virtual IDOMNodePtr getParentNode () const =0

  Gets the parent node of this node.
- virtual IDOMNodePtr getFirstChild () const =0

  Gets the first child node of this node.
- virtual IDOMNodePtr getLastChild () const =0

  Gets the last child node of this node.
- virtual IDOMNodePtr getNextChild (const IDOMNodePtr &child) const =0

  Gets the child node which follows the node passed in.
- virtual IDOMNodePtr getPreviousChild (const IDOMNodePtr &child) const =0

  Gets the child node which precedes the node passed in.
- virtual IDOMNodePtr getPreviousSibling () const =0

  Retrieves the node's previous sibling node.
- virtual IDOMNodePtr getNextSibling () const =0

  Retrieves node's next sibling node.
- virtual void appendChild (const IDOMNodePtr &child)=0

  Appends a node to the end of the node's child list.
- virtual void insertChild (const IDOMNodePtr &ptrPreviousSibling, const IDOMNodePtr &child, bool bCheck↩ Complete=true)=0

  Insert a child node after ptrPreviousSibling.
- virtual IDOMNodePtr extractChild (const IDOMNodePtr &child)=0

  Extracts (that is, finds and removes) a child node from the node children. After extraction the child node is no longer a part of the DOM. If no node is specified, the first available node will be extracted from the node's children.
- virtual void replaceChild (const IDOMNodePtr &oldChild, const IDOMNodePtr &newChild)=0

  Replaces the child node with another.
- virtual bool isComplete () const =0

  Signals the completeness of the node.
  A complete node is one that has no more children to be added to it.
- virtual void **setComplete** ()=0

  Sets the node's completeness status to "true".
- virtual IDOMNodeFlags * getFlags ()=0

  Retrieves the node's flags property.
- virtual void setParentNode (const IDOMNodePtr &ptrParent)=0

  Sets the parent node.
- virtual void setPreviousSibling (const IDOMNodePtr &ptrPreviousSibling)=0

  Sets the previous sibling node.
- virtual void setNextSibling (const IDOMNodePtr &ptrNextSibling)=0

  Sets the next sibling node.
- virtual bool isAncestor (const IDOMNodePtr &ptrCandidate)=0

  Function tests whether a candidate node is a descendant of the node.

- virtual FRect getBounds (bool applyTransform=true, bool applyClip=true)

  Find the conservative bounding box of the marking content of the node.
- virtual bool copyNodeData (IDOMNode *pSourceNode)=0

  Copy the properties collection, the flags and the DOM ID from the given source node to this one.
- virtual IDOMNodePtr cloneNode (IEDLClassFactory *pFactory) const =0

  Simpified node cloning. An exception of type IEDLError will be thrown on failure.
- virtual IDOMNodePtr cloneTree (IEDLClassFactory *pFactory) const =0

  Clone the tree of nodes beginning at this node. An exception of type IEDLError will be thrown on failure.
- virtual void cloneTreeAndAppend (IEDLClassFactory *pFactory, const IDOMNodePtr &dest) const =0

  Clone the tree of nodes beginning at this node, and append the result to the destination tree.
- virtual void **completeTree** ()=0

  Mark the entire tree from this point as complete. You should not ordinarily need to call this function.
- virtual void **removeCompleteFlagFromTree** ()=0

  Mark the entire tree from this point as complete.
- virtual void findChildrenOfType (eDOMNodeType type, CDOMNodeVect &nodes, bool searchForms=false)=0

  Find all children of this node with the given type, appending to the given vector. Does not descend into brushes.
- virtual void walkTree (WalkTreeFunc func, void *priv, bool descendIntoBrushes=false, bool descendInto↩ Forms=false)=0

  Walk through the DOM calling a given function on each node. The function is allowed to:
- virtual void notifyOnDestruct (NodeDeleteFunc func, void *priv)=0

  Register interest in being told when this node is about to be destroyed.
- virtual void unregisterNotify (NodeDeleteFunc func, void *priv)=0

  Unregister interest in being told when this node is about to be destroyed.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves the class id of IDOMJobTkGenericNode.

Static Public Member Functions inherited from IDOMNode

- static EDL_API FMatrix effectiveTransformationOfNode (const IDOMNodePtr &node)

  Attempt to find the effective transformation matrix external to the specified node relative to either a containing page or ultimate parent.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.227.1 Detailed Description

Interface to the IDOMJobTkGenericCharacterData node.

Objects of type IDOMJobTkGenericCharacterData store CDATA associated with an IDOMJobTkGenericNode. Objects of this type are stored as children of IDOMJobTkGenericNodes.

7.227.2 Member Function Documentation

appendCharacterData()

```
virtual void IDOMJobTkGenericCharacterData::appendCharacterData (
            const EDLString & charData )  [pure virtual]
```

Append to the node's CDATA.

Parameters

charData	The additional CDATA as an EDLString

classID()

```
static const CClassID & IDOMJobTkGenericCharacterData::classID ( )  [inline], [static]
```

Retrieves the class id of IDOMJobTkGenericNode.

Returns

CClassID. Class id of the element.

getCharacterData()

```
virtual EDLString IDOMJobTkGenericCharacterData::getCharacterData ( ) const  [pure virtual]
```

Get the node's CDATA.

Returns

EDLString The node CDATA

setCharacterData()

```
virtual void IDOMJobTkGenericCharacterData::setCharacterData (
            const EDLString & charData )  [pure virtual]
```

Set the node's CDATA.

Parameters

charData	The CDATA as an EDLString

The documentation for this class was generated from the following file:

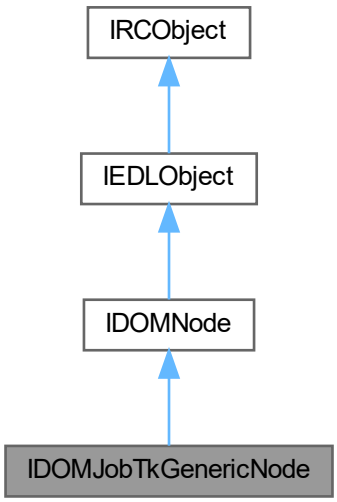
- idomjobtk.h

7.228 IDOMJobTkGenericNode Class Reference

Interface to the IDOMJobTkGenericNode node.

```
#include <idomjobtk.h>
```

Inheritance diagram for IDOMJobTkGenericNode:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual EDLQName getQName () const =0

  Retrieves the node qname.
- virtual void addAttribute (const EDLQName &qname, const EDLSysString &value)=0

  Adds an attribute.
- virtual uint32 getNumAttributes () const =0

  Gets the number of attributes.
- virtual EDLSysString getAttributeAtIndex (uint32 index, EDLQName &qname) const =0

  Get an attribute at the given index.

Public Member Functions inherited from IDOMNode

- virtual ~**IDOMNode** ()

  virtual destructor
- virtual DOMid getDOMid () const =0

  Retrieves the node ID.
- virtual void setDOMid (DOMid id)=0

  Sets the node ID.
- virtual eDOMNodeType getNodeType () const =0

  Retrieves the DOM node type.

- virtual bool getProperty (const EDLSysString &propertyName, PValue &propertyValue) const =0

  Retrieves the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.
- virtual void setProperty (const EDLSysString &propertyName, const PValue &propertyValue)=0

  Sets the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.
- virtual void removeProperty (const EDLSysString &propertyName)=0

  Removes property.
- virtual IEDLSysStringCollectionEnumPtr getPropertyCollectionEnum ()=0

  Retrieves a navigable list of the property names stored on this node.
- virtual bool hasChildNodes () const =0

  Function that indicates whether this node is a parent to other nodes.
- virtual IDOMNodePtr getParentNode () const =0

  Gets the parent node of this node.
- virtual IDOMNodePtr getFirstChild () const =0

  Gets the first child node of this node.
- virtual IDOMNodePtr getLastChild () const =0

  Gets the last child node of this node.
- virtual IDOMNodePtr getNextChild (const IDOMNodePtr &child) const =0

  Gets the child node which follows the node passed in.
- virtual IDOMNodePtr getPreviousChild (const IDOMNodePtr &child) const =0

  Gets the child node which precedes the node passed in.
- virtual IDOMNodePtr getPreviousSibling () const =0

  Retrieves the node's previous sibling node.
- virtual IDOMNodePtr getNextSibling () const =0

  Retrieves node's next sibling node.
- virtual void appendChild (const IDOMNodePtr &child)=0

  Appends a node to the end of the node's child list.
- virtual void insertChild (const IDOMNodePtr &ptrPreviousSibling, const IDOMNodePtr &child, bool bCheck↩ Complete=true)=0

  Insert a child node after ptrPreviousSibling.
- virtual IDOMNodePtr extractChild (const IDOMNodePtr &child)=0

  Extracts (that is, finds and removes) a child node from the node children. After extraction the child node is no longer a part of the DOM. If no node is specified, the first available node will be extracted from the node's children.
- virtual void replaceChild (const IDOMNodePtr &oldChild, const IDOMNodePtr &newChild)=0

  Replaces the child node with another.
- virtual bool isComplete () const =0

  Signals the completeness of the node.
  A complete node is one that has no more children to be added to it.
- virtual void **setComplete** ()=0

  Sets the node's completeness status to "true".
- virtual IDOMNodeFlags * getFlags ()=0

  Retrieves the node's flags property.
- virtual void setParentNode (const IDOMNodePtr &ptrParent)=0

  Sets the parent node.
- virtual void setPreviousSibling (const IDOMNodePtr &ptrPreviousSibling)=0

  Sets the previous sibling node.
- virtual void setNextSibling (const IDOMNodePtr &ptrNextSibling)=0

  Sets the next sibling node.
- virtual bool isAncestor (const IDOMNodePtr &ptrCandidate)=0

Function tests whether a candidate node is a descendant of the node.

- virtual FRect getBounds (bool applyTransform=true, bool applyClip=true)

  Find the conservative bounding box of the marking content of the node.
- virtual bool copyNodeData (IDOMNode *pSourceNode)=0

  Copy the properties collection, the flags and the DOM ID from the given source node to this one.
- virtual IDOMNodePtr cloneNode (IEDLClassFactory *pFactory) const =0

  Simpified node cloning. An exception of type IEDLError will be thrown on failure.
- virtual IDOMNodePtr cloneTree (IEDLClassFactory *pFactory) const =0

  Clone the tree of nodes beginning at this node. An exception of type IEDLError will be thrown on failure.
- virtual void cloneTreeAndAppend (IEDLClassFactory *pFactory, const IDOMNodePtr &dest) const =0

  Clone the tree of nodes beginning at this node, and append the result to the destination tree.
- virtual void **completeTree** ()=0

  Mark the entire tree from this point as complete. You should not ordinarily need to call this function.
- virtual void **removeCompleteFlagFromTree** ()=0

  Mark the entire tree from this point as complete.
- virtual void findChildrenOfType (eDOMNodeType type, CDOMNodeVect &nodes, bool searchForms=false)=0

  Find all children of this node with the given type, appending to the given vector. Does not descend into brushes.
- virtual void walkTree (WalkTreeFunc func, void *priv, bool descendIntoBrushes=false, bool descendInto↩ Forms=false)=0

  Walk through the DOM calling a given function on each node. The function is allowed to:
- virtual void notifyOnDestruct (NodeDeleteFunc func, void *priv)=0

  Register interest in being told when this node is about to be destroyed.
- virtual void unregisterNotify (NodeDeleteFunc func, void *priv)=0

  Unregister interest in being told when this node is about to be destroyed.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves the class id of IDOMJobTkGenericNode.

Static Public Member Functions inherited from IDOMNode

- static EDL_API FMatrix effectiveTransformationOfNode (const IDOMNodePtr &node)

  Attempt to find the effective transformation matrix external to the specified node relative to either a containing page or ultimate parent.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.228.1 Detailed Description

Interface to the IDOMJobTkGenericNode node.

IDOMJobTkGenericNode is a print ticket node designed to represent unhandled XML data in a print ticket. This allows such markup to be retained, interrogated, modified and subsequently written faithfully in a output print ticket. During XPS input processing, a node of this type is created for each tag that is not natively understood by EDL. The QName is the name of the tag (including namespace information) and the attributes are stored within the object.

Nodes of this type may have IDOMJobTkGenericCharacterData children which represent CDATA owned by the tag.

7.228.2 Member Function Documentation

addAttribute()

```
virtual void IDOMJobTkGenericNode::addAttribute (
            const EDLQName & qname,
            const EDLSysString & value )  [pure virtual]
```

Adds an attribute.

Parameters

qname	Qualified name of the attribute
value	The value of the attribute as a string

classID()

```
static const CClassID & IDOMJobTkGenericNode::classID ( )  [inline], [static]
```

Retrieves the class id of IDOMJobTkGenericNode.

Returns

CClassID. Class id of the element.

getAttributeAtIndex()

```
virtual EDLSysString IDOMJobTkGenericNode::getAttributeAtIndex (
            uint32 index,
            EDLQName & qname ) const  [pure virtual]
```

Get an attribute at the given index.

Parameters

index	The index of the attribute, with 0 representing the first attribute
qname	Reference to receive the qualified name of the attribute

Returns

EDLSysString The attribute as a string

getNumAttributes()

```
virtual uint32 IDOMJobTkGenericNode::getNumAttributes ( ) const  [pure virtual]
```

Gets the number of attributes.

Returns

uint32 The number of attributes

getQName()

```
virtual EDLQName IDOMJobTkGenericNode::getQName ( ) const  [pure virtual]
```

Retrieves the node qname.

Returns

EDLQName Qualified name value

The documentation for this class was generated from the following file:

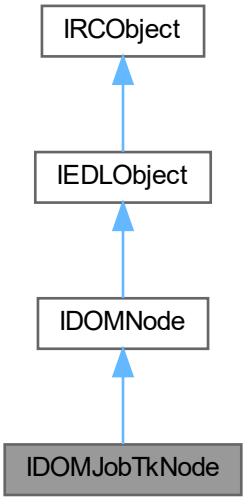
- idomjobtk.h

7.229 IDOMJobTkNode Class Reference

Represents a Job Ticket Node.

`#include <idomjobtk.h>`

Inheritance diagram for IDOMJobTkNode:



Classes

- class Data
  Initialization data.

Public Types

- enum eDOMJobTkNodeType
  DOMJobTk node (Property, Feature, Option, InitParam, ScoredProperty, ParamRef)

Public Member Functions

- virtual void setQName (const EDLQName &qname)=0
  Sets the qualified name of the node.
- virtual EDLQName getQName () const =0
  Retrieves the node qname.
- virtual EDLString getQNameAsString () const =0
  Retrieves the full qualified name of the node as a string.
- virtual void setJobTkNodeType (eDOMJobTkNodeType nodeType)=0
  Sets the node type.
- virtual eDOMJobTkNodeType getJobTkNodeType ()=0

Retrieves the node type.

- virtual IDOMJobTkContentPtr getJobTkContent ()=0

  Returns the JobTicket content of this node, by following up the parent node chain.

- virtual IDOMJobTkValuePtr getChildValue ()=0

  If JobTicket node has child with type eDOMJobTkPTNodeValue then returns this child, otherwise returns NULL.

- virtual IDOMJobTkNodePtr findChild (eDOMJobTkNodeType nodeType, const EDLString &name, const EDLSysString &nmspace)=0

  Searches through the node's direct child set for a node matching the provided node type, name and namespace.

- virtual IDOMJobTkNodePtr findChild (eDOMJobTkNodeType nodeType, const EDLQName &qname)=0

  Searches through the node's direct child set for a node matching the provided node type and qname.

Public Member Functions inherited from IDOMNode

- virtual ~**IDOMNode** ()

  virtual destructor

- virtual DOMid getDOMid () const =0

  Retrieves the node ID.

- virtual void setDOMid (DOMid id)=0

  Sets the node ID.

- virtual eDOMNodeType getNodeType () const =0

  Retrieves the DOM node type.

- virtual bool getProperty (const EDLSysString &propertyName, PValue &propertyValue) const =0

  Retrieves the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.

- virtual void setProperty (const EDLSysString &propertyName, const PValue &propertyValue)=0

  Sets the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.

- virtual void removeProperty (const EDLSysString &propertyName)=0

  Removes property.

- virtual IEDLSysStringCollectionEnumPtr getPropertyCollectionEnum ()=0

  Retrieves a navigable list of the property names stored on this node.

- virtual bool hasChildNodes () const =0

  Function that indicates whether this node is a parent to other nodes.

- virtual IDOMNodePtr getParentNode () const =0

  Gets the parent node of this node.

- virtual IDOMNodePtr getFirstChild () const =0

  Gets the first child node of this node.

- virtual IDOMNodePtr getLastChild () const =0

  Gets the last child node of this node.

- virtual IDOMNodePtr getNextChild (const IDOMNodePtr &child) const =0

  Gets the child node which follows the node passed in.

- virtual IDOMNodePtr getPreviousChild (const IDOMNodePtr &child) const =0

  Gets the child node which precedes the node passed in.

- virtual IDOMNodePtr getPreviousSibling () const =0

  Retrieves the node's previous sibling node.

- virtual IDOMNodePtr getNextSibling () const =0

  Retrieves node's next sibling node.

- virtual void appendChild (const IDOMNodePtr &child)=0

  Appends a node to the end of the node's child list.

- virtual void insertChild (const IDOMNodePtr &ptrPreviousSibling, const IDOMNodePtr &child, bool bCheck↩ Complete=true)=0

  Insert a child node after ptrPreviousSibling.
- virtual IDOMNodePtr extractChild (const IDOMNodePtr &child)=0

  Extracts (that is, finds and removes) a child node from the node children. After extraction the child node is no longer a part of the DOM. If no node is specified, the first available node will be extracted from the node's children.
- virtual void replaceChild (const IDOMNodePtr &oldChild, const IDOMNodePtr &newChild)=0

  Replaces the child node with another.
- virtual bool isComplete () const =0

  Signals the completeness of the node.
  A complete node is one that has no more children to be added to it.
- virtual void **setComplete** ()=0

  Sets the node's completeness status to "true".
- virtual IDOMNodeFlags ∗ getFlags ()=0

  Retrieves the node's flags property.
- virtual void setParentNode (const IDOMNodePtr &ptrParent)=0

  Sets the parent node.
- virtual void setPreviousSibling (const IDOMNodePtr &ptrPreviousSibling)=0

  Sets the previous sibling node.
- virtual void setNextSibling (const IDOMNodePtr &ptrNextSibling)=0

  Sets the next sibling node.
- virtual bool isAncestor (const IDOMNodePtr &ptrCandidate)=0

  Function tests whether a candidate node is a descendant of the node.
- virtual FRect getBounds (bool applyTransform=true, bool applyClip=true)

  Find the conservative bounding box of the marking content of the node.
- virtual bool copyNodeData (IDOMNode ∗pSourceNode)=0

  Copy the properties collection, the flags and the DOM ID from the given source node to this one.
- virtual IDOMNodePtr cloneNode (IEDLClassFactory ∗pFactory) const =0

  Simpified node cloning. An exception of type IEDLError will be thrown on failure.
- virtual IDOMNodePtr cloneTree (IEDLClassFactory ∗pFactory) const =0

  Clone the tree of nodes beginning at this node. An exception of type IEDLError will be thrown on failure.
- virtual void cloneTreeAndAppend (IEDLClassFactory ∗pFactory, const IDOMNodePtr &dest) const =0

  Clone the tree of nodes beginning at this node, and append the result to the destination tree.
- virtual void **completeTree** ()=0

  Mark the entire tree from this point as complete. You should not ordinarily need to call this function.
- virtual void **removeCompleteFlagFromTree** ()=0

  Mark the entire tree from this point as complete.
- virtual void findChildrenOfType (eDOMNodeType type, CDOMNodeVect &nodes, bool searchForms=false)=0

  Find all children of this node with the given type, appending to the given vector. Does not descend into brushes.
- virtual void walkTree (WalkTreeFunc func, void ∗priv, bool descendIntoBrushes=false, bool descendInto↩ Forms=false)=0

  Walk through the DOM calling a given function on each node. The function is allowed to:
- virtual void notifyOnDestruct (NodeDeleteFunc func, void ∗priv)=0

  Register interest in being told when this node is about to be destroyed.
- virtual void unregisterNotify (NodeDeleteFunc func, void ∗priv)=0

  Unregister interest in being told when this node is about to be destroyed.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves the class id of IDOMJobTk.

Static Public Member Functions inherited from IDOMNode

- static EDL_API FMatrix effectiveTransformationOfNode (const IDOMNodePtr &node)

  Attempt to find the effective transformation matrix external to the specified node relative to either a containing page or ultimate parent.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.229.1 Detailed Description

Represents a Job Ticket Node.

7.229.2 Member Function Documentation

classID()

```
static const CClassID & IDOMJobTkNode::classID ( )  [inline], [static]
```

Retrieves the class id of IDOMJobTk.

Returns

CClassID. Class id of the element

findChild() [1/2]

```
virtual IDOMJobTkNodePtr IDOMJobTkNode::findChild (
            eDOMJobTkNodeType nodeType,
            const EDLQName & qname )  [pure virtual]
```

Searches through the node's direct child set for a node matching the provided node type and qname.

Parameters

nodeType	Job ticket node type.
qname	Qualified name of JobTicket node.

Returns

IDOMJobTkNodePtr. Smart pointer to the found JobTicket node

findChild() [2/2]

```
virtual IDOMJobTkNodePtr IDOMJobTkNode::findChild (
            eDOMJobTkNodeType nodeType,
            const EDLString & name,
            const EDLSysString & nmspace )  [pure virtual]
```

Searches through the node's direct child set for a node matching the provided node type, name and namespace.

Parameters

nodeType	Job ticket node type.
name	Name of JobTicket node.
nmspace	Namespace of JobTicket node.

Returns

IDOMJobTkNodePtr. Smart pointer to the found JobTicket node

getChildValue()

```
virtual IDOMJobTkValuePtr IDOMJobTkNode::getChildValue ( )  [pure virtual]
```

If JobTicket node has child with type eDOMJobTkPTNodeValue then returns this child, otherwise returns NULL.

Returns

IDOMJobTkValuePtr. Smart pointer to the JobTicket value node interface

getJobTkContent()

```
virtual IDOMJobTkContentPtr IDOMJobTkNode::getJobTkContent ( )  [pure virtual]
```

Returns the JobTicket content of this node, by following up the parent node chain.

Returns

IDOMJobTkContentPtr. Smart pointer to the JobTicket content interface

getJobTkNodeType()

```
virtual eDOMJobTkNodeType IDOMJobTkNode::getJobTkNodeType ( )  [pure virtual]
```

Retrieves the node type.

Returns

eDOMJobTkNodeType. Returns the node type.

getQName()

```
virtual EDLQName IDOMJobTkNode::getQName ( ) const  [pure virtual]
```

Retrieves the node qname.

Returns

EDLQName Qualified name value

getQNameAsString()

```
virtual EDLString IDOMJobTkNode::getQNameAsString ( ) const  [pure virtual]
```

Retrieves the full qualified name of the node as a string.

Returns

EDLString The node's qualified name as a string.

setJobTkNodeType()

```
virtual void IDOMJobTkNode::setJobTkNodeType (
            eDOMJobTkNodeType nodeType )  [pure virtual]
```

Sets the node type.

Parameters

nodeType	The node type. See eDOMJobTkNodeType.

setQName()

```
virtual void IDOMJobTkNode::setQName (
            const EDLQName & qname )  [pure virtual]
```

Sets the qualified name of the node.

URI references can contain characters not allowed in names, and are often inconveniently long, so expanded names are not used directly to name elements and attributes in XML documents. Instead qualified names are used. A qualified name is a name subject to namespace interpretation.

Parameters

qname	Qualified name value

The documentation for this class was generated from the following file:

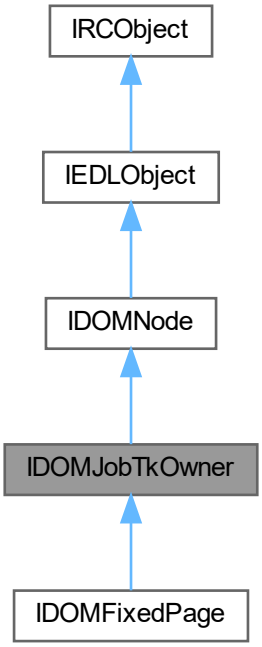
- idomjobtk.h

7.230 IDOMJobTkOwner Class Reference

Interface to the IDOMJobTkOwner node.

```
#include <idomjobtk.h>
```

Inheritance diagram for IDOMJobTkOwner:



Public Member Functions

- virtual IDOMJobTkPtr getJobTicket () const =0

  Retrieves the JobTicket held by this node.
- virtual void setJobTicket (const IDOMJobTkPtr &ptrJobTicket)=0

  Sets the JobTicket for the node.

Public Member Functions inherited from IDOMNode

- virtual ~**IDOMNode** ()

  virtual destructor
- virtual DOMid getDOMid () const =0

  Retrieves the node ID.
- virtual void setDOMid (DOMid id)=0

  Sets the node ID.
- virtual eDOMNodeType getNodeType () const =0

  Retrieves the DOM node type.
- virtual bool getProperty (const EDLSysString &propertyName, PValue &propertyValue) const =0

  Retrieves the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.
- virtual void setProperty (const EDLSysString &propertyName, const PValue &propertyValue)=0

Sets the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.

- virtual void removeProperty (const EDLSysString &propertyName)=0

  Removes property.
- virtual IEDLSysStringCollectionEnumPtr getPropertyCollectionEnum ()=0

  Retrieves a navigable list of the property names stored on this node.
- virtual bool hasChildNodes () const =0

  Function that indicates whether this node is a parent to other nodes.
- virtual IDOMNodePtr getParentNode () const =0

  Gets the parent node of this node.
- virtual IDOMNodePtr getFirstChild () const =0

  Gets the first child node of this node.
- virtual IDOMNodePtr getLastChild () const =0

  Gets the last child node of this node.
- virtual IDOMNodePtr getNextChild (const IDOMNodePtr &child) const =0

  Gets the child node which follows the node passed in.
- virtual IDOMNodePtr getPreviousChild (const IDOMNodePtr &child) const =0

  Gets the child node which precedes the node passed in.
- virtual IDOMNodePtr getPreviousSibling () const =0

  Retrieves the node's previous sibling node.
- virtual IDOMNodePtr getNextSibling () const =0

  Retrieves node's next sibling node.
- virtual void appendChild (const IDOMNodePtr &child)=0

  Appends a node to the end of the node's child list.
- virtual void insertChild (const IDOMNodePtr &ptrPreviousSibling, const IDOMNodePtr &child, bool bCheck↩ Complete=true)=0

  Insert a child node after ptrPreviousSibling.
- virtual IDOMNodePtr extractChild (const IDOMNodePtr &child)=0

  Extracts (that is, finds and removes) a child node from the node children. After extraction the child node is no longer a part of the DOM. If no node is specified, the first available node will be extracted from the node's children.
- virtual void replaceChild (const IDOMNodePtr &oldChild, const IDOMNodePtr &newChild)=0

  Replaces the child node with another.
- virtual bool isComplete () const =0

  Signals the completeness of the node.
  A complete node is one that has no more children to be added to it.
- virtual void **setComplete** ()=0

  Sets the node's completeness status to "true".
- virtual IDOMNodeFlags * getFlags ()=0

  Retrieves the node's flags property.
- virtual void setParentNode (const IDOMNodePtr &ptrParent)=0

  Sets the parent node.
- virtual void setPreviousSibling (const IDOMNodePtr &ptrPreviousSibling)=0

  Sets the previous sibling node.
- virtual void setNextSibling (const IDOMNodePtr &ptrNextSibling)=0

  Sets the next sibling node.
- virtual bool isAncestor (const IDOMNodePtr &ptrCandidate)=0

  Function tests whether a candidate node is a descendant of the node.
- virtual FRect getBounds (bool applyTransform=true, bool applyClip=true)

  Find the conservative bounding box of the marking content of the node.
- virtual bool copyNodeData (IDOMNode *pSourceNode)=0

  Copy the properties collection, the flags and the DOM ID from the given source node to this one.

- virtual IDOMNodePtr cloneNode (IEDLClassFactory *pFactory) const =0

  Simpified node cloning. An exception of type IEDLError will be thrown on failure.
- virtual IDOMNodePtr cloneTree (IEDLClassFactory *pFactory) const =0

  Clone the tree of nodes beginning at this node. An exception of type IEDLError will be thrown on failure.
- virtual void cloneTreeAndAppend (IEDLClassFactory *pFactory, const IDOMNodePtr &dest) const =0

  Clone the tree of nodes beginning at this node, and append the result to the destination tree.
- virtual void **completeTree** ()=0

  Mark the entire tree from this point as complete. You should not ordinarily need to call this function.
- virtual void **removeCompleteFlagFromTree** ()=0

  Mark the entire tree from this point as complete.
- virtual void findChildrenOfType (eDOMNodeType type, CDOMNodeVect &nodes, bool searchForms=false)=0

  Find all children of this node with the given type, appending to the given vector. Does not descend into brushes.
- virtual void walkTree (WalkTreeFunc func, void *priv, bool descendIntoBrushes=false, bool descendInto↩ Forms=false)=0

  Walk through the DOM calling a given function on each node. The function is allowed to:
- virtual void notifyOnDestruct (NodeDeleteFunc func, void *priv)=0

  Register interest in being told when this node is about to be destroyed.
- virtual void unregisterNotify (NodeDeleteFunc func, void *priv)=0

  Unregister interest in being told when this node is about to be destroyed.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves the class id of IDOMJobTkOwner.

Static Public Member Functions inherited from IDOMNode

- static EDL_API FMatrix effectiveTransformationOfNode (const IDOMNodePtr &node)

  Attempt to find the effective transformation matrix external to the specified node relative to either a containing page or ultimate parent.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.230.1 Detailed Description

Interface to the IDOMJobTkOwner node.

IDOMJobTkOwner is the immediate ancestor of the node types which can own a JobTicket. These are the IDOMContentRoot, IDOMDocumentSequence, IDOMFixedDocument and IDOMFixedPage classes.

7.230.2 Member Function Documentation

classID()

```
static const CClassID & IDOMJobTkOwner::classID ( )  [inline], [static]
```

Retrieves the class id of IDOMJobTkOwner.

Returns

CClassID. Class id of the element.

getJobTicket()

```
virtual IDOMJobTkPtr IDOMJobTkOwner::getJobTicket ( ) const  [pure virtual]
```

Retrieves the JobTicket held by this node.

Returns

IDOMJobTkPtr The JobTicket or NULL if there is no job ticket.

setJobTicket()

```
virtual void IDOMJobTkOwner::setJobTicket (
            const IDOMJobTkPtr & ptrJobTicket )  [pure virtual]
```

Sets the JobTicket for the node.

Parameters

ptrJobTicket	Smart pointer to the new JobTicket.

The documentation for this class was generated from the following file:

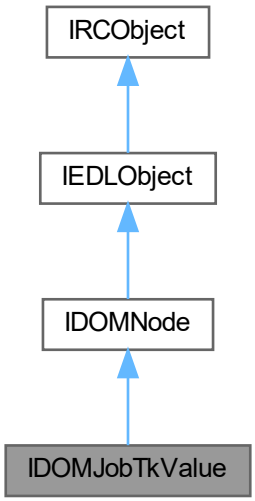
- idomjobtk.h

7.231 IDOMJobTkValue Class Reference

Represents a Job Ticket value element.

`#include <idomjobtk.h>`

Inheritance diagram for IDOMJobTkValue:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual bool isValid () const =0

  Returns an indicator of the validity of the data.
- virtual void setValue (const PValue &value)=0

  Sets the node value.
- virtual PValue getValue () const =0

  Retrieves the node value.

Public Member Functions inherited from IDOMNode

- virtual ~**IDOMNode** ()

  virtual destructor
- virtual DOMid getDOMid () const =0

  Retrieves the node ID.
- virtual void setDOMid (DOMid id)=0

  Sets the node ID.
- virtual eDOMNodeType getNodeType () const =0

  Retrieves the DOM node type.
- virtual bool getProperty (const EDLSysString &propertyName, PValue &propertyValue) const =0

  Retrieves the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.
- virtual void setProperty (const EDLSysString &propertyName, const PValue &propertyValue)=0

  Sets the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.
- virtual void removeProperty (const EDLSysString &propertyName)=0

  Removes property.
- virtual IEDLSysStringCollectionEnumPtr getPropertyCollectionEnum ()=0

  Retrieves a navigable list of the property names stored on this node.
- virtual bool hasChildNodes () const =0

  Function that indicates whether this node is a parent to other nodes.
- virtual IDOMNodePtr getParentNode () const =0

  Gets the parent node of this node.
- virtual IDOMNodePtr getFirstChild () const =0

  Gets the first child node of this node.
- virtual IDOMNodePtr getLastChild () const =0

  Gets the last child node of this node.
- virtual IDOMNodePtr getNextChild (const IDOMNodePtr &child) const =0

  Gets the child node which follows the node passed in.
- virtual IDOMNodePtr getPreviousChild (const IDOMNodePtr &child) const =0

  Gets the child node which precedes the node passed in.
- virtual IDOMNodePtr getPreviousSibling () const =0

  Retrieves the node's previous sibling node.
- virtual IDOMNodePtr getNextSibling () const =0

  Retrieves node's next sibling node.
- virtual void appendChild (const IDOMNodePtr &child)=0

  Appends a node to the end of the node's child list.
- virtual void insertChild (const IDOMNodePtr &ptrPreviousSibling, const IDOMNodePtr &child, bool bCheck↩ Complete=true)=0

  Insert a child node after ptrPreviousSibling.
- virtual IDOMNodePtr extractChild (const IDOMNodePtr &child)=0

  Extracts (that is, finds and removes) a child node from the node children. After extraction the child node is no longer a part of the DOM. If no node is specified, the first available node will be extracted from the node's children.
- virtual void replaceChild (const IDOMNodePtr &oldChild, const IDOMNodePtr &newChild)=0

  Replaces the child node with another.
- virtual bool isComplete () const =0

  Signals the completeness of the node.
  A complete node is one that has no more children to be added to it.
- virtual void **setComplete** ()=0

Sets the node's completeness status to "true".

- virtual IDOMNodeFlags * getFlags ()=0

  Retrieves the node's flags property.
- virtual void setParentNode (const IDOMNodePtr &ptrParent)=0

  Sets the parent node.
- virtual void setPreviousSibling (const IDOMNodePtr &ptrPreviousSibling)=0

  Sets the previous sibling node.
- virtual void setNextSibling (const IDOMNodePtr &ptrNextSibling)=0

  Sets the next sibling node.
- virtual bool isAncestor (const IDOMNodePtr &ptrCandidate)=0

  Function tests whether a candidate node is a descendant of the node.
- virtual FRect getBounds (bool applyTransform=true, bool applyClip=true)

  Find the conservative bounding box of the marking content of the node.
- virtual bool copyNodeData (IDOMNode *pSourceNode)=0

  Copy the properties collection, the flags and the DOM ID from the given source node to this one.
- virtual IDOMNodePtr cloneNode (IEDLClassFactory *pFactory) const =0

  Simpified node cloning. An exception of type IEDLError will be thrown on failure.
- virtual IDOMNodePtr cloneTree (IEDLClassFactory *pFactory) const =0

  Clone the tree of nodes beginning at this node. An exception of type IEDLError will be thrown on failure.
- virtual void cloneTreeAndAppend (IEDLClassFactory *pFactory, const IDOMNodePtr &dest) const =0

  Clone the tree of nodes beginning at this node, and append the result to the destination tree.
- virtual void **completeTree** ()=0

  Mark the entire tree from this point as complete. You should not ordinarily need to call this function.
- virtual void **removeCompleteFlagFromTree** ()=0

  Mark the entire tree from this point as complete.
- virtual void findChildrenOfType (eDOMNodeType type, CDOMNodeVect &nodes, bool searchForms=false)=0

  Find all children of this node with the given type, appending to the given vector. Does not descend into brushes.
- virtual void walkTree (WalkTreeFunc func, void *priv, bool descendIntoBrushes=false, bool descendInto↩ Forms=false)=0

  Walk through the DOM calling a given function on each node. The function is allowed to:
- virtual void notifyOnDestruct (NodeDeleteFunc func, void *priv)=0

  Register interest in being told when this node is about to be destroyed.
- virtual void unregisterNotify (NodeDeleteFunc func, void *priv)=0

  Unregister interest in being told when this node is about to be destroyed.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves the class id of IDOMJobTk.

Static Public Member Functions inherited from IDOMNode

- static EDL_API FMatrix effectiveTransformationOfNode (const IDOMNodePtr &node)

  Attempt to find the effective transformation matrix external to the specified node relative to either a containing page or ultimate parent.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.231.1 Detailed Description

Represents a Job Ticket value element.

7.231.2 Member Function Documentation

classID()

```
static const CClassID & IDOMJobTkValue::classID ( )  [inline], [static]
```

Retrieves the class id of IDOMJobTk.

Returns

CClassID. The class id of the element.

getValue()

```
virtual PValue IDOMJobTkValue::getValue ( ) const  [pure virtual]
```

Retrieves the node value.

Returns

PValue The node value

isValid()

```
virtual bool IDOMJobTkValue::isValid ( ) const  [pure virtual]
```

Returns an indicator of the validity of the data.

Returns

bool. Returns true if the data has been initialised, false if it has not.

setValue()

```
virtual void IDOMJobTkValue::setValue (
            const PValue & value )  [pure virtual]
```

Sets the node value.

Parameters

value	The new node value.

The documentation for this class was generated from the following file:

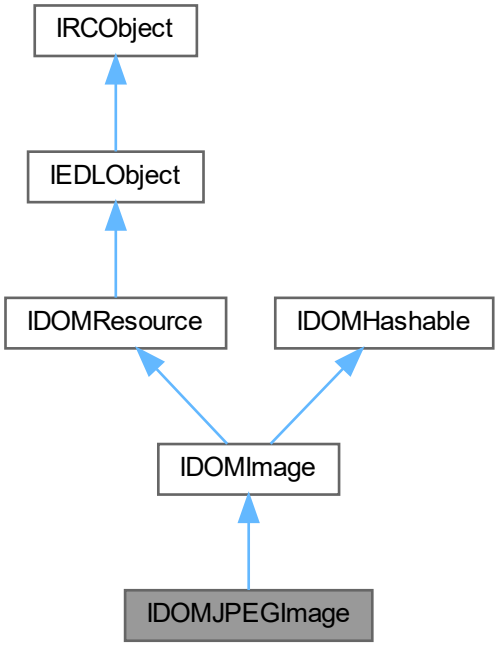
- idomjobtk.h

7.232 IDOMJPEGImage Class Reference

Interface to a class representing a JPEG (.jpg or .jpeg) image.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMJPEGImage:



Static Public Member Functions

- static EDL_API IDOMJPEGImagePtr create (IEDLClassFactory *pFactory, const IInputStreamPtr &stream)

  Create a JPEG Image resource with the given JPEG stream.
- static EDL_API IDOMImagePtr createWriterAndImage (const ISessionPtr &session, IImageFrameWriterPtr &frame, const IDOMColorSpacePtr &colorSpace, uint32 width, uint32 height, uint8 bitsPerComponent=8, double xResolution=96.0, double yResolution=96.0, uint8 quality=3, const IInputStreamPtr &inStream=IInputStreamPtr(), const IOutputStreamPtr &outStream=IOutputStreamPtr())

Create an IDOMJPEGImage and frame that can be used to populate same.

- static EDL_API void encode (const ISessionPtr &pSession, const IDOMImagePtr &image, const IOutput↩ StreamPtr &stream, uint8 quality=3, bool allowCmyk=false, bool invertCmyk=true)

  Encode an image as a JPEG stream. This routine may convert the source image into a form that may be encoded as JPEG, such as by stripping alpha channels or converting to a supported colour space.
- static EDL_API void encode (const ISessionPtr &pSession, const IImageFramePtr &frame, const IOutput↩ StreamPtr &stream, uint8 quality=3, bool allowCmyk=false, bool invertCmyk=true)

  Encode the contents of an IImageFrame as a JPEG stream, returning the stream. This routine may convert the source image into a form that may be encoded as JPEG, such as by stripping alpha channels or converting to a supported colour space.
- static const CClassID & classID ()

  Retrieves class id of IDOMJPEGImage.

Additional Inherited Members

Public Member Functions inherited from IDOMImage

- virtual IImageDecoderPtr createImageDecoder (IEDLClassFactory ∗factory, const IDOMImagePropertiesPtr &imageProperties)=0

  Creates a properly initialized image decoder object that reads from an inputstream that is specific to that image format.
- virtual IImageFramePtr getImageFrame (IEDLClassFactory ∗factory)

  Fetch the image frame; convenience.
- virtual IImageEncoderPtr createImageEncoder (const ISessionPtr &session, const IOutputStreamPtr &imageDest, const IDOMImagePropertiesPtr &imageProperties)=0

  Creates a properly initialized image encoder object that writes to an outputstream that is specific to that image format.
- virtual IDOMImagePropertiesPtr getImageProperties ()=0

  Returns an object that stores the properties for this image object. The properties can then be inspected (or more added) by clients that need to manipulate the image resource.
- virtual eDOMImageType getImageType ()=0

  Retrieves the image type.
- virtual bool getIsRendered ()=0

  Determine if the image is as a result of rendering. This is indicated if the image type is eDITRendered or if the image explicitly notes this is the case (such as for IDOMPDFImage).
- virtual IDOMImagePtr getImageWithSubstitutedColorSpace (IEDLClassFactory ∗factory, const IDOMColor↩ SpacePtr &colorSpace)

  Obtain an image that is the same as this image, but with the colorspace substituted for another.

Public Member Functions inherited from IDOMResource

- virtual IInputStreamPtr getStream () const =0

  Retrieves the resource stream.
- virtual void setStream (const IInputStreamPtr &stream)=0

  Sets the resource stream for the node.
- virtual uint64 getStreamLength () const =0

  Retrieves the stream length, if it is available.
- virtual const EDLSysString & getUri () const =0

  Retrieves the resource URI.
- virtual void setUri (const EDLSysString &uri)=0

  Sets the resource URI.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.232.1 Detailed Description

Interface to a class representing a JPEG (.jpg or .jpeg) image.

7.232.2 Member Function Documentation

classID()

```
static const CClassID & IDOMJPEGImage::classID ( )  [inline], [static]
```

Retrieves class id of IDOMJPEGImage.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMJPEGImagePtr IDOMJPEGImage::create (
            IEDLClassFactory * pFactory,
            const IInputStreamPtr & stream )  [static]
```

Create a JPEG Image resource with the given JPEG stream.

Parameters

pFactory	The EDL Class factory to use.
stream	The stream containing the JPEG image.

Returns

IDOMImagePtr The new image.

createWriterAndImage()

```
static EDL_API IDOMImagePtr IDOMJPEGImage::createWriterAndImage (
            const ISessionPtr & session,
            IImageFrameWriterPtr & frame,
            const IDOMColorSpacePtr & colorSpace,
            uint32 width,
            uint32 height,
            uint8 bitsPerComponent = 8,
            double xResolution = 96.0,
            double yResolution = 96.0,
            uint8 quality = 3,
            const IInputStreamPtr & inStream = IInputStreamPtr(),
            const IOutputStreamPtr & outStream = IOutputStreamPtr() )  [static]
```

Create an IDOMJPEGImage and frame that can be used to populate same.

Parameters

session	The session to use
frame	On exit, this is populated with a frame ready to receive image data via frame->writeScanLine(). Use frame->flushData() to complete the encoding process.
colorSpace	The color space to use. Must be compatible with the JPEG format.
width	The width of the image, in pixels.
height	The height of the image, in pixels.
bitsPerComponent	The number of bits per color component, either 8 or 16. Default is 8.
xResolution	The x resolution, in pixels-per-inch.
yResolution	The y resolution, in pixels-per-inch.
quality	The desired quality in the range 1 through 5, with 1 being lowest and 5 being highest. Default is 3.
inStream	Optional. The first in a pair of streams used to read and write the raw image data if an external stream is desired. If NULL, a temporary store stream will be created. If non NULL, outStream must also be provided.
outStream	Optional. The second in a pair of streams used to read and write the raw image data if an external stream is desired. If NULL, a temporary store stream will be created. If non NULL, inStream must also be provided.

Returns

IDOMImagePtr The resulting image. Not valid until the frame is flushed.

encode() [1/2]

```
static EDL_API void IDOMJPEGImage::encode (
            const ISessionPtr & pSession,
            const IDOMImagePtr & image,
            const IOutputStreamPtr & stream,
            uint8 quality = 3,
            bool allowCmyk = false,
            bool invertCmyk = true )  [static]
```

Encode an image as a JPEG stream. This routine may convert the source image into a form that may be encoded as JPEG, such as by stripping alpha channels or converting to a supported colour space.

Parameters

pSession	The relevant EDL session
image	The image to be encoded
stream	The stream to use to store the image data.
quality	The desired quality in the range 1 through 5, with 1 being lowest and 5 being highest. Default is 3.
allowCmyk	Whether or not CMYK encoding should be allowed. CMYK JPEGs are not universally supported, and so are not generated by default.
invertCmyk	If CMYK data is encoded, this controls whether or not the image data is inverted. If inverted, it is likely that the CMYK image will view correctly in more viewers. However, doing this will result in inverted results when the image is used with Mako.

encode() [2/2]

```
static EDL_API void IDOMJPEGImage::encode (
            const ISessionPtr & pSession,
            const IImageFramePtr & frame,
            const IOutputStreamPtr & stream,
            uint8 quality = 3,
            bool allowCmyk = false,
            bool invertCmyk = true )  [static]
```

Encode the contents of an IImageFrame as a JPEG stream, returning the stream. This routine may convert the source image into a form that may be encoded as JPEG, such as by stripping alpha channels or converting to a supported colour space.

Parameters

pSession	The relevant EDL session
frame	The frame providing the source image data
stream	The stream to use to store the image data.
quality	The desired quality in the range 1 through 5, with 1 being lowest and 5 being highest. Default is 3.
allowCmyk	Whether or not CMYK encoding should be allowed. CMYK JPEGs are not universally supported, and so are not generated by default.
invertCmyk	If CMYK data is encoded, this controls whether or not the image data is inverted. If inverted, it is likely that the CMYK image will view correctly in more viewers. However, doing this will result in inverted results when the image is used with Mako.

The documentation for this class was generated from the following file:

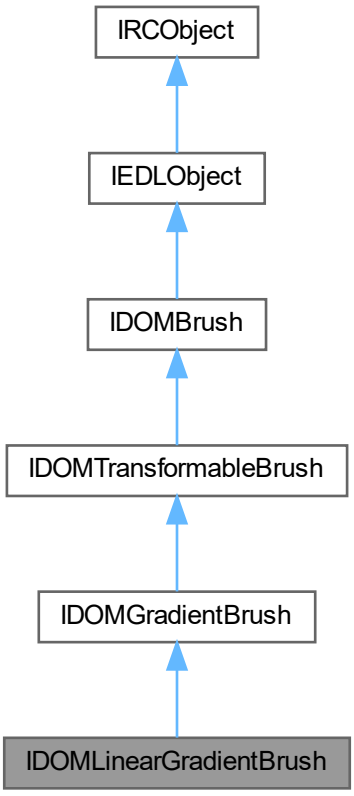
- idomimageresource.h

7.233 IDOMLinearGradientBrush Class Reference

IDOMLinearGradientBrush interface. A linear gradient brush is used to specify a gradient along a vector.

`#include <idombrush.h>`

Inheritance diagram for IDOMLinearGradientBrush:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual const FPoint & getStartPoint () const =0

  Retrieves the start point of the linear gradient.
- virtual void setStartPoint (const FPoint &pt)=0

  Sets the start point of the linear gradient.
- virtual const FPoint & getEndPoint () const =0

  Retrieves the end point of the linear gradient.
- virtual void setEndPoint (const FPoint &pt)=0

  Sets the end point of the linear gradient.
- virtual IDOMShadingPatternType2BrushPtr createShading (IEDLClassFactory *pFactory, bool ignore↩ SpreadMethod, bool useFirstStopColorSpace)=0

  Create a Type2 Shading Pattern brush from this linear brush. Repeat and reflect spread methods cannot be handled using this method (instead use createPdfBrush() which will emulate using a tiling pattern brush. However if ignore↩ SpreadMethod is set, a shading pattern resembling a pad reflect mode will be generated. Further, alpha information in the gradient stops are ignored.
- virtual IDOMBrushPtr createPdfBrush (IEDLClassFactory *pFactory, bool useFirstStopColorSpace=false)=0

  Create a PDF-compatible brush for this gradient. If the pad mode is reflect, the result will be a shading brush, otherwise a tiling pattern will be created to repeat and reflect the shade as required. As per createShading, the opacity in the gradient stops is ignored.

Public Member Functions inherited from IDOMGradientBrush

- virtual eColorInterpolationMode getColorInterpolationMode () const =0

  Retrieves the color interpolation mode value of the radial gradient brush.
- virtual void setColorInterpolationMode (eColorInterpolationMode cim)=0

  Sets the color interpolation mode value of the radial gradient brush.
- virtual eSpreadMethod getSpreadMethod () const =0

  Retrieves the spread method value of the RadialGradientBrush element.
- virtual void setSpreadMethod (eSpreadMethod sm)=0

  Sets spread method value of the RadialGradientBrush element.
- virtual void setGradientStops (const CDOMGradientStopVect &stops)=0

  Set the vector of stops in this gradient. Must not be empty.
- virtual const CDOMGradientStopVect & getGradientStops () const =0

  Retrieves the vector of stops in this gradient. An exception will be thrown if the gradient has no stops.
- virtual void addGradientStop (const IDOMGradientStopPtr &ptrGradientStop)=0

  Append a gradient stop.
- virtual void normalizeStops (IEDLClassFactory *pFactory, const IDOMColorSpacePtr &ptrColor↩ Space=IDOMColorSpacePtr(), eRenderingIntent intent=eRelativeColorimetric, eBlackPointCompensation bpc=eBPCDefault)=0

  Normalize the gradient stops to a single color space, sort them and ensure there are stops at 0.0 and 1.0.

Public Member Functions inherited from IDOMTransformableBrush

- virtual const FMatrix & getRenderTransform () const =0

  Retrieves the render transform matrix.
- virtual void setRenderTransform (const FMatrix &matrix)=0

  Sets the render transform matrix.

Public Member Functions inherited from IDOMBrush

- virtual eBrushType getBrushType () const =0

  Retrieves the type of the brush.
- virtual float getOpacity () const =0

  Retrieves the opacity value of the brush element.
- virtual void setOpacity (float opc)=0

  Sets the opacity value of a brush element.
- virtual IDOMBrushPtr getAdjustedForUseInTransformedNode (IEDLClassFactory *pFactory, const FMatrix &nodeTransform)

  Get a version of this brush adjusted for use inside a node with the given transform.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static EDL_API IDOMLinearGradientBrushPtr create (IEDLClassFactory *pFactory, const FPoint &start↩ Point, const FPoint &endPoint, const CDOMGradientStopVect &stops, float opacity=1.0f, const FMatrix &renderTransform=FMatrix(), eSpreadMethod spreadMethod=eNoSpread, eColorInterpolationMode color↩ InterpolationMode=eSRgbLinearInterpolation)

  Simplified linear gradient brush creation. Throws an IEDLError on failure.
- static const CClassID & classID ()

  Retrieves class id of IDOMLinearGradientBrush.

Additional Inherited Members

Public Types inherited from IDOMBrush

- enum eBrushType {
  eSolidColor , eLinearGradient , eRadialGradient , eImage ,
  eMasked , eVisual , eSoftMask , eTilingPattern ,
  eType1ShadingPattern , eType2ShadingPattern , eType3ShadingPattern , eType4567ShadingPattern ,
  eNull }

  Brush type enumeration.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.233.1 Detailed Description

IDOMLinearGradientBrush interface. A linear gradient brush is used to specify a gradient along a vector.

7.233.2 Member Function Documentation

classID()

```
static const CClassID & IDOMLinearGradientBrush::classID ( )  [inline], [static]
```

Retrieves class id of IDOMLinearGradientBrush.

Returns

CClassID The class ID of IDOMLinearGradientBrush.

create()

```
static EDL_API IDOMLinearGradientBrushPtr IDOMLinearGradientBrush::create (
            IEDLClassFactory * pFactory,
            const FPoint & startPoint,
            const FPoint & endPoint,
            const CDOMGradientStopVect & stops,
            float opacity = 1.0f,
            const FMatrix & renderTransform = FMatrix(),
            eSpreadMethod spreadMethod = eNoSpread,
            eColorInterpolationMode colorInterpolationMode = eSRgbLinearInterpolation )  [static]
```

Simplified linear gradient brush creation. Throws an IEDLError on failure.

Parameters

pFactory	The EDL Class factory to use.
startPoint	The start point of the gradient
endPoint	The end point of the gradient
stops	The gradient stops
opacity	The alpha to use.
renderTransform	The render transform to use
spreadMethod	The desired spread method
colorInterpolationMode	The desired color interpolation mode

Returns

IDOMLinearGradientBrushPtr The new brush.

createPdfBrush()

```
virtual IDOMBrushPtr IDOMLinearGradientBrush::createPdfBrush (
            IEDLClassFactory * pFactory,
            bool useFirstStopColorSpace = false )  [pure virtual]
```

Create a PDF-compatible brush for this gradient. If the pad mode is reflect, the result will be a shading brush, otherwise a tiling pattern will be created to repeat and reflect the shade as required. As per createShading, the opacity in the gradient stops is ignored.

Parameters

pFactory	The EDL class factory
useFirstStopColorSpace	If true the color space in the first stop will be used for the shading. Otherwise either sRGB or scRGB will be used depending on the interpolation mode.

Returns

IDOMPatternBrush The resulting brush.

createShading()

```
virtual IDOMShadingPatternType2BrushPtr IDOMLinearGradientBrush::createShading (
            IEDLClassFactory * pFactory,
            bool ignoreSpreadMethod,
            bool useFirstStopColorSpace )  [pure virtual]
```

Create a Type2 Shading Pattern brush from this linear brush. Repeat and reflect spread methods cannot be handled using this method (instead use createPdfBrush() which will emulate using a tiling pattern brush. However if ignore↩ SpreadMethod is set, a shading pattern resembling a pad reflect mode will be generated. Further, alpha information in the gradient stops are ignored.

Parameters

pFactory	The EDL class factory
useFirstStopColorSpace	If true the color space in the first stop will be used for the shading. Otherwise either sRGB or scRGB will be used depending on the interpolation mode.
ignoreSpreadMethod	If true the spread method will be ignored and assumed to be ePad. If this is set false and the gradient has eRepeat or eReflect spread mode set then a NULL brush will result.

Returns

IDOMShadingPatternType2BrushPtr The resulting brush, or NULL if it was not possible to create such a brush due to spread method.

getEndPoint()

```
virtual const FPoint & IDOMLinearGradientBrush::getEndPoint ( ) const  [pure virtual]
```

Retrieves the end point of the linear gradient.

Returns

FPoint The end point of the linear gradient.

getStartPoint()

```
virtual const FPoint & IDOMLinearGradientBrush::getStartPoint ( ) const  [pure virtual]
```

Retrieves the start point of the linear gradient.

Returns

FPoint Start point of the linear gradient

setEndPoint()

```
virtual void IDOMLinearGradientBrush::setEndPoint (
            const FPoint & pt )  [pure virtual]
```

Sets the end point of the linear gradient.

Parameters

pt	The new end point

setStartPoint()

```
virtual void IDOMLinearGradientBrush::setStartPoint (
            const FPoint & pt )  [pure virtual]
```

Sets the start point of the linear gradient.

Parameters

pt	The new start point.

The documentation for this class was generated from the following file:

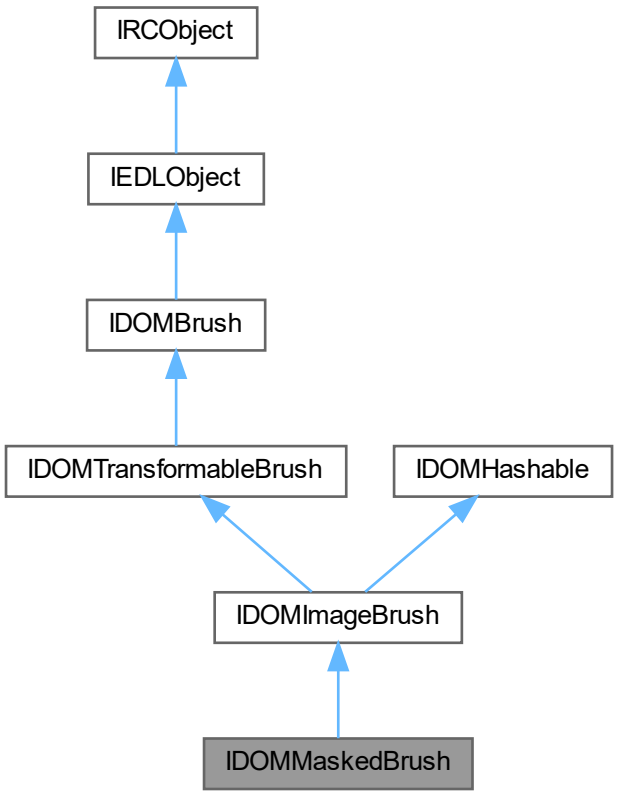
- idombrush.h

7.234 IDOMMaskedBrush Class Reference

IDOMMaskedBrush interface, this describes a generalization of a masked image. The sub-brush (set by getBrush()/setBrush()) is painted through a mask specified by the image. Importantly, the sub-brush is not subject to the IDOMImageBrush render transform. Tiling is not supported for this brush type.

`#include <idombrush.h>`

Inheritance diagram for IDOMMaskedBrush:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual IDOMBrushPtr getBrush () const =0

  Retrieves smart pointer to the brush to be painted through the image.
- virtual void setBrush (const IDOMBrushPtr &brush)=0

  Sets brush.

- virtual bool getIsSoftMask (IEDLClassFactory *pFactory)=0

  Is the mask a soft mask? That is, is the mask not a single bit image? An outright error will result in an exception of type IEDLError.
- virtual IDOMBrushPtr getEquivalentXPSBrush (IEDLClassFactory *pFactory, const FRect &enclosed↩ Bounds)=0

  Gets an equivalent image or visual brush that can be used to express this masked brush in XPS compatible form. This does not guarantee that the sub brush can be represented in XPS; these ought to be processed first. To do this, the bounds of the enclosing object in internal space must be provided (that is, the coordinate space effective inside that object).
- virtual IDOMImageBrushPtr getSimpleImageBrush (IEDLClassFactory *pFactory)=0

  Attempts to create a single IDOMImageBrush that representes the masked result. This can be achieved if the brush masked by the image is a solid color brush, or if the brush masked by the image is an image with the same dimensions. Returns NULL if such a brush isn't possible, and throws an IEDLError exception on outright failures.

Public Member Functions inherited from IDOMImageBrush

- virtual eTilingMode getTileMode () const =0

  Retrieves the tiling mode value of the image brush.
- virtual void setTileMode (eTilingMode tm)=0

  Sets the tiling mode of the image brush.
- virtual eViewUnits getViewBoxUnits () const =0

  Retrieves the viewbox units used by the image brush. Currently, only absolute units are supported.
- virtual void setViewBoxUnits (eViewUnits vu)=0

  Sets the viewbox units value of the image brush. Currently, only absolute units are supported.
- virtual eViewUnits getViewPortUnits () const =0

  Retrieves the viewport units value of the image brush. Currently, only absolute units are supported.
- virtual void setViewPortUnits (eViewUnits vu)=0

  Sets the viewport units used for the image brush. Currently, only absolute units are supported.
- virtual const FRect & getViewBox () const =0

  Retrieves the viewbox rectangle.
- virtual void setViewBox (const FRect &vb)=0

  Sets the viewbox rectangle.
- virtual const FRect & getViewPort () const =0

  Retrieves the viewport rectangle.
- virtual void setViewPort (const FRect &vp)=0

  Sets the viewport rectangle.
- virtual IDOMICCProfilePtr getICCProfile () const =0

  Retrieves the external ICC profile of the brush if present.
- virtual void setICCProfile (const IDOMICCProfilePtr &icc)=0

  Retrieves the external ICC profile of the brush if present.
- virtual IDOMImagePtr getImageSource () const =0

  Retrieves a smart pointer to the image resource.
- virtual void setImageSource (const IDOMImagePtr &ptrImageSource)=0

  Sets the image resource for the brush.
- virtual CDOMAlternateImageVect getAlternateImages () const =0

  Retrieves any alternate images associated with this brush.
- virtual void setAlternateImages (const CDOMAlternateImageVect &alternates)=0

  Set the alternate images associated with this brush.
- virtual JawsMako::IOptionalContentDetailsPtr getOptionalContentDetails () const =0

  Returns any optional content information that applies to this brush.
- virtual void setOptionalContentDetails (const JawsMako::IOptionalContentDetailsPtr &details)=0

Set the optional content details that apply to this brush, or NULL to remove.

- virtual JawsMako::IPDFDictionaryPtr getPdfPropertiesDictionary () const =0

  Get the dictionary containing PDF properties attached to the image object. This dictionary will be as per the XObject dictionary in the original PDF, but with entries handled by existing DOM features stripped. In particular, the following entries described in the PDF specification will not be present in the dictionary:
- virtual void setPdfPropertiesDictionary (const JawsMako::IPDFDictionaryPtr &propertiesDictionary)=0

  Set the dictionary containing PDF properties attached to the image object. Please see getPdfPropertiesDictionary() above for information on the form of this dictionary.
- virtual IDOMTilingPatternBrushPtr getEquivalentTilingBrush (IEDLClassFactory *pFactory)=0

  Gets an equivalent IDOMTilingPattern brush. If the receiver has a tile mode of eNoTile, this call will fail.
- virtual IDOMVisualBrushPtr getEquivalentVisualBrush (IEDLClassFactory *pFactory)=0

  For tiled images, returns an equivalent visual brush containing the image without tiling. Will throw an exception if the image is not tiled. Not available for masked brushes as tiling is not supported for those brushes.

Public Member Functions inherited from IDOMTransformableBrush

- virtual const FMatrix & getRenderTransform () const =0

  Retrieves the render transform matrix.
- virtual void setRenderTransform (const FMatrix &matrix)=0

  Sets the render transform matrix.

Public Member Functions inherited from IDOMBrush

- virtual eBrushType getBrushType () const =0

  Retrieves the type of the brush.
- virtual float getOpacity () const =0

  Retrieves the opacity value of the brush element.
- virtual void setOpacity (float opc)=0

  Sets the opacity value of a brush element.
- virtual IDOMBrushPtr getAdjustedForUseInTransformedNode (IEDLClassFactory *pFactory, const FMatrix &nodeTransform)

  Get a version of this brush adjusted for use inside a node with the given transform.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static EDL_API IDOMMaskedBrushPtr create (IEDLClassFactory *pFactory, const IDOMImage↩ Ptr &mask, const IDOMBrushPtr &brush, const FRect &viewBox, const FRect &viewPort, const FMatrix &renderTransform=FMatrix(), float opacity=1.0f, const CDOMAlternateImageVect &alternate↩ Images=CDOMAlternateImageVect(), const JawsMako::IOptionalContentDetailsPtr &optionalContent↩ Details=JawsMako::IOptionalContentDetailsPtr(), const JawsMako::IPDFDictionaryPtr &properties↩ Dictionary=JawsMako::IPDFDictionaryPtr())

  Simplified creator for a masked brush. Throws an IEDLError on failure.
- static const CClassID & classID ()

  Retrieves class id of IDOM.

Static Public Member Functions inherited from IDOMImageBrush

- static EDL_API IDOMImageBrushPtr create (IEDLClassFactory *pFactory, const IDOMImagePtr &image, const FRect &viewBox, const FRect &viewPort, const FMatrix &renderTransform=FMatrix(), float opacity=1.0f, eTilingMode tileMode=eNoTile, const CDOMAlternateImageVect &alternate↩ Images=CDOMAlternateImageVect(), const JawsMako::IOptionalContentDetailsPtr &optionalContent↩ Details=JawsMako::IOptionalContentDetailsPtr(), const JawsMako::IPDFDictionaryPtr &properties↩ Dictionary=JawsMako::IPDFDictionaryPtr())

  Simplified creator for an image brush. Throws an IEDLError on failure.
- static const CClassID & classID ()

  Retrieves class id of IDOM.

Additional Inherited Members

Public Types inherited from IDOMBrush

- enum eBrushType {
  eSolidColor , eLinearGradient , eRadialGradient , eImage ,
  eMasked , eVisual , eSoftMask , eTilingPattern ,
  eType1ShadingPattern , eType2ShadingPattern , eType3ShadingPattern , eType4567ShadingPattern ,
  eNull }

  Brush type enumeration.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.234.1 Detailed Description

IDOMMaskedBrush interface, this describes a generalization of a masked image. The sub-brush (set by getBrush()/setBrush()) is painted through a mask specified by the image. Importantly, the sub-brush is not subject to the IDOMImageBrush render transform. Tiling is not supported for this brush type.

7.234.2 Member Function Documentation

classID()

```
static const CClassID & IDOMMaskedBrush::classID ( )  [inline], [static]
```

Retrieves class id of IDOM.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMMaskedBrushPtr IDOMMaskedBrush::create (
            IEDLClassFactory * pFactory,
            const IDOMImagePtr & mask,
            const IDOMBrushPtr & brush,
            const FRect & viewBox,
            const FRect & viewPort,
            const FMatrix & renderTransform = FMatrix(),
            float opacity = 1.0f,
            const CDOMAlternateImageVect & alternateImages = CDOMAlternateImageVect(),
            const JawsMako::IOptionalContentDetailsPtr & optionalContentDetails = JawsMako:↩
:IOptionalContentDetailsPtr(),
            const JawsMako::IPDFDictionaryPtr & propertiesDictionary = JawsMako::IPDFDictionaryPtr()
)  [static]
```

Simplified creator for a masked brush. Throws an IEDLError on failure.

Parameters

pFactory	The factory to use.
mask	The image mask to use.
brush	The brush to use
viewBox	The desired view box. If empty, the viewPort will be set to the full area of the image.
viewPort	The desired view port.
renderTransform	The desired render transform.
opacity	The desired brush opacity.
alternateImages	A vector containing alternate images.
optionalContentDetails	Any optional content details to be applied to this masked case
propertiesDictionary	Dictionary containing any additional PDF properties

Returns

IDOMMaskedBrushPtr The new brush.

getBrush()

```
virtual IDOMBrushPtr IDOMMaskedBrush::getBrush ( ) const  [pure virtual]
```

Retrieves smart pointer to the brush to be painted through the image.

Returns

IDOMBrushPtr The brush

getEquivalentXPSBrush()

```
virtual IDOMBrushPtr IDOMMaskedBrush::getEquivalentXPSBrush (
            IEDLClassFactory * pFactory,
            const FRect & enclosedBounds )  [pure virtual]
```

Gets an equivalent image or visual brush that can be used to express this masked brush in XPS compatible form. This does not guarantee that the sub brush can be represented in XPS; these ought to be processed first. To do this, the bounds of the enclosing object in internal space must be provided (that is, the coordinate space effective inside that object).

Parameters

pFactory	A pointer to an EDL class factory
enclosedBounds	The bounds in internal object space of the object that has this brush.

Returns

IDOMBrushPtr The XPS compatible form of this brush. This brush will be cached, so do not edit; make a clone if the returned brush needs to be changed.

getIsSoftMask()

```
virtual bool IDOMMaskedBrush::getIsSoftMask (
            IEDLClassFactory * pFactory )  [pure virtual]
```

Is the mask a soft mask? That is, is the mask not a single bit image? An outright error will result in an exception of type IEDLError.

Parameters

pFactory	A pointer to an EDL class factory

Returns

bool True if the mask is soft.

getSimpleImageBrush()

```
virtual IDOMImageBrushPtr IDOMMaskedBrush::getSimpleImageBrush (
            IEDLClassFactory * pFactory )  [pure virtual]
```

Attempts to create a single IDOMImageBrush that representes the masked result. This can be achieved if the brush masked by the image is a solid color brush, or if the brush masked by the image is an image with the same dimensions. Returns NULL if such a brush isn't possible, and throws an IEDLError exception on outright failures.

Parameters

pFactory	pFactory A pointer to an EDL class factory

Returns

IDOMImageBrushPtr The image brush. This brush will be cached, so do not edit; make a clone if the returned brush needs to be changed.

setBrush()

```
virtual void IDOMMaskedBrush::setBrush (
            const IDOMBrushPtr & brush )  [pure virtual]
```

Sets brush.

Parameters

brush	Smart pointer to the brush resource to be painted through the image.

The documentation for this class was generated from the following file:

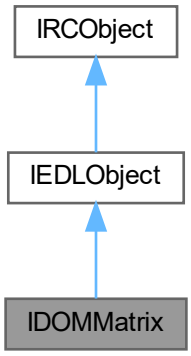
- idombrush.h

7.235 IDOMMatrix Class Reference

Defines the render transform matrix.

```
#include <idomresources.h>
```

Inheritance diagram for IDOMMatrix:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual const FMatrix & getRenderTransform () const =0

  Retrieves the render transform matrix.
- virtual void setRenderTransform (const FMatrix &matrix)=0

  Sets the render transform matrix.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves the class id of IDOMMatrix.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.235.1 Detailed Description

Defines the render transform matrix.

7.235.2 Member Function Documentation

classID()

```
static const CClassID & IDOMMatrix::classID ( )  [inline], [static]
```

Retrieves the class id of IDOMMatrix.

Returns

CClassID. Returns the class id of the element.

getRenderTransform()

```
virtual const FMatrix & IDOMMatrix::getRenderTransform ( ) const  [pure virtual]
```

Retrieves the render transform matrix.

Returns

FMatrix Returns the render transform matrix

setRenderTransform()

```
virtual void IDOMMatrix::setRenderTransform (
            const FMatrix & matrix )  [pure virtual]
```

Sets the render transform matrix.

Parameters

matrix	Render transform matrix

The documentation for this class was generated from the following file:

- idomresources.h

7.236 IDOMMatteRemoverFilter Class Reference

An image filter that removes a Matte and undoes premultiplication for a PDF Matte'd image and soft mask. The resulting image does not have alpha, and can be used with the mask to generate the desired result.

`#include <idomimageresource.h>`

7.236.1 Detailed Description

An image filter that removes a Matte and undoes premultiplication for a PDF Matte'd image and soft mask. The resulting image does not have alpha, and can be used with the mask to generate the desired result.

The documentation for this class was generated from the following file:

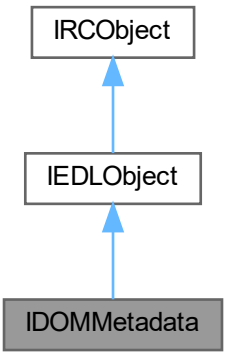
- idomimageresource.h

7.237 IDOMMetadata Class Reference

The IDOMMetadata interface provides access to the metadata attached to the DocumentSequence node. The IDOMMetadata interface is designed to be flexible enough to represent different types of metadata.

`#include <idommetadata.h>`

Inheritance diagram for IDOMMetadata:



Public Types

- enum eType {
  eCoreProperties = 0 , eDocumentInfo , eViewerPreferences , ePageView ,
  ePDFInfo , eMetadataTypeCnt }

  Metadata types data type.
- enum eXmpContainerType { eXmpContainer_None = 0 , eXmpContainer_Alt = 1 , eXmpContainer_Bag = 2 , eXmpContainer_Seq = 3 }

  The variant values passed to PValue when storing a CEDLStringVect.

Public Member Functions

- virtual bool getProperty (eType propertyType, const EDLSysString &propertyName, PValue &propertyValue) const =0

  Retrieves the value of a named property.
- virtual bool setProperty (eType propertyType, const EDLSysString &propertyName, const PValue &propertyValue)=0

  Sets value of a property.
- virtual bool removeProperty (eType propertyType, const EDLSysString &propertyName)=0

  Removes a property, if it exists.
- virtual IEDLSysStringCollectionEnumPtr getPropertyCollectionEnum (eType propertyType)=0

  Retrieves a navigable list of the property names in the metadata.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static EDL_API IDOMMetadataPtr **create** (IEDLClassFactory *pFactory)

  Create a new IDOMMetadata object.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.237.1 Detailed Description

The IDOMMetadata interface provides access to the metadata attached to the DocumentSequence node. The IDOMMetadata interface is designed to be flexible enough to represent different types of metadata.

The metadata set on IDOMDocumentSequence was designed to hold general information about the document, such as document author or creation/modification date.
Metadata can also describe the sort of information that can be found in the document information dictionary for a PDF document. The closest analogue for the XPS world would be the information contained in the CoreProperties part.

Several groups of metadata properties are supported by EDL, as described by the eType enumeration.
The properties in each group are as follows:

eCoreProperties - Core Properties

- **category** - EDLSysString
- **contentStatus** - EDLSysString
- **contentType** - EDLSysString
- **keywords** - EDLSysString
- **lastModifiedBy** - EDLSysString
- **lastPrinted** - IEDLTime
- **revision** - EDLSysString
- **version** - EDLSysString
- **creator** - EDLSysString
- **description** - EDLSysString
- **identifier** - EDLSysString
- **language** - EDLSysString
- **subject** - EDLSysString
- **title** - EDLSysString
- **created** - IEDLTime
- **modified** - IEDLTime

eViewerPreferences - Viewer Preferences

- **HideToolbar** - bool
- **HideMenubar** - bool
- **HideWindowUI** - bool
- **FitWindow** - bool
- **CenterWindow** - bool
- **DisplayDocTitle** - bool
- **PickTrayByPDFSize** - bool

- **NumCopies** - int32
- **NonFullScreenPageMode** - EDLSysString
- **Direction** - EDLSysString
- **ViewArea** - EDLSysString
- **ViewClip** - EDLSysString
- **PrintArea** - EDLSysString
- **PrintClip** - EDLSysString
- **PrintScaling** - EDLSysString
- **Duplex** - EDLSysString

eDocumentInfo - Document Information

- **Title** - EDLSysString
- **Author** - EDLSysString or CEDLStringVect with variant value (one of eXmpContainerType)
- **Subject** - EDLSysString or CEDLStringVect with variant value (one of eXmpContainerType)
- **Keywords** - EDLSysString or CEDLStringVect with variant value (one of eXmpContainerType)
- **Creator** - EDLSysString
- **Producer** - EDLSysString
- **CreationDate** - IEDLTime
- **ModDate** - IEDLTime
- **Trapped** - EDLSysString

ePDFInfo - Read-only PDF document properties

- **Version** - EDLSysString property in a string form **M.m**, where M is the major and m is the minor version number
- **ExtensionLevel** - int32 representing the extension level
- **Linearized** - bool indicator of original PDF document linearization
- **FileIdentifier1** - EDLSysString property representing the first part of PDF file ID
- **FileIdentifier2** - EDLSysString property representing second part of PDF file ID
- **Standard** - EDLSysString property representing the PDF/A, PDF/E, PDF/X or PDF/UA standard the PDF purports to conform to
- **Marked** - bool true if the PDF document claims to be a tagged PDF
- **UserProperties** - bool true if the PDF document is tagged and structure elements that contain user properties are present
- **Suspects** - bool true if the PDF contains tagged suspects

ePageView - Document-specific view controls

- **PageMode** - EDLSysString. For example, set this to "UseOutlines" to initially display the Bookmarks panel
- **PageLayout** - EDLSysString

7.237.2 Member Function Documentation

getProperty()

```
virtual bool IDOMMetadata::getProperty (
            eType propertyType,
            const EDLSysString & propertyName,
            PValue & propertyValue ) const  [pure virtual]
```

Retrieves the value of a named property.

Parameters

propertyType	The type of the property whose value is to be retrieved.
propertyName	The name of the property whose value is to be retrieved.
propertyValue	A smart pointer to receive the retrieved property value.

Returns

bool. Returns true on success, false if the call fails.

getPropertyCollectionEnum()

```
virtual IEDLSysStringCollectionEnumPtr IDOMMetadata::getPropertyCollectionEnum (
            eType propertyType )  [pure virtual]
```

Retrieves a navigable list of the property names in the metadata.

Returns

IEDLSysStringCollectionEnumPtr. Returns a smart pointer to the list of property names.

removeProperty()

```
virtual bool IDOMMetadata::removeProperty (
            eType propertyType,
            const EDLSysString & propertyName )  [pure virtual]
```

Removes a property, if it exists.

Parameters

propertyType	The type of the property to be removed.
propertyName	The name of the property to be removed.

Returns

bool. Returns true on success, false if the call fails.

setProperty()

```
virtual bool IDOMMetadata::setProperty (
            eType propertyType,
            const EDLSysString & propertyName,
            const PValue & propertyValue )  [pure virtual]
```

Sets value of a property.

Parameters

propertyType	The type of the property whose value is to be set.
propertyName	The name of the property whose value is to be set.
propertyValue	The new value to be set for the property.

Returns

bool. Returns true on success, false if the call fails.

The documentation for this class was generated from the following file:

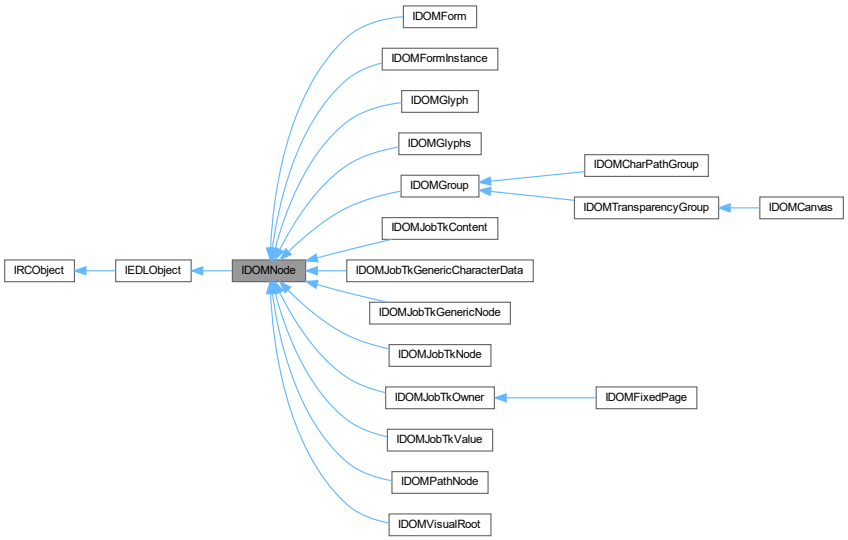
- idommetadata.h

7.238 IDOMNode Class Reference

Abstract class providing the interface to basic DOM node functionality. IDOMNode is the base class for many of the other DOM node types, and defines many of the basic functions of DOM nodes. Exceptions of type IEDLError are thrown on outright failures.

`#include <idomnode.h>`

Inheritance diagram for IDOMNode:



Public Member Functions

- virtual ~**IDOMNode** ()

  virtual destructor
- virtual DOMid getDOMid () const =0

  Retrieves the node ID.
- virtual void setDOMid (DOMid id)=0

  Sets the node ID.
- virtual eDOMNodeType getNodeType () const =0

  Retrieves the DOM node type.
- virtual bool getProperty (const EDLSysString &propertyName, PValue &propertyValue) const =0

  Retrieves the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.
- virtual void setProperty (const EDLSysString &propertyName, const PValue &propertyValue)=0

  Sets the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.
- virtual void removeProperty (const EDLSysString &propertyName)=0

  Removes property.
- virtual IEDLSysStringCollectionEnumPtr getPropertyCollectionEnum ()=0

  Retrieves a navigable list of the property names stored on this node.
- virtual bool hasChildNodes () const =0

  Function that indicates whether this node is a parent to other nodes.
- virtual IDOMNodePtr getParentNode () const =0

  Gets the parent node of this node.
- virtual IDOMNodePtr getFirstChild () const =0

  Gets the first child node of this node.
- virtual IDOMNodePtr getLastChild () const =0

  Gets the last child node of this node.
- virtual IDOMNodePtr getNextChild (const IDOMNodePtr &child) const =0

  Gets the child node which follows the node passed in.
- virtual IDOMNodePtr getPreviousChild (const IDOMNodePtr &child) const =0

  Gets the child node which precedes the node passed in.
- virtual IDOMNodePtr getPreviousSibling () const =0

  Retrieves the node's previous sibling node.
- virtual IDOMNodePtr getNextSibling () const =0

  Retrieves node's next sibling node.
- virtual void appendChild (const IDOMNodePtr &child)=0

  Appends a node to the end of the node's child list.
- virtual void insertChild (const IDOMNodePtr &ptrPreviousSibling, const IDOMNodePtr &child, bool bCheck↩ Complete=true)=0

  Insert a child node after ptrPreviousSibling.
- virtual IDOMNodePtr extractChild (const IDOMNodePtr &child)=0

  Extracts (that is, finds and removes) a child node from the node children. After extraction the child node is no longer a part of the DOM. If no node is specified, the first available node will be extracted from the node's children.
- virtual void replaceChild (const IDOMNodePtr &oldChild, const IDOMNodePtr &newChild)=0

  Replaces the child node with another.
- virtual bool isComplete () const =0

  Signals the completeness of the node.
  A complete node is one that has no more children to be added to it.
- virtual void **setComplete** ()=0

Sets the node's completeness status to "true".

- virtual IDOMNodeFlags * getFlags ()=0

  Retrieves the node's flags property.
- virtual void setParentNode (const IDOMNodePtr &ptrParent)=0

  Sets the parent node.
- virtual void setPreviousSibling (const IDOMNodePtr &ptrPreviousSibling)=0

  Sets the previous sibling node.
- virtual void setNextSibling (const IDOMNodePtr &ptrNextSibling)=0

  Sets the next sibling node.
- virtual bool isAncestor (const IDOMNodePtr &ptrCandidate)=0

  Function tests whether a candidate node is a descendant of the node.
- virtual FRect getBounds (bool applyTransform=true, bool applyClip=true)

  Find the conservative bounding box of the marking content of the node.
- virtual bool copyNodeData (IDOMNode *pSourceNode)=0

  Copy the properties collection, the flags and the DOM ID from the given source node to this one.
- virtual IDOMNodePtr cloneNode (IEDLClassFactory *pFactory) const =0

  Simpified node cloning. An exception of type IEDLError will be thrown on failure.
- virtual IDOMNodePtr cloneTree (IEDLClassFactory *pFactory) const =0

  Clone the tree of nodes beginning at this node. An exception of type IEDLError will be thrown on failure.
- virtual void cloneTreeAndAppend (IEDLClassFactory *pFactory, const IDOMNodePtr &dest) const =0

  Clone the tree of nodes beginning at this node, and append the result to the destination tree.
- virtual void **completeTree** ()=0

  Mark the entire tree from this point as complete. You should not ordinarily need to call this function.
- virtual void **removeCompleteFlagFromTree** ()=0

  Mark the entire tree from this point as complete.
- virtual void findChildrenOfType (eDOMNodeType type, CDOMNodeVect &nodes, bool searchForms=false)=0

  Find all children of this node with the given type, appending to the given vector. Does not descend into brushes.
- virtual void walkTree (WalkTreeFunc func, void *priv, bool descendIntoBrushes=false, bool descendInto↩ Forms=false)=0

  Walk through the DOM calling a given function on each node. The function is allowed to:
- virtual void notifyOnDestruct (NodeDeleteFunc func, void *priv)=0

  Register interest in being told when this node is about to be destroyed.
- virtual void unregisterNotify (NodeDeleteFunc func, void *priv)=0

  Unregister interest in being told when this node is about to be destroyed.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static EDL_API FMatrix effectiveTransformationOfNode (const IDOMNodePtr &node)

  Attempt to find the effective transformation matrix external to the specified node relative to either a containing page or ultimate parent.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.238.1 Detailed Description

Abstract class providing the interface to basic DOM node functionality. IDOMNode is the base class for many of the other DOM node types, and defines many of the basic functions of DOM nodes. Exceptions of type IEDLError are thrown on outright failures.

7.238.2 Member Function Documentation

appendChild()

```
virtual void IDOMNode::appendChild (
            const IDOMNodePtr & child )  [pure virtual]
```

Appends a node to the end of the node's child list.

Parameters

child	The child node to append.

cloneNode()

```
virtual IDOMNodePtr IDOMNode::cloneNode (
            IEDLClassFactory * pFactory ) const  [pure virtual]
```

Simpified node cloning. An exception of type IEDLError will be thrown on failure.

Parameters

pFactory	The EDL class factory.

Returns

IDOMNodePtr The cloned node.

cloneTree()

```
virtual IDOMNodePtr IDOMNode::cloneTree (
            IEDLClassFactory * pFactory ) const  [pure virtual]
```

Clone the tree of nodes beginning at this node. An exception of type IEDLError will be thrown on failure.

Parameters

pFactory	The EDL class factory.

Returns

IDOMNodePtr The cloned node.

cloneTreeAndAppend()

```
virtual void IDOMNode::cloneTreeAndAppend (
            IEDLClassFactory * pFactory,
            const IDOMNodePtr & dest ) const  [pure virtual]
```

Clone the tree of nodes beginning at this node, and append the result to the destination tree.

Parameters

pFactory	The EDL class factory.
dest	The destination node

copyNodeData()

```
virtual bool IDOMNode::copyNodeData (
            IDOMNode * pSourceNode )  [pure virtual]
```

Copy the properties collection, the flags and the DOM ID from the given source node to this one.

Parameters

pSourceNode	Smart pointer to the source node.

Returns

bool True on success, false if the call fails.

effectiveTransformationOfNode()

```
static EDL_API FMatrix IDOMNode::effectiveTransformationOfNode (
            const IDOMNodePtr & node )  [static]
```

Attempt to find the effective transformation matrix external to the specified node relative to either a containing page or ultimate parent.

Parameters

node	The specified node

Returns

FMatrix The effective transformation matrix

extractChild()

```
virtual IDOMNodePtr IDOMNode::extractChild (
            const IDOMNodePtr & child )  [pure virtual]
```

Extracts (that is, finds and removes) a child node from the node children. After extraction the child node is no longer a part of the DOM. If no node is specified, the first available node will be extracted from the node's children.

Parameters

child	A pointer to the child node to extract. If set to NULL, the first available child node will be extracted.

Returns

IDOMNodePtr A smart pointer to the extracted child node, or NULL if child was NULL and there are no children.

findChildrenOfType()

```
virtual void IDOMNode::findChildrenOfType (
            eDOMNodeType type,
            CDOMNodeVect & nodes,
            bool searchForms = false )  [pure virtual]
```

Find all children of this node with the given type, appending to the given vector. Does not descend into brushes.

Parameters

type	the type of node to find
nodes	Vector to receive the found nodes.
searchForms	If true, search inside forms referenced from form instance nodes.

getBounds()

```
virtual FRect IDOMNode::getBounds (
            bool applyTransform = true,
            bool applyClip = true )  [inline], [virtual]
```

Find the conservative bounding box of the marking content of the node.

```
For composite objects such as canvases
and pages, this function will recurse through its current children.
```

Parameters

applyTransform	Controls whether or not the receiver's RenderTransform is applied to the bounds (if it has one) • Pass true (default) to return results in the coordinate space of the object enclosing the receiver • Pass false to return results in the coordinate space active inside the object
applyClip	Controls whether or not the receiver's Clip is applied to the bounds (if it has a clip). This parameter only applies to the current node clip path, it does not apply to child nodes. It is generally used to test if the current nodes clip path is effective.

Returns

FRect The computed bounding box.

getDOMid()

```
virtual DOMid IDOMNode::getDOMid ( ) const  [pure virtual]
```

Retrieves the node ID.

Returns

DOMid The numeric value that uniquely identifies the node.

getFirstChild()

```
virtual IDOMNodePtr IDOMNode::getFirstChild ( ) const  [pure virtual]
```

Gets the first child node of this node.

Returns

IDOMNodePtr A smart pointer to the first child node.

getFlags()

```
virtual IDOMNodeFlags * IDOMNode::getFlags ( )  [pure virtual]
```

Retrieves the node's flags property.

Returns

IDOMNodeFlags A pointer to the node's flags property.

getLastChild()

```
virtual IDOMNodePtr IDOMNode::getLastChild ( ) const  [pure virtual]
```

Gets the last child node of this node.

Returns

IDOMNodePtr A smart pointer to the last child node.

getNextChild()

```
virtual IDOMNodePtr IDOMNode::getNextChild (
            const IDOMNodePtr & child ) const  [pure virtual]
```

Gets the child node which follows the node passed in.

Parameters

child	The "current" child node; the node whose next sibling is required.

Returns

IDOMNodePtr A smart pointer to the next child node. If the child node passed in was the last one under the node, returns a smart pointer to the node's first child.

getNextSibling()

```
virtual IDOMNodePtr IDOMNode::getNextSibling ( ) const  [pure virtual]
```

Retrieves node's next sibling node.

Returns

IDOMNodePtr A smart pointer to the node's next sibling.

getNodeType()

```
virtual eDOMNodeType IDOMNode::getNodeType ( ) const  [pure virtual]
```

Retrieves the DOM node type.

Returns

eDOMNodeType The DOM node type.

getParentNode()

```
virtual IDOMNodePtr IDOMNode::getParentNode ( ) const  [pure virtual]
```

Gets the parent node of this node.

Returns

IDOMNodePtr A smart pointer to the parent node.

getPreviousChild()

```
virtual IDOMNodePtr IDOMNode::getPreviousChild (
            const IDOMNodePtr & child ) const  [pure virtual]
```

Gets the child node which precedes the node passed in.

Parameters

child	The "current" child node; the node whose previous sibling is required.

Returns

IDOMNodePtr A smart pointer to the previous child node. If the child node passed in was the last one under the node, returns a smart pointer to the node's last child.

getPreviousSibling()

```
virtual IDOMNodePtr IDOMNode::getPreviousSibling ( ) const  [pure virtual]
```

Retrieves the node's previous sibling node.

Returns

IDOMNodePtr A smart pointer to the node's previous sibling.

getProperty()

```
virtual bool IDOMNode::getProperty (
            const EDLSysString & propertyName,
            PValue & propertyValue ) const  [pure virtual]
```

Retrieves the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.

Parameters

propertyName	The name of the property.
propertyValue	Smart pointer to receive the value of the property.

Returns

bool True on success, false if the property does not exist.

getPropertyCollectionEnum()

```
virtual IEDLSysStringCollectionEnumPtr IDOMNode::getPropertyCollectionEnum ( )  [pure virtual]
```

Retrieves a navigable list of the property names stored on this node.

Returns

IEDLSysStringCollectionEnumPtr A smart pointer to the property names list or NULL if there are no properties.

hasChildNodes()

```
virtual bool IDOMNode::hasChildNodes ( ) const  [pure virtual]
```

Function that indicates whether this node is a parent to other nodes.

Returns

bool Returns true if the node has child nodes, false if it does not.

insertChild()

```
virtual void IDOMNode::insertChild (
            const IDOMNodePtr & ptrPreviousSibling,
            const IDOMNodePtr & child,
            bool bCheckComplete = true )  [pure virtual]
```

Insert a child node after ptrPreviousSibling.

Parameters

ptrPreviousSibling	Smart pointer to the previous sibling node. If ptrPreviousSibling is NULL then child will be inserted as the first node
child	The child node to insert.
bCheckComplete	If false insert node even if parent is complete.

isAncestor()

```
virtual bool IDOMNode::isAncestor (
            const IDOMNodePtr & ptrCandidate )  [pure virtual]
```

Function tests whether a candidate node is a descendant of the node.

Parameters

ptrCandidate	Smart pointer to the candidate node.

Returns

bool True if ptrCandidate is a descendant of this node.

isComplete()

```
virtual bool IDOMNode::isComplete ( ) const  [pure virtual]
```

Signals the completeness of the node.
A complete node is one that has no more children to be added to it.

Returns

bool True if the node is complete, false if the node is incomplete.

notifyOnDestruct()

```
virtual void IDOMNode::notifyOnDestruct (
            NodeDeleteFunc func,
            void * priv )  [pure virtual]
```

Register interest in being told when this node is about to be destroyed.

Parameters

func	The function to call when this node is about to be destroyed
priv	A private pointer to be passed to the notification structure.

removeProperty()

```
virtual void IDOMNode::removeProperty (
            const EDLSysString & propertyName )  [pure virtual]
```

Removes property.

The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.

Parameters

propertyName	The name of the property.

replaceChild()

```
virtual void IDOMNode::replaceChild (
            const IDOMNodePtr & oldChild,
            const IDOMNodePtr & newChild )  [pure virtual]
```

Replaces the child node with another.

Parameters

oldChild	Pointer to the child node to be replaced.
newChild	Pointer to the replacing node.

setDOMid()

```
virtual void IDOMNode::setDOMid (
            DOMid id )  [pure virtual]
```

Sets the node ID.

Parameters

id	The new DOM ID for the node.

setNextSibling()

```
virtual void IDOMNode::setNextSibling (
            const IDOMNodePtr & ptrNextSibling )  [pure virtual]
```

Sets the next sibling node.

Parameters

ptrNextSibling	Smart pointer to the new next sibling node.

setParentNode()

```
virtual void IDOMNode::setParentNode (
            const IDOMNodePtr & ptrParent )  [pure virtual]
```

Sets the parent node.

Parameters

ptrParent	Smart pointer to the new parent node.

setPreviousSibling()

```
virtual void IDOMNode::setPreviousSibling (
            const IDOMNodePtr & ptrPreviousSibling )  [pure virtual]
```

Sets the previous sibling node.

Parameters

ptrPreviousSibling	Smart pointer to the new previous sibling node.

setProperty()

```
virtual void IDOMNode::setProperty (
            const EDLSysString & propertyName,
            const PValue & propertyValue )  [pure virtual]
```

Sets the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.

Parameters

propertyName	The name of the property.
propertyValue	The new value for the property.

unregisterNotify()

```
virtual void IDOMNode::unregisterNotify (
            NodeDeleteFunc func,
            void * priv )  [pure virtual]
```

Unregister interest in being told when this node is about to be destroyed.

Parameters

func	The function that was registered to be called when the node is destroyed.
priv	The private pointer that was passed to notifyOnDestruct() along with func.

walkTree()

```
virtual void IDOMNode::walkTree (
```

```
WalkTreeFunc func,
void * priv,
bool descendIntoBrushes = false,
bool descendIntoForms = false )  [pure virtual]
```

Walk through the DOM calling a given function on each node. The function is allowed to:

- edit or inspect the node in any way
- replace the node (via parent->replaceNode())
- remove the node (via parent->extractChild())
- edit or modify the node's children in any way However the result is undefined if the nodes siblings are removed or reordered. Keep in mind that if a node within a brush or form is altered, all uses of that brush or form will see the change.

Parameters

func	The function to be called for each node. Return false from this function if you wish to stop the walking. It is also safe to throw exceptions from this function.
priv	A bare pointer that is passed to func for each call.
descendIntoBrushes	If true, the walker will descend into composite brushes, only once per invocation.
descendIntoForms	If true, the walker will descend into forms, only once per invocation.

The documentation for this class was generated from the following file:

- idomnode.h

7.239 IDOMNodeFlags Class Reference

A collection of bit flags used to signal various conditions of the node. For example, the **eNodeRenderFlag** flag identifies nodes that require rendering.

`#include <idomnode.h>`

Public Types

- enum DOMNodeFlags {
  eNodeRenderFlag = 8 , eNodeUnpackFlag = 9 , eNodeDirtyFlag = 10 , eNodeInterestingFlag = 11 ,
  eNodeSelectedFlag = 15 , eNodeUserFlag = 16 }

  DOM node flag enumeration.

Public Member Functions

- virtual ~**IDOMNodeFlags** ()

  Virtual destructor.
- virtual bool get (uint32 position) const =0

  Tests whether the bit at a specified position is set to 1.
- virtual void set (uint32 position, bool value=true)=0

  Sets the bit at the specified position to the specified value.

7.239.1 Detailed Description

A collection of bit flags used to signal various conditions of the node. For example, the **eNodeRenderFlag** flag identifies nodes that require rendering.

The IDOMNodeFlags class keeps primary, secondary and tertiary flags. The first eight bits (0..7) are reserved for EDL private use - OEM user cannot change these bits. The next eight bits (8..15) are reserved for EDL public use - an OEM user can change these bits and the bit's meaning is documented. The next sixteen bits (16..32) are reserved for OEM usage.

7.239.2 Member Enumeration Documentation

DOMNodeFlags

```
enum IDOMNodeFlags::DOMNodeFlags
```

DOM node flag enumeration.

- The first eight bits (0..7) are reserved for EDL private use - OEM user cannot change these bits.
- The next eight bits (8..15) are reserved for EDL public use - OEM user can change these bits, and their meaning is documented. For example, bit 8 is the **eNodeRenderFlag**.
- The next sixteen bits (16..32) are reserved for OEM usage

Enumerator

eNodeRenderFlag	Marks a node as not to be rendered
eNodeUnpackFlag	Marks a node as unpacked
eNodeDirtyFlag	Marks a node as dirty (i.e with changed content)
eNodeInterestingFlag	Marks a node as interesting for rendering purposes See **IJawsRenderer** for details.
eNodeSelectedFlag	Marks a node as selected
eNodeUserFlag	Marks a node with a user flag whose meaning is determined elsewhere

7.239.3 Member Function Documentation

get()

```
virtual bool IDOMNodeFlags::get (
            uint32 position ) const  [pure virtual]
```

Tests whether the bit at a specified position is set to 1.

Parameters

position	The bit position to test.

Returns

bool Returns true if the flag bit is set to 1, false if it is set to zero.

set()

```
virtual void IDOMNodeFlags::set (
            uint32 position,
            bool value = true )  [pure virtual]
```

Sets the bit at the specified position to the specified value.

Parameters

position	The bit position to set.
value	The bit value to set.

The documentation for this class was generated from the following file:

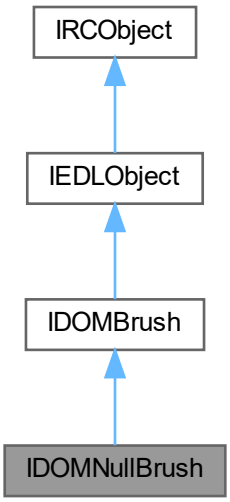
- idomnode.h

7.240 IDOMNullBrush Class Reference

IDOMNullBrush provides a way of representing the default marking brush in a Type3 postscript glyph definition or a tiling pattern with paintType 2. This is more of a placeholder that gets replaced when the Type3 glyph or paintType 2 tiling pattern is actually invoked.

```
#include <idombrush.h>
```

Inheritance diagram for IDOMNullBrush:



Classes

- class Data

  Initialization data.

Static Public Member Functions

- static EDL_API IDOMNullBrushPtr create (IEDLClassFactory *pFactory)

  Simplified creator for a Null brush. Throws an IEDLError on failure.
- static const CClassID & classID ()

  Retrieves class id of IDOMNullBrush.

Additional Inherited Members

Public Types inherited from IDOMBrush

- enum eBrushType {
  eSolidColor , eLinearGradient , eRadialGradient , eImage ,
  eMasked , eVisual , eSoftMask , eTilingPattern ,
  eType1ShadingPattern , eType2ShadingPattern , eType3ShadingPattern , eType4567ShadingPattern ,
  eNull }

  Brush type enumeration.

Public Member Functions inherited from IDOMBrush

- virtual eBrushType getBrushType () const =0

  Retrieves the type of the brush.
- virtual float getOpacity () const =0

  Retrieves the opacity value of the brush element.
- virtual void setOpacity (float opc)=0

  Sets the opacity value of a brush element.
- virtual IDOMBrushPtr getAdjustedForUseInTransformedNode (IEDLClassFactory *pFactory, const FMatrix &nodeTransform)

  Get a version of this brush adjusted for use inside a node with the given transform.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.240.1 Detailed Description

IDOMNullBrush provides a way of representing the default marking brush in a Type3 postscript glyph definition or a tiling pattern with paintType 2. This is more of a placeholder that gets replaced when the Type3 glyph or paintType 2 tiling pattern is actually invoked.

7.240.2 Member Function Documentation

classID()

```
static const CClassID & IDOMNullBrush::classID ( )  [inline], [static]
```

Retrieves class id of IDOMNullBrush.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMNullBrushPtr IDOMNullBrush::create (
            IEDLClassFactory * pFactory )  [static]
```

Simplified creator for a Null brush. Throws an IEDLError on failure.

Parameters

pFactory	The factory to use.

Returns

IDOMNullBrushPtr The new brush.

The documentation for this class was generated from the following file:

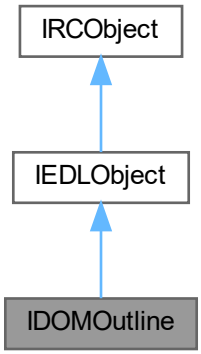
- idombrush.h

7.241 IDOMOutline Class Reference

Represents the outline of the document, which is the collection of bookmarks for the document.

`#include <idomoutline.h>`

Inheritance diagram for IDOMOutline:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual bool getLanguage (EDLString &strLanguage) const =0

  Retrieves the default language of the outline node.
- virtual bool setLanguage (const EDLString &strLanguage)=0

  the default language of the outline node.
- virtual IDOMOutlineTreePtr getOutlineTree ()=0

  Retrieves the outline tree.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static EDL_API IDOMOutlinePtr create (IEDLClassFactory *pFactory, const EDLString &language=EDLString(L"und"))

  Simplified creator to create an outline.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.241.1 Detailed Description

Represents the outline of the document, which is the collection of bookmarks for the document.

See also

IDOMOutlineEntry

IDOMOutline is similar to the table of contents in a book. Each entry into the outline is represented by an IDOMOutlineEntry object. IDOMOutlineEntry inherits from the IDOMNode and these outline entries are organized into a tree-like structure. IDOMOutline holds the reference to the root node of that tree.

7.241.2 Member Function Documentation

create()

```
static EDL_API IDOMOutlinePtr IDOMOutline::create (
            IEDLClassFactory * pFactory,
            const EDLString & language = EDLString(L"und") )  [static]
```

Simplified creator to create an outline.

Parameters

pFactory	The EDL factory to use
language	The language code to use (see setLanguage below)

getLanguage()

```
virtual bool IDOMOutline::getLanguage (
            EDLString & strLanguage ) const  [pure virtual]
```

Retrieves the default language of the outline node.

English is defined as en_GB and American English as en_US. There is no default setting. If the language is not known it is set to "und" (undetermined). For further information see http://www.w3.org/↩International/articles/language-tags/. The language is specified according to RFC 3066.

Parameters

strLanguage	Reference parameter to receive the default language.

Returns

bool Returns true on success, false if the call fails.

getOutlineTree()

```
virtual IDOMOutlineTreePtr IDOMOutline::getOutlineTree ( )  [pure virtual]
```

Retrieves the outline tree.

Returns

IDOMOutlineTreePtr. Returns a smart pointer to the outline tree.

setLanguage()

```
virtual bool IDOMOutline::setLanguage (
            const EDLString & strLanguage )  [pure virtual]
```

the default language of the outline node.

English is defined as en_GB and American English as en_US. There is no default setting. If the language is not known it is set to "und" (undetermined). For further information see http://www.w3.org/↩International/articles/language-tags/. The language is specified according to RFC 3066.

Parameters

strLanguage	The new default language

Returns

bool Returns true on success, false if the call fails.

The documentation for this class was generated from the following file:

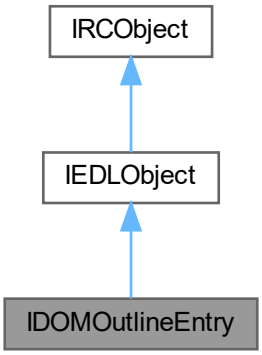
- idomoutline.h

7.242 IDOMOutlineEntry Class Reference

Represents an index to a specific location in the document or a specific location external to the document.

`#include <idomoutline.h>`

Inheritance diagram for IDOMOutlineEntry:



Classes

- class Data

  Initialization data.

Public Types

- enum eTextStyle { eTextStyleNone = 0x00 , eTextStyleItalic = 0x01 , eTextStyleBold = 0x02 , eTextStyleBoldItalic = eTextStyleItalic | eTextStyleBold }

  Specifies an outline text style.

Public Member Functions

- virtual bool getLanguage (EDLString &strLanguage) const =0

  Retrieves the default language of the outline entry node.
- virtual bool setLanguage (const EDLString &strLanguage)=0

  Retrieves the default language of the outline entry node.
- virtual bool getDescription (EDLString &strDescription) const =0

  Retrieves the description of the outline entry node.
- virtual bool setDescription (const EDLString &strDescription)=0

  Sets the description of the outline entry node.
- virtual bool getTarget (IDOMTargetPtr &ptrTarget) const =0

  Retrieves the target of the outline entry node. If outline entry doesn't have target then ptrTarget will be set to NULL.
- virtual bool setTarget (const IDOMTargetPtr &ptrTarget)=0

  Sets the target of the outline entry node. NULL is a valid value of ptrTarget parameter.

- virtual bool getTextColor (IDOMColorPtr &ptrColor) const =0

  Retrieves the color to be used for the outline entry's text.
- virtual bool setTextColor (const IDOMColorPtr &ptrColor)=0

  Sets the color to be used for the outline entry's text.
- virtual eTextStyle getTextStyle () const =0

  Retrieves the style to be used for the outline entry's text.
- virtual void setTextStyle (eTextStyle style)=0

  Sets the outline entry's text style.
- virtual bool getExpanded () const =0

  Retrieves "expanded" flag value.
- virtual void setExpanded (bool expanded)=0

  Sets the outline entry's "expanded" flag.
- virtual bool getStructureElement (IEDLObjectPtr &ptrSE) const =0

  Retrieves the structure element.
- virtual bool setStructureElement (const IEDLObjectPtr &ptrSE)=0

  Sets the structure element.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static EDL_API IDOMOutlineTreeNodePtr createNode (IEDLClassFactory *pFactory, const EDLString &description, bool expanded=true, const IDOMTargetPtr &target=IDOMTargetPtr(), const IDOMColorPtr &textColor=IDOMColorPtr(), eTextStyle style=eTextStyleNone, EDLString language=EDLString(L"und"))

  Simplified creator to create an outline tree entry with a new outline tree node.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.242.1 Detailed Description

Represents an index to a specific location in the document or a specific location external to the document.

See also

IDOMOutline

IDOMOutlineEntry objects are used for the individual components of the IDOMOutline. You can use the document outline information to support interactive functionality.

Every outline entry associates a text description of the bookmark with certain location within the document (represented as an IDOMInternalTarget or an IDOMPageTarget) or with an external location (represented as an IDOMExternalTarget). IDOMOutlineEntry inherits from IDOMNode and the outline entries are organized into a tree-like structure. There is no limit set on number of outline entries, so there can be as many as the available memory allows.

Using this information it is possible to create a navigation pane that uses the Unicode value of the description attribute of the outline entry node. The corresponding location is specified by the Target attribute.

7.242.2 Member Function Documentation

createNode()

```
static EDL_API IDOMOutlineTreeNodePtr IDOMOutlineEntry::createNode (
            IEDLClassFactory * pFactory,
            const EDLString & description,
            bool expanded = true,
            const IDOMTargetPtr & target = IDOMTargetPtr(),
            const IDOMColorPtr & textColor = IDOMColorPtr(),
            eTextStyle style = eTextStyleNone,
            EDLString language = EDLString(L"und") )  [static]
```

Simplified creator to create an outline tree entry with a new outline tree node.

Parameters

pFactory	The EDL factory to use
description	The description of the outline entry
expanded	If the outline entry's children should be shown by default
target	The destination for the outline entry, or NULL for no target.
textColor	The color for the outline entry, or NULL for default. If provided, it must use the DeviceRGB color space.
style	The desired text style
language	The language code to use (see setLanguage below)

getDescription()

```
virtual bool IDOMOutlineEntry::getDescription (
            EDLString & strDescription ) const  [pure virtual]
```

Retrieves the description of the outline entry node.

Parameters

strDescription	Reference parameter to receive the outline entry node descipriton.

Returns

bool Returns true on success, false if the call fails.

getExpanded()

```
virtual bool IDOMOutlineEntry::getExpanded ( ) const  [pure virtual]
```

Retrieves "expanded" flag value.

Returns

bool Returns true if the outline entry is expanded, false otherwise.

getLanguage()

```
virtual bool IDOMOutlineEntry::getLanguage (
            EDLString & strLanguage ) const  [pure virtual]
```

Retrieves the default language of the outline entry node.

English is defined as en_GB and American English as en_US. There is no default setting. If the language is not known it is set to "und" (undetermined). For further information see http://www.w3.org/↩International/articles/language-tags/. The language is specified according to RFC 3066.

Parameters

strLanguage	Default language

Returns

bool Returns true on success, false if the call fails.

getStructureElement()

```
virtual bool IDOMOutlineEntry::getStructureElement (
            IEDLObjectPtr & ptrSE ) const  [pure virtual]
```

Retrieves the structure element.

Parameters

ptrSE	Smart pointer to receive the structure element

Returns

bool Returns true on success, false if the call fails.

getTarget()

```
virtual bool IDOMOutlineEntry::getTarget (
            IDOMTargetPtr & ptrTarget ) const  [pure virtual]
```

Retrieves the target of the outline entry node. If outline entry doesn't have target then ptrTarget will be set to NULL.

Parameters

ptrTarget	Smart pointer to receive the target of the outline entry node.

Returns

bool Returns true on success, false if the call fails.

getTextColor()

```
virtual bool IDOMOutlineEntry::getTextColor (
            IDOMColorPtr & ptrColor ) const  [pure virtual]
```

Retrieves the color to be used for the outline entry's text.

Parameters

ptrColor	Smart pointer to the outline entry text's color.

Returns

bool Returns true on success, false if the call fails.

getTextStyle()

```
virtual eTextStyle IDOMOutlineEntry::getTextStyle ( ) const  [pure virtual]
```

Retrieves the style to be used for the outline entry's text.

See also

eTextStyle

Returns

eTextStyle. Returns the style value for the outline entry's text.

setDescription()

```
virtual bool IDOMOutlineEntry::setDescription (
            const EDLString & strDescription )  [pure virtual]
```

Sets the description of the outline entry node.

Parameters

strDescription	The new descripton for the outline entry node.

Returns

bool Returns true on success, false if the call fails.

setExpanded()

```
virtual void IDOMOutlineEntry::setExpanded (
            bool expanded )  [pure virtual]
```

Sets the outline entry's "expanded" flag.

Parameters

expanded	New value for the outline entry's "expanded" flag.

setLanguage()

```
virtual bool IDOMOutlineEntry::setLanguage (
            const EDLString & strLanguage )  [pure virtual]
```

Retrieves the default language of the outline entry node.

English is defined as en_GB and American English as en_US. There is no default setting. If the language is not known it is set to "und" (undetermined). For further information see http://www.w3.org/↩International/articles/language-tags/. The language is specified according to RFC 3066.

Parameters

strLanguage	Default language

Returns

bool Returns true on success, false if the call fails.

setStructureElement()

```
virtual bool IDOMOutlineEntry::setStructureElement (
            const IEDLObjectPtr & ptrSE )  [pure virtual]
```

Sets the structure element.

Parameters

ptrSE	Smart pointer to the new structure element.

Returns

bool Returns true on success, false if the call fails.

setTarget()

```
virtual bool IDOMOutlineEntry::setTarget (
            const IDOMTargetPtr & ptrTarget )  [pure virtual]
```

Sets the target of the outline entry node. NULL is a valid value of ptrTarget parameter.

Parameters

ptrTarget	Smart pointer to the new target of the outline entry node.

Returns

bool Returns true on success, false if the call fails.

setTextColor()

```
virtual bool IDOMOutlineEntry::setTextColor (
            const IDOMColorPtr & ptrColor )  [pure virtual]
```

Sets the color to be used for the outline entry's text.

Parameters

ptrColor	Smart pointer to the new outline entry text color.

Returns

bool Returns true on success, false if the call fails.

setTextStyle()

```
virtual void IDOMOutlineEntry::setTextStyle (
            eTextStyle style )  [pure virtual]
```

Sets the outline entry's text style.

See also

eTextStyle

Parameters

style	The new value for the outline entry's text style.

The documentation for this class was generated from the following file:

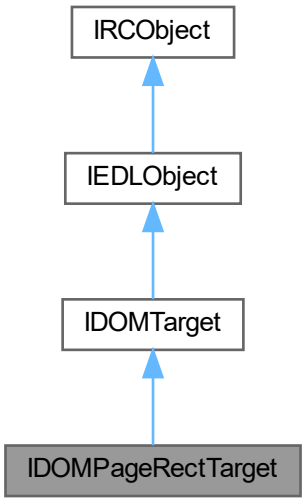
- idomoutline.h

7.243 IDOMPageRectTarget Class Reference

IDOMPageRectTarget nodes are used to describe hyperlinks on a page rectangle to targets on the same page.

`#include <idomtarget.h>`

Inheritance diagram for IDOMPageRectTarget:



Public Types

- enum eFitType

  Destination fit types enumeration.

Public Types inherited from IDOMTarget

- enum eTargetType {
  eExternal , eInternal , ePage , ePageRect ,
  eActionGoToR , eActionGoToE , eActionLaunch , eActionThread ,
  eActionSound , eActionMovie , eActionHide , eActionNamed ,
  eActionSubmitForm , eActionResetForm , eActionImportData , eActionJavaScript ,
  eActionSetOCGState , eActionRendition , eActionTrans , eActionGoTo3DView ,
  eActionArray }

  An enumeration of target types.

Public Member Functions

- virtual DOMid getPageId () const =0

  Retrieves the target page DOM id.
- virtual void setPageId (DOMid domId)=0

  Sets the target page DOM id.
- virtual eFitType getFitType () const =0

  Gets the fit type.
- virtual void setFitType (eFitType fitType)=0

  Sets the fit type.
- virtual double getZoom () const =0

  Gets the zoom value.
- virtual void setZoom (double zoom)=0

  Sets for the zoom value.
- virtual doubleWithNull getLeft () const =0

  Gets the left coordinate value of the target.
- virtual void setLeft (const doubleWithNull &left)=0

  Sets the left coordinate value of the target.
- virtual doubleWithNull getTop () const =0

  Gets the top coordinate value of the target.
- virtual void setTop (const doubleWithNull &top)=0

  Sets the top coordinate value of the target.
- virtual doubleWithNull getRight () const =0

  Gets the right coordinate value of the target.
- virtual void setRight (const doubleWithNull &right)=0

  Sets the right coordinate value of the target.
- virtual doubleWithNull getBottom () const =0

  Gets the bottom coordinate value of the target.
- virtual void setBottom (const doubleWithNull &bottom)=0

  Sets the bottom coordinate value of the target.
- virtual eTargetType getTargetType () const

  Implementation of getTargetType for IDOMPageTarget.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static EDL_API IDOMPageRectTargetPtr create (IEDLClassFactory *pFactory, DOMid pageId, eFitType fitType=eFTFit, double zoom=0.0, const doubleWithNull &left=doubleWithNull(), const doubleWithNull &top=doubleWithNull(), const doubleWithNull &right=doubleWithNull(), const doubleWithNull &bottom=doubleWithNull())

  Simplified creator for a page rect target. Throws an exception of type IEDLError on failure.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.243.1 Detailed Description

IDOMPageRectTarget nodes are used to describe hyperlinks on a page rectangle to targets on the same page.

7.243.2 Member Function Documentation

create()

```
static EDL_API IDOMPageRectTargetPtr IDOMPageRectTarget::create (
            IEDLClassFactory * pFactory,
            DOMid pageId,
            eFitType fitType = eFTFit,
            double zoom = 0.0,
            const doubleWithNull & left = doubleWithNull(),
            const doubleWithNull & top = doubleWithNull(),
            const doubleWithNull & right = doubleWithNull(),
            const doubleWithNull & bottom = doubleWithNull() )  [static]
```

Simplified creator for a page rect target. Throws an exception of type IEDLError on failure.

Parameters

pFactory	The EDL factory to use
pageId	The DOM id of the target page
fitType	The fit type
zoom	The desired zoom level. Zero indicates that the zoom should be unchanged when navigating to this target.
left	The left of the target rect. For certain fit types, this will be ignored.
top	The top of the target rect. For certain fit types, this will be ignored.
right	The right of the target rect. For certain fit types, this will be ignored.
bottom	The left of the target rect. For certain fit types, this will be ignored.

Returns

IDOMPageRectTarget The new target

getBottom()

```
virtual doubleWithNull IDOMPageRectTarget::getBottom ( ) const  [pure virtual]
```

Gets the bottom coordinate value of the target.

Returns

doubleWithNull Returns the bottom coordinate value of the target.

getFitType()

```
virtual eFitType IDOMPageRectTarget::getFitType ( ) const  [pure virtual]
```

Gets the fit type.

See also

eFitType

Returns

eFitType Returns the fit type.

getLeft()

```
virtual doubleWithNull IDOMPageRectTarget::getLeft ( ) const  [pure virtual]
```

Gets the left coordinate value of the target.

Returns

doubleWithNull Returns the left coordinate value of the target.

getPageId()

```
virtual DOMid IDOMPageRectTarget::getPageId ( ) const  [pure virtual]
```

Retrieves the target page DOM id.

Returns

DOMid The target page id

getRight()

```
virtual doubleWithNull IDOMPageRectTarget::getRight ( ) const  [pure virtual]
```

Gets the right coordinate value of the target.

Returns

doubleWithNull Returns the right coordinate value of the target.

getTargetType()

```
virtual eTargetType IDOMPageRectTarget::getTargetType ( ) const  [inline], [virtual]
```

Implementation of getTargetType for IDOMPageTarget.

Returns

eTargetType. Returns "ePage".

Implements IDOMTarget.

getTop()

```
virtual doubleWithNull IDOMPageRectTarget::getTop ( ) const  [pure virtual]
```

Gets the top coordinate value of the target.

Returns

doubleWithNull Returns the top coordinate value of the target.

getZoom()

```
virtual double IDOMPageRectTarget::getZoom ( ) const  [pure virtual]
```

Gets the zoom value.

Returns

double Returns the zoom value. Zero zoom value specifies that the current value of that parameter is to be retained unchanged

setBottom()

```
virtual void IDOMPageRectTarget::setBottom (
            const doubleWithNull & bottom )  [pure virtual]
```

Sets the bottom coordinate value of the target.

Parameters

bottom	The new bottom coordinate value.

setFitType()

```
virtual void IDOMPageRectTarget::setFitType (
            eFitType fitType )  [pure virtual]
```

Sets the fit type.

See also

eFitType

Parameters

fitType	The new fit type.

setLeft()

```
virtual void IDOMPageRectTarget::setLeft (
            const doubleWithNull & left )  [pure virtual]
```

Sets the left coordinate value of the target.

Parameters

left	The new left coordinate value.

setPageId()

```
virtual void IDOMPageRectTarget::setPageId (
            DOMid domId )  [pure virtual]
```

Sets the target page DOM id.

Parameters

domId	The new target page id

setRight()

```
virtual void IDOMPageRectTarget::setRight (
            const doubleWithNull & right )  [pure virtual]
```

Sets the right coordinate value of the target.

Parameters

right	The new right coordinate value.

setTop()

```
virtual void IDOMPageRectTarget::setTop (
            const doubleWithNull & top )  [pure virtual]
```

Sets the top coordinate value of the target.

Parameters

top	The new top coordinate value.

setZoom()

```
virtual void IDOMPageRectTarget::setZoom (
            double zoom )  [pure virtual]
```

Sets for the zoom value.

Parameters

zoom	The new zoom value. Zero zoom value specifies that the current value of that parameter is to be retained unchanged.

The documentation for this class was generated from the following file:

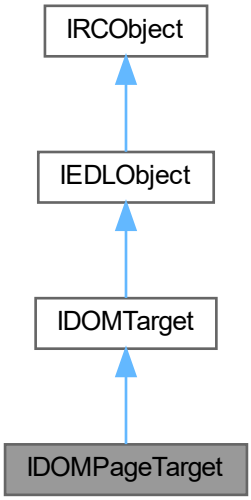
- idomtarget.h

7.244 IDOMPageTarget Class Reference

IDOMPageTarget nodes are used to describe hyperlinks on a page to targets on the same page.

```
#include <idomtarget.h>
```

Inheritance diagram for IDOMPageTarget:



Public Member Functions

- virtual uint32 getTargetPage () const =0

  Retrieves the target page number for a page target. A page target has the whole page as the target. It refers to a page using the absolute page number within the document sequence, where the very first page in the first document is number 1.
- virtual void setTargetPage (uint32 page)=0

  Sets the target page number for a page target. A page target has the whole page as the target. It refers to a page using the absolute page number within the document sequence, where the very first page in the first document is number 1.
- virtual eTargetType getTargetType () const

  Implementation of getTargetType for IDOMPageTarget.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Public Types inherited from IDOMTarget

- enum eTargetType {
  eExternal , eInternal , ePage , ePageRect ,
  eActionGoToR , eActionGoToE , eActionLaunch , eActionThread ,
  eActionSound , eActionMovie , eActionHide , eActionNamed ,
  eActionSubmitForm , eActionResetForm , eActionImportData , eActionJavaScript ,
  eActionSetOCGState , eActionRendition , eActionTrans , eActionGoTo3DView ,
  eActionArray }

  An enumeration of target types.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.244.1 Detailed Description

IDOMPageTarget nodes are used to describe hyperlinks on a page to targets on the same page.

7.244.2 Member Function Documentation

getTargetPage()

```
virtual uint32 IDOMPageTarget::getTargetPage ( ) const  [pure virtual]
```

Retrieves the target page number for a page target. A page target has the whole page as the target. It refers to a page using the absolute page number within the document sequence, where the very first page in the first document is number 1.

Returns

uint32 The target page number.

getTargetType()

```
virtual eTargetType IDOMPageTarget::getTargetType ( ) const  [inline], [virtual]
```

Implementation of getTargetType for IDOMPageTarget.

Returns

eTargetType. Returns "ePage".

Implements IDOMTarget.

setTargetPage()

```
virtual void IDOMPageTarget::setTargetPage (
            uint32 page )  [pure virtual]
```

Sets the target page number for a page target. A page target has the whole page as the target. It refers to a page using the absolute page number within the document sequence, where the very first page in the first document is number 1.

Parameters

page	The new target page number.

The documentation for this class was generated from the following file:

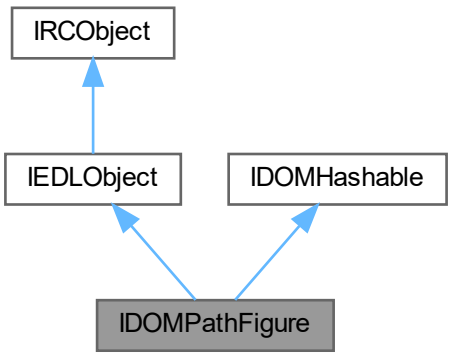
- idomtarget.h

7.245 IDOMPathFigure Class Reference

Interface to the path figure element. A path figure is a single shape comprised of continuous path segments. One or more path figures collectively define an entire path geometry. A path geometry may define the fill algorithm to be used on the component path figures. Instances of this type use exceptions of IEDLError for error handling.

```
#include <idompathgeometry.h>
```

Inheritance diagram for IDOMPathFigure:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual bool getIsClosed () const =0

  Retrieves IsClosed for the path figure. IsClosed specifies whether the path is closed, that is, whether the last point in the last segment of the path figure should be connected to the start point of the figure, otherwise the stroke is drawn open, and the last point is not connected to the start point. This is only applicable if the path figure is used in a Path that specifies a stroke.
- virtual void setIsClosed (bool closed)=0

  Sets IsClosed for the path figure. IsClosed specifies whether the path is closed. When this is set to true it specifies that the path is closed-thatis, the last point in the PathFigure is connected to the first. An exception is thrown if this segment is immutable.
- virtual const FPoint & getStartPoint () const =0

  Retrieves the start point for the first segment in the figure.
- virtual void setStartPoint (const FPoint &sp)=0

  Sets the start point for the first segment in the figure. An exception is thrown if this segment is immutable.
- virtual bool getIsFilled () const =0

  Retrieves IsFilled for the figure. IsFilled specifies whether the path figure is used in computing the area of the containing path geometry. When set to false, the path figure is considered only for stroking.
- virtual void setIsFilled (bool filled)=0

  Sets IsFilled for the figure. IsFilled specifies whether the path figure is used in computing the area of the containing path geometry. When set to false, the path figure is considered only for stroking. An exception is thrown if this segment is immutable.
- virtual const CDOMPathSegmentVect & getSegments () const =0

  Retrieves the collection of segments that comprise this path figure.
- virtual uint32 getSegmentsCount () const =0

  Retrieves the number of path segments in the figure.
- virtual void **clearSegments** ()=0

Removes all path segments from the figure. An exception is thrown if this segment is immutable.

- virtual void addSegment (const IDOMPathSegmentPtr &pathSegment)=0

  Append a path segment to the figure. An exception is thrown if this segment is immutable.
- virtual FRect getBounds () const =0

  Finds the conservative bounding box of the figure.
- virtual bool getIsImmutable () const =0

  Determine if the segment is immutable (non-editable).
- virtual void **setImmutable** ()=0

  Force the segment to be flagged immutable.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves the class id of IDOMPathFigure.
- static EDL_API IDOMPathFigurePtr create (IEDLClassFactory *factory, const FPoint &startPoint=FPoint(), bool isClosed=false, bool isFilled=true, const CDOMPathSegmentVect &segments=CDOMPathSegment↩ Vect())

  Simplified creator for a path figure.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.245.1 Detailed Description

Interface to the path figure element. A path figure is a single shape comprised of continuous path segments. One or more path figures collectively define an entire path geometry. A path geometry may define the fill algorithm to be used on the component path figures. Instances of this type use exceptions of IEDLError for error handling.

See also

IDOMPathNode

IDOMPathSegment

IDOMPathGeometry

7.245.2 Member Function Documentation

addSegment()

```
virtual void IDOMPathFigure::addSegment (
            const IDOMPathSegmentPtr & pathSegment )  [pure virtual]
```

Append a path segment to the figure. An exception is thrown if this segment is immutable.

Parameters

pathSegment	Smart pointer to the path segment to add.

classID()

```
static const CClassID & IDOMPathFigure::classID ( )  [inline], [static]
```

Retrieves the class id of IDOMPathFigure.

Returns

CClassID. Returns the class id of the element.

create()

```
static EDL_API IDOMPathFigurePtr IDOMPathFigure::create (
            IEDLClassFactory * factory,
```

```
const FPoint & startPoint = FPoint(),
bool isClosed = false,
bool isFilled = true,
const CDOMPathSegmentVect & segments = CDOMPathSegmentVect() )  [static]
```

Simplified creator for a path figure.

Parameters

factory	The factory to use.
isClosed	Should the figure be closed if stroking.
isFilled	Should the figure be filled, if used in a filling path.
segments	The segments.

Returns

IDOMPolyLineSegmentPtr The new segment.

getBounds()

```
virtual FRect IDOMPathFigure::getBounds ( ) const  [pure virtual]
```

Finds the conservative bounding box of the figure.

Returns

FRect The conservative bounding box.

getIsClosed()

```
virtual bool IDOMPathFigure::getIsClosed ( ) const  [pure virtual]
```

Retrieves IsClosed for the path figure. IsClosed specifies whether the path is closed, that is, whether the last point in the last segment of the path figure should be connected to the start point of the figure, otherwise the stroke is drawn open, and the last point is not connected to the start point. This is only applicable if the path figure is used in a Path that specifies a stroke.

The default is false.

Returns

bool True if the path is closed.

getIsFilled()

```
virtual bool IDOMPathFigure::getIsFilled ( ) const  [pure virtual]
```

Retrieves IsFilled for the figure. IsFilled specifies whether the path figure is used in computing the area of the containing path geometry. When set to false, the path figure is considered only for stroking.

Returns

bool Returns true if the path is filled, false otherwise.

getIsImmutable()

```
virtual bool IDOMPathFigure::getIsImmutable ( ) const  [pure virtual]
```

Determine if the segment is immutable (non-editable).

Returns

bool True if the segment may not be edited.

getSegments()

```
virtual const CDOMPathSegmentVect & IDOMPathFigure::getSegments ( ) const  [pure virtual]
```

Retrieves the collection of segments that comprise this path figure.

Returns

CDOMSegmentVect the segments.

getSegmentsCount()

```
virtual uint32 IDOMPathFigure::getSegmentsCount ( ) const  [pure virtual]
```

Retrieves the number of path segments in the figure.

Returns

uint32 The number of path segments in the figure.

getStartPoint()

```
virtual const FPoint & IDOMPathFigure::getStartPoint ( ) const  [pure virtual]
```

Retrieves the start point for the first segment in the figure.

Returns

FPoint The start point for the first segment in the figure.

setIsClosed()

```
virtual void IDOMPathFigure::setIsClosed (
            bool closed )  [pure virtual]
```

Sets IsClosed for the path figure. IsClosed specifies whether the path is closed. When this is set to true it specifies that the path is closed-thatis, the last point in the PathFigure is connected to the first. An exception is thrown if this segment is immutable.

The default is false.

Parameters

closed	New value of IsClosed.

setIsFilled()

```
virtual void IDOMPathFigure::setIsFilled (
            bool filled )  [pure virtual]
```

Sets IsFilled for the figure. IsFilled specifies whether the path figure is used in computing the area of the containing path geometry. When set to false, the path figure is considered only for stroking. An exception is thrown if this segment is immutable.

Parameters

filled	New value of IsFilled.

setStartPoint()

```
virtual void IDOMPathFigure::setStartPoint (
            const FPoint & sp )  [pure virtual]
```

Sets the start point for the first segment in the figure. An exception is thrown if this segment is immutable.

Parameters

sp	New value of the start point of the first segment in the figure.

The documentation for this class was generated from the following file:

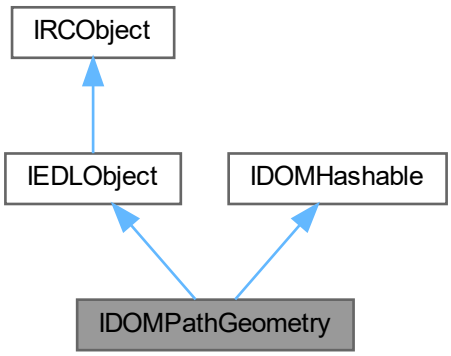
- idompathgeometry.h

7.246 IDOMPathGeometry Class Reference

Interface to a path geometry node.

```
#include <idompathgeometry.h>
```

Inheritance diagram for IDOMPathGeometry:



Classes

- class Data

  Initialization data.

Public Types

- enum eFillRule { eFREvenOdd , eFRNonZero }

  Specifies the algorithm to determine whether or not a point is inside a shape on the canvas.

Public Member Functions

- virtual eFillRule getFillRule () const =0

  Retrieves the fill rule for the path. The valid values are specified by eFillRule.
- virtual void setFillRule (eFillRule fr)=0

  Sets the fill rule for the path. The valid values are specified by eFillRule.
- virtual const FMatrix & getRenderTransform () const =0

  Retrieves the render transform matrix for the path.
- virtual void setRenderTransform (const FMatrix &matrix)=0

  Sets the render transform matrix for the path.
- virtual CDOMPathFigureVect getFigures () const =0

  Retrieves the list of figures comprising the geometry.
- virtual uint32 getFiguresCount () const =0

  Retrieves the number of figures in the geometry.
- virtual void **clearFigures** ()=0

  Removes all the figures from the geometry. An exception is thrown if the figures in this geometry are not editable.
- virtual void addFigure (const IDOMPathFigurePtr &pathFigure)=0

  Appends a figure to the path figure collection. An exception is thrown if the figures in this geometry are not editable.
- virtual bool getFiguresImmutable () const =0

  Determine if the figures immutable (non-editable).

- virtual IDOMPathGeometryPtr getMutableGeometry (IEDLClassFactory *factory) const =0

  Get a mutable version of the path geometry. Will return this object if the geometry is already mutable. Otherwise a new geometry will be created where the figures are editable.
- virtual FRect getBounds (bool applyTransform=true) const =0

  Finds the conservative bounding box of the geometry.
- virtual bool getIsRect (FRect &rect)=0

  Determines if the geometry is a simple rectangle. NB: It does not check to see if the path is closed, only that its shape is a rectangle that is orthogonal to a regular cartesian axis.
- virtual IDOMShapePtr getShape (IEDLClassFactory *factory, const FMatrix &transform, float resolution)=0

  Get the scan-converted shape of this path geometry.
- virtual IDOMPathGeometryPtr getSimplifiedGeometry (IEDLClassFactory *pFactory, bool simplifyQuads, bool simplifyArcs)=0

  Get a simplified version of this path geometry, with certain segment types reduced to simpler (or more common) types as directed. This may provide the called object if the path is already simple.
- virtual IDOMPathGeometryPtr getFlattenedGeometry (IEDLClassFactory *pFactory, const FMatrix &transform, float resolution)=0

  Get a flattened version of the path; that is, with any curves converted to straight line approximations. This may provide the called object if the geometry has no curved segments.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static EDL_API IDOMPathGeometryPtr create (IEDLClassFactory *factory, const FRect &rect, bool close=true, const FMatrix &renderTransform=FMatrix())

  Simplified creator for a rectangular path geometry. The geometry will consist of a path beginning at x,y value of the rect, proceeding first in the y direction.
- static EDL_API IDOMPathGeometryPtr create (IEDLClassFactory *factory, const EDLSysString &abbreviatedGeometry, const FMatrix &renderTransform=FMatrix())

  Simplified creator for a path using XPS abbreviated geometry format.
- static EDL_API IDOMPathGeometryPtr create (IEDLClassFactory *factory, const CDOMPathFigure↩ Vect &figures=CDOMPathFigureVect(), const FMatrix &renderTransform=FMatrix(), eFillRule fill↩ Rule=eFREvenOdd)

  Simplified creator for a path consisting of a series of figures.
- static EDL_API IDOMPathGeometryPtr createEllipse (IEDLClassFactory *factory, const FRect &rect, const FMatrix &renderTransform=FMatrix())

  Simplified creator for a path consisting of a circle/ellipse touching the given rectangle on all sides. The circle/ellipse will be approximated with four cubic beziers, and will proceed in a clockwise direction.
- static EDL_API IDOMPathGeometryPtr createPolygon (IEDLClassFactory *factory, const FRect &rect, uint32 numSides, double rotationAngle=0.0)

  Simplified creator for a regular convex polygon path.
- static const CClassID & classID ()

  Retrieves class id of IDOMPathGeometry.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.246.1 Detailed Description

Interface to a path geometry node.

See also

IDOMPathFigure

IDOMPathNode

```
A path geometry node constitutes a complete geometry definition. The smallest unit in a geometry
is a segment. One or more segments are combined into a path figure definition. A path figure is a
single shape comprised of continuous segments. One or more path figures collectively define an
entire path geometry. A path geometry may define the fill algorithm to be used on the component path fig

A single path geometry node may be used in the data property of the path node to describe its overall
geometry. A path geometry node may also be used in the clip property of the Canvas, Path, or Glyphs
nodes to describe a clipping region.

Instances of this type use exceptions of IEDLError for error handling.
```

7.246.2 Member Function Documentation

addFigure()

```
virtual void IDOMPathGeometry::addFigure (
            const IDOMPathFigurePtr & pathFigure )  [pure virtual]
```

Appends a figure to the path figure collection. An exception is thrown if the figures in this geometry are not editable.

Parameters

pathFigure	Path figure to add.

classID()

```
static const CClassID & IDOMPathGeometry::classID ( )  [inline], [static]
```

Retrieves class id of IDOMPathGeometry.

Returns

CClassID class id of the element.

create() [1/3]

```
static EDL_API IDOMPathGeometryPtr IDOMPathGeometry::create (
            IEDLClassFactory * factory,
            const CDOMPathFigureVect & figures = CDOMPathFigureVect(),
            const FMatrix & renderTransform = FMatrix(),
            eFillRule fillRule = eFREvenOdd )  [static]
```

Simplified creator for a path consisting of a series of figures.

Parameters

factory	The factory to use.
figures	The figures comprising the geometry.
renderTransform	The transform to apply to the geometry.
fillRule	The fill rule to apply to the geometry.

Returns

IDOMPathGeometryPtr The new geometry.

create() [2/3]

```
static EDL_API IDOMPathGeometryPtr IDOMPathGeometry::create (
            IEDLClassFactory * factory,
            const EDLSysString & abbreviatedGeometry,
            const FMatrix & renderTransform = FMatrix() )  [static]
```

Simplified creator for a path using XPS abbreviated geometry format.

Parameters

factory	The factory to use.
abbreviatedGeometry	Path geometry string specified using XPS abbreviated geometry syntax.
renderTransform	The transform to apply to the geometry.

Returns

IDOMPathGeometryPtr The new geometry.

create() [3/3]

```
static EDL_API IDOMPathGeometryPtr IDOMPathGeometry::create (
            IEDLClassFactory * factory,
            const FRect & rect,
            bool close = true,
            const FMatrix & renderTransform = FMatrix() )  [static]
```

Simplified creator for a rectangular path geometry. The geometry will consist of a path beginning at x,y value of the rect, proceeding first in the y direction.

Parameters

factory	The factory to use.
rect	The rect to use.
close	Whether or not the rectangular path should be closed.
renderTransform	The transform to apply to the geometry.

Returns

IDOMPathGeometryPtr The new geometry.

createEllipse()

```
static EDL_API IDOMPathGeometryPtr IDOMPathGeometry::createEllipse (
            IEDLClassFactory * factory,
            const FRect & rect,
            const FMatrix & renderTransform = FMatrix() )  [static]
```

Simplified creator for a path consisting of a circle/ellipse touching the given rectangle on all sides. The circle/ellipse will be approximated with four cubic beziers, and will proceed in a clockwise direction.

Parameters

factory	The factory to use.
rect	The rect describing the bounds of the circle/ellipse.
renderTransform	The transform to apply to the geometry.

Returns

IDOMPathGeometryPtr The new geometry.

createPolygon()

```
static EDL_API IDOMPathGeometryPtr IDOMPathGeometry::createPolygon (
            IEDLClassFactory * factory,
```

```
            const FRect & rect,
            uint32 numSides,
            double rotationAngle = 0.0 )  [static]
```

Simplified creator for a regular convex polygon path.

Parameters

factory	The factory to use.
rect	The rect describing the bounds of the polygon.
numSides	The desired number of sides.
rotationAngle	The desired rotation angle in degrees.

Returns

IDOMPathGeometryPtr The new geometry.

getBounds()

```
virtual FRect IDOMPathGeometry::getBounds (
            bool applyTransform = true ) const  [pure virtual]
```

Finds the conservative bounding box of the geometry.

Parameters

applyTransform	Controls whether or not the receiver's transform is applied to the bounds (if it has one). Passing true (default) gives you results in the coordinate space of the object enclosing the receiver, while passing false will give you results in the coordinate space active inside the object.

Returns

FRect The conservative bounds.

getFigures()

```
virtual CDOMPathFigureVect IDOMPathGeometry::getFigures ( ) const  [pure virtual]
```

Retrieves the list of figures comprising the geometry.

Returns

CDOMPathFigureVect The path figures.

getFiguresCount()

```
virtual uint32 IDOMPathGeometry::getFiguresCount ( ) const  [pure virtual]
```

Retrieves the number of figures in the geometry.

Returns

uint32 Returns the number of figures in the path collection.

getFiguresImmutable()

```
virtual bool IDOMPathGeometry::getFiguresImmutable ( ) const  [pure virtual]
```

Determine if the figures immutable (non-editable).

Returns

bool True if the figures may not be edited.

getFillRule()

```
virtual eFillRule IDOMPathGeometry::getFillRule ( ) const  [pure virtual]
```

Retrieves the fill rule for the path. The valid values are specified by eFillRule.

```
The FillRule attribute specifies a fill algorithm. The fillable area of the geometry is
defined by taking all of the contained PathFigures and applying the fill algorithm to
determine the enclosed area. Fill algorithms determine how the intersecting areas of
geometric shapes are combined to form a region.
```

Returns

eFillRule Fill rule (eRFEvenOdd default).

getFlattenedGeometry()

```
virtual IDOMPathGeometryPtr IDOMPathGeometry::getFlattenedGeometry (
            IEDLClassFactory * pFactory,
            const FMatrix & transform,
            float resolution )  [pure virtual]
```

Get a flattened version of the path; that is, with any curves converted to straight line approximations. This may provide the called object if the geometry has no curved segments.

Parameters

pFactory	A pointer to the EDL class factory.
transform	The external transformation applied to the geometry. Needed to determine the visible size of the geometry, and hence influences the level of curve approximation.
resolution	The intended display resolution. The flattened geometry will be computed such that the curved approximation is not noticeable when viewed at that resolution.

Returns

IDOMPathGeometryPtr The resulting flattened geometry, or this geometry if no flattening is required.

getIsRect()

```
virtual bool IDOMPathGeometry::getIsRect (
            FRect & rect )  [pure virtual]
```

Determines if the geometry is a simple rectangle. NB: It does not check to see if the path is closed, only that its shape is a rectangle that is orthogonal to a regular cartesian axis.

Parameters

rect	Reference to receive the dimensions of the rectangle if the geometry indeed represents a rectangle.

Returns

bool. True if the geometry is a simple orthogonal rectangle.

getMutableGeometry()

```
virtual IDOMPathGeometryPtr IDOMPathGeometry::getMutableGeometry (
            IEDLClassFactory * factory ) const  [pure virtual]
```

Get a mutable version of the path geometry. Will return this object if the geometry is already mutable. Otherwise a new geometry will be created where the figures are editable.

Parameters

factory	A pointer to the EDL class factory.

Returns

IDOMPathGometryPtr The mutable geometry.

getRenderTransform()

```
virtual const FMatrix & IDOMPathGeometry::getRenderTransform ( ) const  [pure virtual]
```

Retrieves the render transform matrix for the path.

Returns

FMatrix The transformation.

getShape()

```
virtual IDOMShapePtr IDOMPathGeometry::getShape (
            IEDLClassFactory * factory,
            const FMatrix & transform,
            float resolution )  [pure virtual]
```

Get the scan-converted shape of this path geometry.

Parameters

factory	A pointer to the EDL class factory.
transform	The transform that should be applied before scan conversion.
resolution	The resolution that should be used for scan conversion.

Returns

IDOMShapePtr The scan-converted shape.

getSimplifiedGeometry()

```
virtual IDOMPathGeometryPtr IDOMPathGeometry::getSimplifiedGeometry (
            IEDLClassFactory * pFactory,
            bool simplifyQuads,
            bool simplifyArcs )  [pure virtual]
```

Get a simplified version of this path geometry, with certain segment types reduced to simpler (or more common) types as directed. This may provide the called object if the path is already simple.

Parameters

pFactory	A pointer to the EDL class factory.
simplifyQuads	If true, any quadratic segments will be converted to regular cubic segments in the simplified geometry.
simplifyArcs	If true, any arc segments will be converted to regular cubic segments or line segments as appropriate.

Returns

IDOMPathGeometryPtr The simplified path geometry.

setFillRule()

```
virtual void IDOMPathGeometry::setFillRule (
            eFillRule fr )  [pure virtual]
```

Sets the fill rule for the path. The valid values are specified by eFillRule.

```
The FillRule attribute specifies a fill algorithm. The fillable area of the geometry is
defined by taking all of the contained PathFigures and applying the fill algorithm to
determine the enclosed area. Fill algorithms determine how the intersecting areas of
geometric shapes are combined to form a region.
```

Parameters

fr	Fill rule.

setRenderTransform()

```
virtual void IDOMPathGeometry::setRenderTransform (
            const FMatrix & matrix )  [pure virtual]
```

Sets the render transform matrix for the path.

Parameters

matrix	The new render transform matrix.

The documentation for this class was generated from the following file:

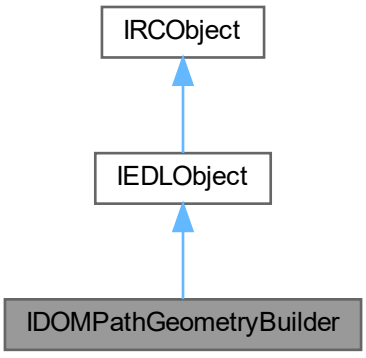
- idompathgeometry.h

7.247 IDOMPathGeometryBuilder Class Reference

Interface to a path geometry builder.

```
#include <idompathgeometry.h>
```

Inheritance diagram for IDOMPathGeometryBuilder:



Public Member Functions

- virtual void moveTo (const FPoint &point, bool absolute=true)=0

  Move to the given user point. Throws an IEDLError exception on failure.
- virtual void lineTo (const FPoint &point, bool absolute=true, bool stroked=true)=0

  Append a line segment to the current path. Throws an IEDLError exception on failure.
- virtual void curveTo (const FPoint &control0, const FPoint &control1, const FPoint &point, bool absolute=true, bool stroked=true)=0

  Append a section of a cubic Bézier curve to the current path. Throws an IEDLError exception on failure.
- virtual void quadCurveTo (const FPoint &control, const FPoint &point, bool absolute=true, bool stroked=true)=0

  Append a section of a quadratic Bézier curve to the current path. Throws an IEDLError exception on failure.
- virtual void arcTo (const FPoint &radius, double rotation, bool isLargeArc, IDOMArcSegment::eSweepDirection sweepDirection, const FPoint &point, bool absolute=true, bool stroked=true)=0

  Append an elliptical arc segment to the current path. Throws an IEDLError exception on failure.
- virtual void close (bool stroked=true)=0

  Close the subpath, if it is not already. Throws an IEDLError exception on failure.
- virtual IDOMPathGeometryPtr createGeometry (IEDLClassFactory *pFactory, IDOMPathGeometry::eFillRule fillRule, bool ignoreDegenerateSubpaths=false, const FMatrix &renderTransform=FMatrix()) const =0

  Get the path as geometry, in user units according to the given CTM and fill rule. Throws an IEDLError exception on failure.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves class id of IDOMPathGeometryBuilder.
- static EDL_API IDOMPathGeometryBuilderPtr create (IEDLClassFactory *factory)

  Simplified creation function for IDOMPathGeometryBuilder. Throws an IEDLError exception on failure.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()
  Virtual destructor.

7.247.1 Detailed Description

Interface to a path geometry builder.

See also

IDOMPathGeometry

```
A path geometry builder can be used to build geometry using PostScript style path construction commands.

Instances of this type use exceptions of IEDLError for error handling.
```

7.247.2 Member Function Documentation

arcTo()

```
virtual void IDOMPathGeometryBuilder::arcTo (
            const FPoint & radius,
            double rotation,
            bool isLargeArc,
            IDOMArcSegment::eSweepDirection sweepDirection,
            const FPoint & point,
            bool absolute = true,
            bool stroked = true )  [pure virtual]
```

Append an elliptical arc segment to the current path. Throws an IEDLError exception on failure.

Parameters

radius	The x and y radius of the arc.
rotation	The rotation angle.
isLargeArc	Whether or not the large arc should be used.
sweepDirection	The direction of the sweep - clockwise or counter clockwise - between the start and end points.
point	The end point of the arc.
absolute	Whether the points are specified in absolute coordinates. The default value is true.
stroked	Whether the stroke for this segment of the path is drawn. The default value is true.

classID()

```
static const CClassID & IDOMPathGeometryBuilder::classID ( )  [inline], [static]
```

Retrieves class id of IDOMPathGeometryBuilder.

Returns

CClassID class id of the path geometry builder.

close()

```
virtual void IDOMPathGeometryBuilder::close (
            bool stroked = true )  [pure virtual]
```

Close the subpath, if it is not already. Throws an IEDLError exception on failure.

Parameters

stroked	Whether the stroke for this segment of the path is drawn. The default value is true.

create()

```
static EDL_API IDOMPathGeometryBuilderPtr IDOMPathGeometryBuilder::create (
            IEDLClassFactory * factory )  [static]
```

Simplified creation function for IDOMPathGeometryBuilder. Throws an IEDLError exception on failure.

Parameters

factory	The EDL class factory.

Returns

IDOMPathGeometryBuilderPtr A smart pointer to the geometry builder.

createGeometry()

```
virtual IDOMPathGeometryPtr IDOMPathGeometryBuilder::createGeometry (
            IEDLClassFactory * pFactory,
            IDOMPathGeometry::eFillRule fillRule,
            bool ignoreDegenerateSubpaths = false,
            const FMatrix & renderTransform = FMatrix() ) const  [pure virtual]
```

Get the path as geometry, in user units according to the given CTM and fill rule. Throws an IEDLError exception on failure.

Parameters

fillRule	The fill rule.
ignoreDegenerateSubpaths	Whether to generate ignore degenerate subpaths. The default value is false.
renderTransform	The transform to apply to the geometry.

Returns

IDOMPathGeometryPtr The path geometry.

curveTo()

```
virtual void IDOMPathGeometryBuilder::curveTo (
            const FPoint & control0,
            const FPoint & control1,
            const FPoint & point,
            bool absolute = true,
            bool stroked = true )  [pure virtual]
```

Append a section of a cubic Bézier curve to the current path. Throws an IEDLError exception on failure.

Parameters

control0	The first control point used to define the curve.
control1	The second control point used to define the curve.
point	The new end point.
absolute	Whether the points are specified in absolute coordinates. The default value is true.
stroked	Whether the stroke for this segment of the path is drawn. The default value is true.

lineTo()

```
virtual void IDOMPathGeometryBuilder::lineTo (
            const FPoint & point,
            bool absolute = true,
            bool stroked = true )  [pure virtual]
```

Append a line segment to the current path. Throws an IEDLError exception on failure.

Parameters

point	The new end point.
absolute	Whether the points are specified in absolute coordinates. The default value is true.
stroked	Whether the stroke for this segment of the path is drawn. The default value is true.

moveTo()

```
virtual void IDOMPathGeometryBuilder::moveTo (
            const FPoint & point,
            bool absolute = true )  [pure virtual]
```

Move to the given user point. Throws an IEDLError exception on failure.

Parameters

point	The point to move to.
absolute	Whether the point is specified in absolute coordinates. The default value is true.

quadCurveTo()

```
virtual void IDOMPathGeometryBuilder::quadCurveTo (
            const FPoint & control,
            const FPoint & point,
            bool absolute = true,
            bool stroked = true )  [pure virtual]
```

Append a section of a quadratic Bézier curve to the current path. Throws an IEDLError exception on failure.

Parameters

control	The control point used to define the curve.
point	The new end point.
absolute	Whether the points are specified in absolute coordinates. The default value is true.
stroked	Whether the stroke for this segment of the path is drawn. The default value is true.

The documentation for this class was generated from the following file:

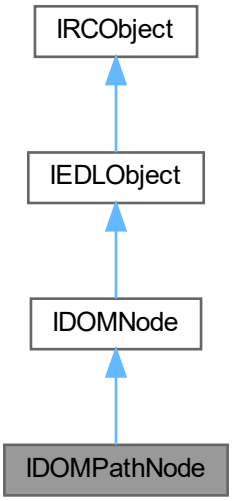
- idompathgeometry.h

7.248 IDOMPathNode Class Reference

Interface to an EDL path node. A path node specifies a geometry that can be filled with a brush.

`#include <idompath.h>`

Inheritance diagram for IDOMPathNode:



Classes

- class Data

  Initialization data.

Public Types

- enum eStrokeLineJoin { eMiterJoin , eBevelJoin , eRoundJoin }

  Specifies the different ways in which the lines in the path could be joined.
- enum eStrokeMiterLimitTreatment { eClipLongMiters , eBevelLongMiters }

  Chooses how miters that extend beyond the miter limit should be treated. ClipLongMiters specifies XPS style behaviour, where miters extending beyond the limit are clipped to the limit. BevelLongMiters specifies PDF/PS style behaviour where miters longer than the limit are instead replaced with a bevel join.
- enum eStrokeLineCap { eFlatCap , eSquareCap , eRoundCap , eTriangleCap }

  Specifies the different types of line end caps available.

Public Member Functions

- virtual const FMatrix & getRenderTransform () const =0

  Retrieves the render transform matrix. The render transform matrix establishes a new coordinate frame for all attributes of the path and for all child elements of the path, such as the path geometry.
- virtual void setRenderTransform (const FMatrix &matrix)=0

  Sets the render transform matrix. The render transform matrix establishes a new coordinate frame for all attributes of the path and for all child elements of the path, such as the path geometry.
- virtual float getOpacity () const =0

  Retrieves the opacity value of the path. The opacity value defines the uniform transparency of the path. This is a number between 0 (fully transparent) and 1 (fully opaque). Default value 1.0.
- virtual void setOpacity (float opc)=0

  Sets the opacity value of the path. The opacity value defines the uniform transparency of the path. This is a number between 0 (fully transparent) and 1 (fully opaque).
- virtual eBlendMode getBlendMode () const =0

  Get the blend mode to be used for rendering this path.
- virtual void setBlendMode (eBlendMode blendMode)=0

  Set the blend mode to be used for rendering this path. Note: modes other than Normal are not directly representable in XPS.
- virtual eEdgeMode getEdgeMode () const =0

  Retrieves render options edge mode of the path.
- virtual void setEdgeMode (eEdgeMode em)=0

  Sets render the options edge mode of the path.
- virtual double getStrokeThickness () const =0

  Retrieves the stroke thickness. The stroke thickness specifies the thickness of a stroke, in units of the effective coordinate space including the path's render transform. The stroke is drawn on top of the boundary of the geometry specified by the path geometry information. Half of the stroke thickness extends outside of the geometry and the other half extends inside of the geometry.
- virtual void setStrokeThickness (double st)=0

  Sets stroke thickness The stroke thickness specifies the thickness of a stroke, in units of the effective coordinate space including the path's render transform. The stroke is drawn on top of the boundary of the geometry specified by the path geometry information. Half of the stroke thickness extends outside of the geometry and the other half extends inside of the geometry.
- virtual bool getShouldZeroWidthLinesBeVisible () const =0

  Should zero width strokes be visible as cosmetic lines? Zero width lines in XPS files are not visible, but in PDF and PostScript, such lines are rendered as a single pixel cosmetic line. This routine returns true if a zero width stroke for this path should be rendered.

- virtual void setShouldZeroWidthLinesBeVisible (bool visible)=0

  Sets whether or not zero width strokes in this path should be rendered visibly as a cosmetic line.
- virtual double getStrokeMiterLimit () const =0

  Retrieves the stroke miter limit. The stroke miter limit is the ratio between the maximum miter length and half of the stroke thickness. This value must be equal to or greater than 1.0. The value is significant only if the StrokeLineJoin attribute specifies mitered joins.
- virtual void setStrokeMiterLimit (double sml)=0

  Sets the stroke miter limit. The stroke miter limit is the ratio between the maximum miter length and half of the stroke thickness. This value must be equal to or greater than 1.0. The value is significant only if the StrokeLineJoin attribute specifies mitered joins.
- virtual eStrokeMiterLimitTreatment getStrokeMiterLimitTreatment () const =0

  Retrieves the miter limit treatment for this path. The stroke miter treatment specifies how miters extending beyond the limit should be treated.
- virtual void setStrokeMiterLimitTreatment (eStrokeMiterLimitTreatment treatment)=0

  Sets the stroke miter limit treatment. The stroke miter treatment specifies how miters extending beyond the limit should be treated.
- virtual double getStrokeDashOffset () const =0

  Retrieves the stroke dash offset value. This adjusts the start point for repeating the dash array pattern. If this value is omitted, the dash array aligns with the origin of the stroke. Values are specified as multiples of the stroke thickness.
- virtual void setStrokeDashOffset (double sdo)=0

  Sets the stroke dash offset value. This adjusts the start point for repeating the dash array pattern. If this value is omitted, the dash array aligns with the origin of the stroke. Values are specified as multiples of the stroke thickness.
- virtual eStrokeLineJoin getStrokeLineJoin () const =0

  Retrieves the stroke line join value. The stroke line join specifies how a stroke is drawn at a corner of a path. Valid values are specified by eStrokeLineJoin. If mitered joins are selected, the value of StrokeMiterLimit is used in drawing the stroke.
- virtual void setStrokeLineJoin (eStrokeLineJoin slj)=0

  Sets the stroke line join value. The stroke line join specifies how a stroke is drawn at a corner of a path. Valid values are specified by eStrokeLineJoin. If mitered joins are selected, the value of StrokeMiterLimit is used in drawing the stroke.
- virtual eStrokeLineCap getStrokeStartLineCap () const =0

  Retrieves the line cap type for the start of the stroke.
- virtual void setStrokeStartLineCap (eStrokeLineCap slc)=0

  Sets the line cap type for the start of the stroke.
- virtual eStrokeLineCap getStrokeEndLineCap () const =0

  Retrieves the line cap type for the end of the stroke.
- virtual void setStrokeEndLineCap (eStrokeLineCap slc)=0

  Sets the line cap type for the end of the stroke.
- virtual eStrokeLineCap getStrokeDashLineCap () const =0

  Gets the stroke dash line cap.
- virtual void setStrokeDashLineCap (eStrokeLineCap slc)=0

  Sets the stroke dash line cap.
- virtual bool getSnapsToDevicePixels () const =0

  Retrieves the snapsToDevicePixels setting.
- virtual void setSnapsToDevicePixels (bool s2dp)=0

  Sets snapsToDevicePixels.
- virtual EDLString getLanguage () const =0

  Retrieves the default language of the path node and any of its children.
- virtual void setLanguage (const EDLString &lang)=0

  Sets default language of the <Path> element and any of its children.
- virtual IDOMTargetPtr getNavigateLink () const =0

  Retrieves the target of a hyperlink.
- virtual void setNavigateLink (const IDOMTargetPtr &target)=0

Sets the target of a hyperlink.

- virtual EDLString getAutomationPropertiesName () const =0

  Retrieves the automation properties name of the path.
- virtual void setAutomationPropertiesName (const EDLString &propname)=0

  Sets the automation properties name of the path.
- virtual EDLString getAutomationPropertiesHelpText () const =0

  Retrieves the automation properties help text of the path.
- virtual void setAutomationPropertiesHelpText (const EDLString &helptext)=0

  Sets the automation properties help text of the path.
- virtual bool getIsDashed ()=0

  Checks to see if the path is dashed. Even if a dash array is provided it may still effectively represent a plain un-dashed line. In this case this member will return false.
- virtual const CEDLVector< double > & getStrokeDashPattern () const =0

  Retrieves the stroke dash array.
- virtual void setStrokeDashPattern (const CEDLVector< double > &pattern)=0

  Set the stroke dash array.
- virtual uint32 getStrokeDashsCount ()=0

  Retrieves the length of the stroke dash array.
- virtual void **clearStrokeDashCollection** ()=0

  Clears the stroke dash array.
- virtual void addStrokeDash (double value)=0

  Append a stroke dash to the stroke dash array.
- virtual IDOMBrushPtr getFill () const =0

  Retrieves the fill brush for the path.
- virtual void setFill (const IDOMBrushPtr &ptrFill)=0

  Sets the fill brush for the path.
- virtual IDOMBrushPtr getStroke () const =0

  Retrieves the stroke brush for the path.
- virtual void setStroke (const IDOMBrushPtr &ptrStroke)=0

  Sets the stroke brush for the path.
- virtual IDOMBrushPtr getOpacityMask () const =0

  Retrieves the opacity mask for the path.
- virtual void setOpacityMask (const IDOMBrushPtr &ptrOpacityMask)=0

  Sets the opacity mask for the path.
- virtual IDOMPathGeometryPtr getPathData () const =0

  Retrieves a smart pointer to the path geometry node.
- virtual void setPathData (const IDOMPathGeometryPtr &ptrPathData)=0

  Sets the path geometry node.
- virtual IDOMPathGeometryPtr getClip () const =0

  Retrieves a smart pointer to the clip geometry node.
- virtual void setClip (const IDOMPathGeometryPtr &ptrClip)=0

  Sets the clip geometry.
- virtual IDOMNodePtr split (IEDLClassFactory *pFactory)=0

  If the path represents both a fill and a stroke, separate the fill and stroke into separate paths. Throws an IEDLError on failure.
- virtual IDOMShapePtr getShape (const FMatrix &transform, float resolution, IEDLClassFactory *pFactory)=0

  Get the scan-converted shape of this path.

Public Member Functions inherited from IDOMNode

- virtual ~**IDOMNode** ()

  virtual destructor
- virtual DOMid getDOMid () const =0

  Retrieves the node ID.
- virtual void setDOMid (DOMid id)=0

  Sets the node ID.
- virtual eDOMNodeType getNodeType () const =0

  Retrieves the DOM node type.
- virtual bool getProperty (const EDLSysString &propertyName, PValue &propertyValue) const =0

  Retrieves the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.
- virtual void setProperty (const EDLSysString &propertyName, const PValue &propertyValue)=0

  Sets the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.
- virtual void removeProperty (const EDLSysString &propertyName)=0

  Removes property.
- virtual IEDLSysStringCollectionEnumPtr getPropertyCollectionEnum ()=0

  Retrieves a navigable list of the property names stored on this node.
- virtual bool hasChildNodes () const =0

  Function that indicates whether this node is a parent to other nodes.
- virtual IDOMNodePtr getParentNode () const =0

  Gets the parent node of this node.
- virtual IDOMNodePtr getFirstChild () const =0

  Gets the first child node of this node.
- virtual IDOMNodePtr getLastChild () const =0

  Gets the last child node of this node.
- virtual IDOMNodePtr getNextChild (const IDOMNodePtr &child) const =0

  Gets the child node which follows the node passed in.
- virtual IDOMNodePtr getPreviousChild (const IDOMNodePtr &child) const =0

  Gets the child node which precedes the node passed in.
- virtual IDOMNodePtr getPreviousSibling () const =0

  Retrieves the node's previous sibling node.
- virtual IDOMNodePtr getNextSibling () const =0

  Retrieves node's next sibling node.
- virtual void appendChild (const IDOMNodePtr &child)=0

  Appends a node to the end of the node's child list.
- virtual void insertChild (const IDOMNodePtr &ptrPreviousSibling, const IDOMNodePtr &child, bool bCheck↩ Complete=true)=0

  Insert a child node after ptrPreviousSibling.
- virtual IDOMNodePtr extractChild (const IDOMNodePtr &child)=0

  Extracts (that is, finds and removes) a child node from the node children. After extraction the child node is no longer a part of the DOM. If no node is specified, the first available node will be extracted from the node's children.
- virtual void replaceChild (const IDOMNodePtr &oldChild, const IDOMNodePtr &newChild)=0

  Replaces the child node with another.
- virtual bool isComplete () const =0

  Signals the completeness of the node.
  A complete node is one that has no more children to be added to it.
- virtual void **setComplete** ()=0

Sets the node's completeness status to "true".

- virtual IDOMNodeFlags * getFlags ()=0

  Retrieves the node's flags property.
- virtual void setParentNode (const IDOMNodePtr &ptrParent)=0

  Sets the parent node.
- virtual void setPreviousSibling (const IDOMNodePtr &ptrPreviousSibling)=0

  Sets the previous sibling node.
- virtual void setNextSibling (const IDOMNodePtr &ptrNextSibling)=0

  Sets the next sibling node.
- virtual bool isAncestor (const IDOMNodePtr &ptrCandidate)=0

  Function tests whether a candidate node is a descendant of the node.
- virtual FRect getBounds (bool applyTransform=true, bool applyClip=true)

  Find the conservative bounding box of the marking content of the node.
- virtual bool copyNodeData (IDOMNode *pSourceNode)=0

  Copy the properties collection, the flags and the DOM ID from the given source node to this one.
- virtual IDOMNodePtr cloneNode (IEDLClassFactory *pFactory) const =0

  Simpified node cloning. An exception of type IEDLError will be thrown on failure.
- virtual IDOMNodePtr cloneTree (IEDLClassFactory *pFactory) const =0

  Clone the tree of nodes beginning at this node. An exception of type IEDLError will be thrown on failure.
- virtual void cloneTreeAndAppend (IEDLClassFactory *pFactory, const IDOMNodePtr &dest) const =0

  Clone the tree of nodes beginning at this node, and append the result to the destination tree.
- virtual void **completeTree** ()=0

  Mark the entire tree from this point as complete. You should not ordinarily need to call this function.
- virtual void **removeCompleteFlagFromTree** ()=0

  Mark the entire tree from this point as complete.
- virtual void findChildrenOfType (eDOMNodeType type, CDOMNodeVect &nodes, bool searchForms=false)=0

  Find all children of this node with the given type, appending to the given vector. Does not descend into brushes.
- virtual void walkTree (WalkTreeFunc func, void *priv, bool descendIntoBrushes=false, bool descendInto↩ Forms=false)=0

  Walk through the DOM calling a given function on each node. The function is allowed to:
- virtual void notifyOnDestruct (NodeDeleteFunc func, void *priv)=0

  Register interest in being told when this node is about to be destroyed.
- virtual void unregisterNotify (NodeDeleteFunc func, void *priv)=0

  Unregister interest in being told when this node is about to be destroyed.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves class id of IDOMPathNode.
- static EDL_API IDOMPathNodePtr createFilled (IEDLClassFactory *pFactory, const IDOMPath↩ GeometryPtr &geometry, const IDOMBrushPtr &brush, const FMatrix renderTransform=FMatrix(), const IDOMPathGeometryPtr &clip=IDOMPathGeometryPtr(), float opacity=1.0f, eBlendMode blend↩ Mode=eBlendModeNormal, eEdgeMode edgeMode=eEMDefault)

  Simplified creator for a filled path. Throws an IEDLError on failure.
- static EDL_API IDOMPathNodePtr createStroked (IEDLClassFactory *pFactory, const IDOMPath↩ GeometryPtr &geometry, const IDOMBrushPtr &brush, const FMatrix renderTransform=FMatrix(), const IDOMPathGeometryPtr &clip=IDOMPathGeometryPtr(), double strokeThickness=1.0, double miter↩ Limit=10.0, eStrokeLineJoin join=eMiterJoin, eStrokeLineCap startCap=eFlatCap, eStrokeLineCap end↩ Cap=eFlatCap, eStrokeLineCap dashCap=eFlatCap, eStrokeMiterLimitTreatment miterTreatment=eClipLongMiters, double dashOffset=0.0, const CEDLVector< double > dashPattern=CEDLVector< double >(), bool zero↩ WidthLinesAreVisible=false, float opacity=1.0f, eBlendMode blendMode=eBlendModeNormal, eEdgeMode edgeMode=eEMDefault)

  Simplified creator for a stroked path. Throws an IEDLError on failure.
- static EDL_API IDOMPathNodePtr createImage (IEDLClassFactory *pFactory, const IDOMImagePtr &image, const FRect &viewPort, const FMatrix &renderTransform=FMatrix())

  Convenience creator for a path containing a rectangular image. Throws an IEDLError on failure.

Static Public Member Functions inherited from IDOMNode

- static EDL_API FMatrix effectiveTransformationOfNode (const IDOMNodePtr &node)

  Attempt to find the effective transformation matrix external to the specified node relative to either a containing page or ultimate parent.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.248.1 Detailed Description

Interface to an EDL path node. A path node specifies a geometry that can be filled with a brush.

Vector graphics are created using the PathNode class. A full set of properties is available to describe the visual characteristics of the graphic. The description of the geometry of the path is described by the Data property. Raster images are included in fixed page markup by specifying a PathNode filled with an ImageBrush.

A PathNode is the sole means of adding vector graphics and images to a fixed page. It defines a single vector graphic to be rendered on a page. Some properties of the PathNode are composable, meaning that the markings rendered to the page are determined by a combination of the property and all of the like-named properties of its parent and ancestor elements.

7.248.2 Member Function Documentation

addStrokeDash()

```
virtual void IDOMPathNode::addStrokeDash (
            double value )  [pure virtual]
```

Append a stroke dash to the stroke dash array.

Parameters

value	The new stroke dash array item.

classID()

```
static const CClassID & IDOMPathNode::classID ( )  [inline], [static]
```

Retrieves class id of IDOMPathNode.

Returns

CClassID Class id of the element

createFilled()

```
static EDL_API IDOMPathNodePtr IDOMPathNode::createFilled (
            IEDLClassFactory * pFactory,
            const IDOMPathGeometryPtr & geometry,
            const IDOMBrushPtr & brush,
            const FMatrix renderTransform = FMatrix(),
            const IDOMPathGeometryPtr & clip = IDOMPathGeometryPtr(),
            float opacity = 1.0f,
            eBlendMode blendMode = eBlendModeNormal,
            eEdgeMode edgeMode = eEMDefault )  [static]
```

Simplified creator for a filled path. Throws an IEDLError on failure.

Parameters

pFactory	The factory to use.
geometry	The path data/geometry to fill.
brush	The fill brush to use.
renderTransform	The render transform to use.
clip	The geometry to use for clipping. NULL if no clip.
opacity	The opacity to use.
blendMode	The blend mode to use.
edgeMode	The edge mode to use.

Returns

IDOMPathNodePtr The new path.

createImage()

```
static EDL_API IDOMPathNodePtr IDOMPathNode::createImage (
            IEDLClassFactory * pFactory,
```

```
            const IDOMImagePtr & image,
            const FRect & viewPort,
            const FMatrix & renderTransform = FMatrix() )  [static]
```

Convenience creator for a path containing a rectangular image. Throws an IEDLError on failure.

The resulting path will have an image brush which paints the entire image scaled to the given destination rectangle.

Parameters

pFactory	The factory to use.
image	The image to use.
viewPort	The destination rectangle.
renderTransform	The render transform to apply.

Returns

IDOMPathNodePtr The new path.

createStroked()

```
static EDL_API IDOMPathNodePtr IDOMPathNode::createStroked (
            IEDLClassFactory * pFactory,
            const IDOMPathGeometryPtr & geometry,
            const IDOMBrushPtr & brush,
            const FMatrix renderTransform = FMatrix(),
            const IDOMPathGeometryPtr & clip = IDOMPathGeometryPtr(),
            double strokeThickness = 1.0,
            double miterLimit = 10.0,
            eStrokeLineJoin join = eMiterJoin,
            eStrokeLineCap startCap = eFlatCap,
            eStrokeLineCap endCap = eFlatCap,
            eStrokeLineCap dashCap = eFlatCap,
            eStrokeMiterLimitTreatment miterTreatment = eClipLongMiters,
            double dashOffset = 0.0,
            const CEDLVector< double > dashPattern = CEDLVector< double >(),
            bool zeroWidthLinesAreVisible = false,
            float opacity = 1.0f,
            eBlendMode blendMode = eBlendModeNormal,
            eEdgeMode edgeMode = eEMDefault )  [static]
```

Simplified creator for a stroked path. Throws an IEDLError on failure.

Parameters

pFactory	The factory to use.
geometry	The path data/geometry to fill.
brush	The fill brush to use.
renderTransform	The render transform to use.
clip	The geometry to use for clipping. NULL if no clip.
strokeThickness	The stroke thickness. If this value is 0 and zeroWidthLinesAreVisible then treat the stroke as a hairline stroke.
miterLimit	The miter limit to be used, if applicable

Parameters

join	The line join to use
startCap	The starting line cap
endCap	The ending line cap
dashCap	The line cap to be used for dashed sections
miterTreatment	How to treat miters that exceed the miter limit (see setStrokeMiterLimit() for details)
dashOffset	The dash offset to use if dashing is to be used
dashPattern	The dash pattern to be used. Pass an empty vector is no dashing is required. If strokeThickness is 0 then this is in user units, otherwise this is scaled up by the strokeThickness.
zeroWidthLinesAreVisible	Set to true if zero width lines should mark (see getShouldZeroWidthLinesBeVisible() for details)
opacity	The opacity to use.
edgeMode	The edge mode to use.
blendMode	The blend mode to use.

Returns

IDOMPathNodePtr The new path.

getAutomationPropertiesHelpText()

```
virtual EDLString IDOMPathNode::getAutomationPropertiesHelpText ( ) const  [pure virtual]
```

Retrieves the automation properties help text of the path.

Automation properties help text is a detailed description of the path's content, used for accessibility purposes, particularly if the path is filled with an image brush, or a set of vector graphics and text elements intended to comprise a single vector graphic.

Returns

EDLString Automation properties help text

getAutomationPropertiesName()

```
virtual EDLString IDOMPathNode::getAutomationPropertiesName ( ) const  [pure virtual]
```

Retrieves the automation properties name of the path.

The automation properties name is a brief description of the path's content, used for accessibility purposes, particularly if the path is filled with an image brush, or a set of vector graphics and text elements intended to comprise a single vector graphic.

Returns

EDLString Automation properties name

getBlendMode()

```
virtual eBlendMode IDOMPathNode::getBlendMode ( ) const  [pure virtual]
```

Get the blend mode to be used for rendering this path.

Returns

eBlendMode The blend mode.

getClip()

```
virtual IDOMPathGeometryPtr IDOMPathNode::getClip ( ) const  [pure virtual]
```

Retrieves a smart pointer to the clip geometry node.

The clip geometry node specifies a clipping region which describes the geometric area to be preserved. The remainder is not rendered.

Returns

IDOMPathGeometryPtr The clip, or NULL if there is no clip.

getEdgeMode()

```
virtual eEdgeMode IDOMPathNode::getEdgeMode ( ) const  [pure virtual]
```

Retrieves render options edge mode of the path.

Render options edge mode controls how the edges of the path are painted. The only valid value for edge mode is Aliased. Omitting this attribute causes the edges to be rendered in the consumer's default manner

Returns

eEdgeMode The edge mode of the path.

getFill()

```
virtual IDOMBrushPtr IDOMPathNode::getFill ( ) const  [pure virtual]
```

Retrieves the fill brush for the path.

Returns

IDOMBrushPtr The fill brush, or NULL if not present.

getIsDashed()

```
virtual bool IDOMPathNode::getIsDashed ( )  [pure virtual]
```

Checks to see if the path is dashed. Even if a dash array is provided it may still effectively represent a plain un-dashed line. In this case this member will return false.

Returns

bool True if the stroke is affected by dashing, false otherwise.

getLanguage()

```
virtual EDLString IDOMPathNode::getLanguage ( ) const  [pure virtual]
```

Retrieves the default language of the path node and any of its children.

English is defined as en_GB and American English as en_US. There is no default setting. If the language is not known it is set to und (undetermined). For further information see http://www.w3.org/↩International/articles/language-tags/.

The language is specified according to RFC 3066.

Returns

EDLString The default language

getNavigateLink()

```
virtual IDOMTargetPtr IDOMPathNode::getNavigateLink ( ) const  [pure virtual]
```

Retrieves the target of a hyperlink.

Returns

IDOMTargetPtr the target, or NULL if there is no target.

getOpacity()

```
virtual float IDOMPathNode::getOpacity ( ) const  [pure virtual]
```

Retrieves the opacity value of the path. The opacity value defines the uniform transparency of the path. This is a number between 0 (fully transparent) and 1 (fully opaque). Default value 1.0.

Returns

float The opacity value

getOpacityMask()

```
virtual IDOMBrushPtr IDOMPathNode::getOpacityMask ( ) const  [pure virtual]
```

Retrieves the opacity mask for the path.

The opacity mask specifies a mask of alpha values that is applied to the path in the same fashion as the simple opacity setting, but allowing different alpha values for different area of the canvas. With an opacity mask, you can combine an object with one or more other objects to define transparent areas of the shape.

Opacity masks use the values of one object or image to define the transparent areas of another. The objects that define the transparency can be any type of visual element - solid objects, strokes, gradients, raster images, text or combinations of all of the above.

Returns

IDOMBrushPtr The opacity mask, or NULL if not present.

getPathData()

```
virtual IDOMPathGeometryPtr IDOMPathNode::getPathData ( ) const  [pure virtual]
```

Retrieves a smart pointer to the path geometry node.

Returns

IDOMPathGeometryPtr The path geometry or NULL if no path data is present.

getRenderTransform()

```
virtual const FMatrix & IDOMPathNode::getRenderTransform ( ) const  [pure virtual]
```

Retrieves the render transform matrix. The render transform matrix establishes a new coordinate frame for all attributes of the path and for all child elements of the path, such as the path geometry.

Returns

FMatrix Render transform matrix.

getShape()

```
virtual IDOMShapePtr IDOMPathNode::getShape (
            const FMatrix & transform,
            float resolution,
            IEDLClassFactory * pFactory )  [pure virtual]
```

Get the scan-converted shape of this path.

Parameters

transform	The transform that should be applied before scan conversion.
resolution	The resolution that should be used for scan conversion.
pFactory	The factory to use.

Returns

IDOMShapePtr The shape.

getShouldZeroWidthLinesBeVisible()

```
virtual bool IDOMPathNode::getShouldZeroWidthLinesBeVisible ( ) const  [pure virtual]
```

Should zero width strokes be visible as cosmetic lines? Zero width lines in XPS files are not visible, but in PDF and PostScript, such lines are rendered as a single pixel cosmetic line. This routine returns true if a zero width stroke for this path should be rendered.

Returns

bool True if a zero width stroke in this path should be visible as a cosmetic line.

getSnapsToDevicePixels()

```
virtual bool IDOMPathNode::getSnapsToDevicePixels ( ) const  [pure virtual]
```

Retrieves the snapsToDevicePixels setting.

The ignorable attribute snapsToDevicePixels can be set to allow consumers or viewers that perform anti-aliasing to "snap" any path control points that are situated on the path bounding box to whole device pixels.

Returns

bool True if control points snap to the nearest device pixels.

getStroke()

```
virtual IDOMBrushPtr IDOMPathNode::getStroke ( ) const  [pure virtual]
```

Retrieves the stroke brush for the path.

Returns

IDOMBrushPtr The stroke brush, or NULL if not present.

getStrokeDashLineCap()

```
virtual eStrokeLineCap IDOMPathNode::getStrokeDashLineCap ( ) const  [pure virtual]
```

Gets the stroke dash line cap.

See also

eStrokeLineCap

Returns

eStrokeLineCap Stroke dash line cap (Flat default)

getStrokeDashOffset()

```
virtual double IDOMPathNode::getStrokeDashOffset ( ) const  [pure virtual]
```

Retrieves the stroke dash offset value. This adjusts the start point for repeating the dash array pattern. If this value is omitted, the dash array aligns with the origin of the stroke. Values are specified as multiples of the stroke thickness.

Returns

double Stroke dash offset (0.0 default)

getStrokeDashPattern()

```
virtual const CEDLVector< double > & IDOMPathNode::getStrokeDashPattern ( ) const  [pure virtual]
```

Retrieves the stroke dash array.

The stroke dash array specifies the length of dashes and gaps of the outline stroke. The dash and gap lengths are specified as multiples of the stroke thickness and are stored in an array containing an even number of non-negative values.

The first argument in the array is the width of the first dash. The second is the width of the gap following the first dash. The third argument is the second dash width, followed by another gap width, and so on. If you specify an odd number of elements, the elements are repeated to produce an even number.

When a stroke is drawn, the dashes and gaps specified by these values are repeated to cover the length of the stroke. If this attribute is omitted, the stroke is drawn solid, without any gaps.

Returns

CEDLVector<**double**> The stroke dash array

getStrokeDashsCount()

```
virtual uint32 IDOMPathNode::getStrokeDashsCount ( )  [pure virtual]
```

Retrieves the length of the stroke dash array.

Returns

uint32 The number of entries in the stroke dash array.

getStrokeEndLineCap()

```
virtual eStrokeLineCap IDOMPathNode::getStrokeEndLineCap ( ) const  [pure virtual]
```

Retrieves the line cap type for the end of the stroke.

See also

eStrokeLineCap

Returns

eStrokeLineCap The type of line cap to use at the end of the stroke.

getStrokeLineJoin()

```
virtual eStrokeLineJoin IDOMPathNode::getStrokeLineJoin ( ) const  [pure virtual]
```

Retrieves the stroke line join value. The stroke line join specifies how a stroke is drawn at a corner of a path. Valid values are specified by eStrokeLineJoin. If mitered joins are selected, the value of StrokeMiterLimit is used in drawing the stroke.

See also

eStrokeLineJoin

Returns

eStrokeLineJoin Stroke line join (Miter default)

getStrokeMiterLimit()

```
virtual double IDOMPathNode::getStrokeMiterLimit ( ) const  [pure virtual]
```

Retrieves the stroke miter limit. The stroke miter limit is the ratio between the maximum miter length and half of the stroke thickness. This value must be equal to or greater than 1.0. The value is significant only if the StrokeLineJoin attribute specifies mitered joins.

Returns

double Stroke miter limit (10.0 default).

getStrokeMiterLimitTreatment()

```
virtual eStrokeMiterLimitTreatment IDOMPathNode::getStrokeMiterLimitTreatment ( ) const  [pure
virtual]
```

Retrieves the miter limit treatment for this path. The stroke miter treatment specifies how miters extending beyond the limit should be treated.

Returns

eStrokeMiterLimitTreatment Stroke miter treatment. (ClipLongMiters default).

getStrokeStartLineCap()

```
virtual eStrokeLineCap IDOMPathNode::getStrokeStartLineCap ( ) const  [pure virtual]
```

Retrieves the line cap type for the start of the stroke.

See also

eStrokeLineCap

Returns

eStrokeLineCap The type of line cap to use at the start of the stroke.

getStrokeThickness()

```
virtual double IDOMPathNode::getStrokeThickness ( ) const  [pure virtual]
```

Retrieves the stroke thickness. The stroke thickness specifies the thickness of a stroke, in units of the effective coordinate space including the path's render transform. The stroke is drawn on top of the boundary of the geometry specified by the path geometry information. Half of the stroke thickness extends outside of the geometry and the other half extends inside of the geometry.

Returns

double Stroke thickness (1.0 default)

setAutomationPropertiesHelpText()

```
virtual void IDOMPathNode::setAutomationPropertiesHelpText (
            const EDLString & helptext )  [pure virtual]
```

Sets the automation properties help text of the path.

Automation properties help text is a detailed description of the path's content, used for accessibility purposes, particularly if the path is filled with an image brush, or a set of vector graphics and text elements intended to comprise a single vector graphic.

Parameters

helptext	Automation properties help text

setAutomationPropertiesName()

```
virtual void IDOMPathNode::setAutomationPropertiesName (
            const EDLString & propname )  [pure virtual]
```

Sets the automation properties name of the path.

Automation properties help text is a detailed description of the path's content, used for accessibility purposes, particularly if the path is filled with an image brush, or a set of vector graphics and text elements intended to comprise a single vector graphic.

Parameters

propname	The new automation properties name.

setBlendMode()

```
virtual void IDOMPathNode::setBlendMode (
            eBlendMode blendMode )  [pure virtual]
```

Set the blend mode to be used for rendering this path. Note: modes other than Normal are not directly representable in XPS.

Parameters

blendMode	The desired blend mode.

setClip()

```
virtual void IDOMPathNode::setClip (
            const IDOMPathGeometryPtr & ptrClip )  [pure virtual]
```

Sets the clip geometry.

The clip geometry node specifies a clipping region which describes the geometric area to be preserved. The remainder is not rendered.

Parameters

ptrClip	Smart pointer to the new clip geometry node, or NULL to clear

setEdgeMode()

```
virtual void IDOMPathNode::setEdgeMode (
            eEdgeMode em )  [pure virtual]
```

Sets render the options edge mode of the path.

The EdgeMode property controls how the edges of the path are rendered. The only valid value is Aliased. Omitting this attribute causes the edges to be rendered in the consumers default manner. The EdgeMode property can be set in a path to instruct anti-aliasing consumers to render the path without performing antialiasing.

Parameters

em	The new edge mode for the canvas.

setFill()

```
virtual void IDOMPathNode::setFill (
            const IDOMBrushPtr & ptrFill )  [pure virtual]
```

Sets the fill brush for the path.

Parameters

ptrFill	Smart pointer to the new fill brush, or NULL to clear.

setLanguage()

```
virtual void IDOMPathNode::setLanguage (
            const EDLString & lang )  [pure virtual]
```

Sets default language of the <Path> element and any of its children.

English is defined as en_GB and American English as en_US. There is no default setting. If the language is not known it is set to und (undetermined). For further information see http://www.w3.org/International/articles/language-tags/.

The language is specified according to RFC 3066.

Parameters

lang	Default language

setNavigateLink()

```
virtual void IDOMPathNode::setNavigateLink (
            const IDOMTargetPtr & target )  [pure virtual]
```

Sets the target of a hyperlink.

Parameters

target	The target to associate with the node, or NULL to clear.

setOpacity()

```
virtual void IDOMPathNode::setOpacity (
            float opc )  [pure virtual]
```

Sets the opacity value of the path. The opacity value defines the uniform transparency of the path. This is a number between 0 (fully transparent) and 1 (fully opaque).

Parameters

opc	The new opacity value.

setOpacityMask()

```
virtual void IDOMPathNode::setOpacityMask (
            const IDOMBrushPtr & ptrOpacityMask )  [pure virtual]
```

Sets the opacity mask for the path.

The opacity mask specifies a mask of alpha values that is applied to the path in the same fashion as the simple opacity setting, but allowing different alpha values for different area of the canvas. With an opacity mask, you can combine an object with one or more other objects to define transparent areas of the shape.

Opacity masks use the values of one object or image to define the transparent areas of another. The objects that define the transparency can be any type of visual element - solid objects, strokes, gradients, raster images, text or combinations of all of the above.

Parameters

ptrOpacityMask	Smart pointer to brush to set, or NULL to clear.

setPathData()

```
virtual void IDOMPathNode::setPathData (
            const IDOMPathGeometryPtr & ptrPathData )  [pure virtual]
```

Sets the path geometry node.

Parameters

ptrPathData	Smart pointer to the new path geometry node.

setRenderTransform()

```
virtual void IDOMPathNode::setRenderTransform (
            const FMatrix & matrix )  [pure virtual]
```

Sets the render transform matrix. The render transform matrix establishes a new coordinate frame for all attributes of the path and for all child elements of the path, such as the path geometry.

Parameters

matrix	The new render transform matrix.

setShouldZeroWidthLinesBeVisible()

```
virtual void IDOMPathNode::setShouldZeroWidthLinesBeVisible (
            bool visible )  [pure virtual]
```

Sets whether or not zero width strokes in this path should be rendered visibly as a cosmetic line.

Parameters

visible	Whether or not zero width strokes should be rendered as a cosmetic line.

setSnapsToDevicePixels()

```
virtual void IDOMPathNode::setSnapsToDevicePixels (
            bool s2dp )  [pure virtual]
```

Sets snapsToDevicePixels.

The ignorable attribute snapsToDevicePixels can be set to allow consumers or viewers that perform anti-aliasing to "snap" any path control points that are situated on the path bounding box to whole device pixels.

Parameters

s2dp	New value of snapsToDevicePixels.

setStroke()

```
virtual void IDOMPathNode::setStroke (
            const IDOMBrushPtr & ptrStroke )  [pure virtual]
```

Sets the stroke brush for the path.

Parameters

ptrStroke	Smart pointer to the new stroke brush, or NULL to clear.

setStrokeDashLineCap()

```
virtual void IDOMPathNode::setStrokeDashLineCap (
            eStrokeLineCap slc )  [pure virtual]
```

Sets the stroke dash line cap.

See also

eStrokeLineCap

Parameters

slc	The stroke dash line cap (Flat default)

setStrokeDashOffset()

```
virtual void IDOMPathNode::setStrokeDashOffset (
            double sdo )  [pure virtual]
```

Sets the stroke dash offset value. This adjusts the start point for repeating the dash array pattern. If this value is omitted, the dash array aligns with the origin of the stroke. Values are specified as multiples of the stroke thickness.

Parameters

sdo	Stroke dash offset (0.0 default)

setStrokeDashPattern()

```
virtual void IDOMPathNode::setStrokeDashPattern (
            const CEDLVector< double > & pattern )  [pure virtual]
```

Set the stroke dash array.

The stroke dash array specifies the length of dashes and gaps of the outline stroke. The dash and gap lengths are specified as multiples of the stroke thickness and are stored in an array containing an even number of non-negative values.

The first argument in the array is the width of the first dash. The second is the width of the gap following the first dash. The third argument is the second dash width, followed by another gap width, and so on. If you specify an odd number of elements, the elements are repeated to produce an even number.

When a stroke is drawn, the dashes and gaps specified by these values are repeated to cover the length of the stroke. If this attribute is omitted, the stroke is drawn solid, without any gaps.

Parameters

pattern	The desired dash pattern.

setStrokeEndLineCap()

```
virtual void IDOMPathNode::setStrokeEndLineCap (
            eStrokeLineCap slc )  [pure virtual]
```

Sets the line cap type for the end of the stroke.

See also

eStrokeLineCap

Parameters

slc	The stroke end line cap (Flat default)

setStrokeLineJoin()

```
virtual void IDOMPathNode::setStrokeLineJoin (
            eStrokeLineJoin slj )  [pure virtual]
```

Sets the stroke line join value. The stroke line join specifies how a stroke is drawn at a corner of a path. Valid values are specified by eStrokeLineJoin. If mitered joins are selected, the value of StrokeMiterLimit is used in drawing the stroke.

See also

eStrokeLineJoin

Parameters

slj	The stroke line join (Miter default)

setStrokeMiterLimit()

```
virtual void IDOMPathNode::setStrokeMiterLimit (
            double sml )  [pure virtual]
```

Sets the stroke miter limit. The stroke miter limit is the ratio between the maximum miter length and half of the stroke thickness. This value must be equal to or greater than 1.0. The value is significant only if the StrokeLineJoin attribute specifies mitered joins.

Parameters

sml	Stroke miter limit (10.0 default).

setStrokeMiterLimitTreatment()

```
virtual void IDOMPathNode::setStrokeMiterLimitTreatment (
            eStrokeMiterLimitTreatment treatment )  [pure virtual]
```

Sets the stroke miter limit treatment. The stroke miter treatment specifies how miters extending beyond the limit should be treated.

Parameters

treatment	The desired stroke miter treatment.

setStrokeStartLineCap()

```
virtual void IDOMPathNode::setStrokeStartLineCap (
            eStrokeLineCap slc )  [pure virtual]
```

Sets the line cap type for the start of the stroke.

See also

eStrokeLineCap

Parameters

slc	The stroke start line cap (Flat default)

setStrokeThickness()

```
virtual void IDOMPathNode::setStrokeThickness (
            double st )  [pure virtual]
```

Sets stroke thickness The stroke thickness specifies the thickness of a stroke, in units of the effective coordinate space including the path's render transform. The stroke is drawn on top of the boundary of the geometry specified by the path geometry information. Half of the stroke thickness extends outside of the geometry and the other half extends inside of the geometry.

Parameters

st	Stroke thickness (1.0 default)

split()

```
virtual IDOMNodePtr IDOMPathNode::split (
            IEDLClassFactory * pFactory )  [pure virtual]
```

If the path represents both a fill and a stroke, separate the fill and stroke into separate paths. Throws an IEDLError on failure.

Parameters

pFactory	The factory to use.

Returns

IDOMNodePtr Either an IDOMGroup, IDOMTransparencyGroup or canvas representing the split results, or the path itself if no splitting is required.

The documentation for this class was generated from the following file:

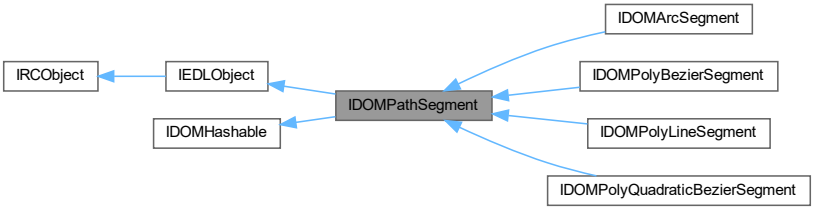
- idompath.h

7.249 IDOMPathSegment Class Reference

Interface to path segment element. The path segment is the smallest unit in a path geometry.

`#include <idompathgeometry.h>`

Inheritance diagram for IDOMPathSegment:



Public Member Functions

- virtual bool getIsStroked () const =0

  Retrieves the value for IsStroked. IsStroked specifies whether the stroke for this segment of the path is drawn. Can be true or false.
- virtual void setIsStroked (bool isStroked)=0

  Sets the value of IsStroked. IsStroked specifies whether the stroke for this segment of the path is drawn. Can be true or false. An exception is thrown if this segment is immutable.
- virtual FRect getBounds (FPoint &startPoint) const =0

  Gets the conservative bounding box of the segment given the start point.
- virtual const FPoint & getEndPoint () const =0

  Gets the end point of the segment.
- virtual bool getIsImmutable () const =0

  Determine if the segment is immutable (non-editable).
- virtual void **setImmutable** ()=0

  Force the segment to be flagged immutable.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.249.1 Detailed Description

Interface to path segment element. The path segment is the smallest unit in a path geometry.

See also

IDOMPathNode

IDOMPathGeometry

IDOMPathFigure

Segments may be lines or curves. One or more segments are combined into a PathFigure definition. A PathFigure is a single shape comprised of continuous segments. One or more PathFigures collectively define an entire path geometry.

Instances and subclasses of this type use exceptions of IEDLError for error handling.

7.249.2 Member Function Documentation

getBounds()

```
virtual FRect IDOMPathSegment::getBounds (
            FPoint & startPoint ) const  [pure virtual]
```

Gets the conservative bounding box of the segment given the start point.

Parameters

startPoint	The start point of the segment.

Returns

FRect The conservative bounding box of the segment.

getEndPoint()

```
virtual const FPoint & IDOMPathSegment::getEndPoint ( ) const  [pure virtual]
```

Gets the end point of the segment.

Returns

FPoint The end point.

getIsImmutable()

```
virtual bool IDOMPathSegment::getIsImmutable ( ) const  [pure virtual]
```

Determine if the segment is immutable (non-editable).

Returns

bool True if the segment may not be edited.

getIsStroked()

```
virtual bool IDOMPathSegment::getIsStroked ( ) const  [pure virtual]
```

Retrieves the value for IsStroked. IsStroked specifies whether the stroke for this segment of the path is drawn. Can be true or false.

Returns

bool Returns true if this line segment is stroked, false if it is not.

setIsStroked()

```
virtual void IDOMPathSegment::setIsStroked (
            bool isStroked )  [pure virtual]
```

Sets the value of IsStroked. IsStroked specifies whether the stroke for this segment of the path is drawn. Can be true or false. An exception is thrown if this segment is immutable.

Parameters

isStroked	New value of IsStroked.

The documentation for this class was generated from the following file:

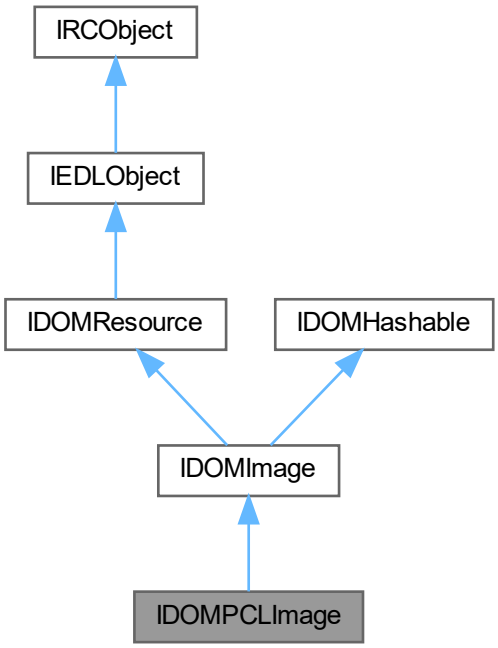
- idompathgeometry.h

7.250 IDOMPCLImage Class Reference

Interface to a class representing an image extracted from a PCLXL file.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMPCLImage:



Classes

- class Data

  Initialization data.

Public Types

- enum eCompressMode

  The image compression mode.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves class id of IDOMCLImage.

Additional Inherited Members

Public Member Functions inherited from IDOMImage

- virtual IImageDecoderPtr createImageDecoder (IEDLClassFactory *factory, const IDOMImagePropertiesPtr &imageProperties)=0

  Creates a properly initialized image decoder object that reads from an inputstream that is specific to that image format.
- virtual IImageFramePtr getImageFrame (IEDLClassFactory *factory)

  Fetch the image frame; convenience.
- virtual IImageEncoderPtr createImageEncoder (const ISessionPtr &session, const IOutputStreamPtr &imageDest, const IDOMImagePropertiesPtr &imageProperties)=0

  Creates a properly initialized image encoder object that writes to an outputstream that is specific to that image format.
- virtual IDOMImagePropertiesPtr getImageProperties ()=0

  Returns an object that stores the properties for this image object. The properties can then be inspected (or more added) by clients that need to manipulate the image resource.
- virtual eDOMImageType getImageType ()=0

  Retrieves the image type.
- virtual bool getIsRendered ()=0

  Determine if the image is as a result of rendering. This is indicated if the image type is eDITRendered or if the image explicitly notes this is the case (such as for IDOMPDFImage).
- virtual IDOMImagePtr getImageWithSubstitutedColorSpace (IEDLClassFactory *factory, const IDOMColor↩SpacePtr &colorSpace)

  Obtain an image that is the same as this image, but with the colorspace substituted for another.

Public Member Functions inherited from IDOMResource

- virtual IInputStreamPtr getStream () const =0

  Retrieves the resource stream.
- virtual void setStream (const IInputStreamPtr &stream)=0

  Sets the resource stream for the node.
- virtual uint64 getStreamLength () const =0

  Retrieves the stream length, if it is available.
- virtual const EDLSysString & getUri () const =0

  Retrieves the resource URI.
- virtual void setUri (const EDLSysString &uri)=0

  Sets the resource URI.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.250.1 Detailed Description

Interface to a class representing an image extracted from a PCLXL file.

7.250.2 Member Function Documentation

classID()

```
static const CClassID & IDOMPCLImage::classID ( )  [inline], [static]
```

Retrieves class id of IDOMCLImage.

Returns

CClassID Class id of the element

The documentation for this class was generated from the following file:

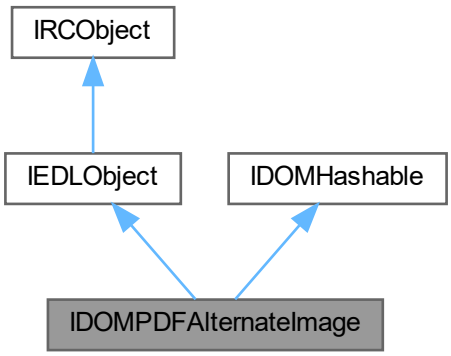
- idomimageresource.h

7.251 IDOMPDFAlternateImage Class Reference

An encapsulation of a PDF alternate image, which may be referenced from an image or mask brush.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMPDFAlternateImage:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual IDOMImageBrushPtr getImageBrush () const =0

  Get the image brush for this alternate image.
- virtual bool getDefaultForPrinting () const =0

  Get if the alternate image is marked as default for printing.
- virtual JawsMako::IOptionalContentDetailsPtr getOptionalContentDetails () const =0

  Get the optional content properties, if present.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static EDL_API IDOMPDFAlternateImagePtr create (IEDLClassFactory *pFactory, const IDOMImageBrush↩ Ptr &alternateBrush, bool defaultForPrinting=false, const JawsMako::IOptionalContentDetailsPtr &optional↩ ContentDetails=JawsMako::IOptionalContentDetailsPtr())

  Creator for an alternate image.
  Although regular, soft mask and masked images are allowed, there are several restrictions on the brushes that can be used for alternate images in order to ensure correct operation.
- static const CClassID & classID ()

  Retrieves class id of IDOMPDFAlternateImage.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.251.1 Detailed Description

An encapsulation of a PDF alternate image, which may be referenced from an image or mask brush.

7.251.2 Member Function Documentation

classID()

```
static const CClassID & IDOMPDFAlternateImage::classID ( )  [inline], [static]
```

Retrieves class id of IDOMPDFAlternateImage.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMPDFAlternateImagePtr IDOMPDFAlternateImage::create (
            IEDLClassFactory * pFactory,
            const IDOMImageBrushPtr & alternateBrush,
            bool defaultForPrinting = false,
            const JawsMako::IOptionalContentDetailsPtr & optionalContentDetails = JawsMako:↩
:IOptionalContentDetailsPtr() )  [static]
```

Creator for an alternate image.
Although regular, soft mask and masked images are allowed, there are several restrictions on the brushes that can be used for alternate images in order to ensure correct operation.

- The render transform of the alternate brush must be identity.
- The viewport of the alternate brush must match the viewport of the host brush. Note - these two restrictions are in place as the alternate image must be the same final dimensions as the host brush.
- The viewbox of the alternate brush must select the entire image.
- If the alternate brush is a masked brush using an image sub brush, the sub brush is subject to all the restrictions above.
- Alternate images are ignored if the host image brush has tiling enabled
- Optional content and alternate image information must not be present in the alternate brush.

Parameters

pFactory	The class factory to use
alternateBrush	The alternate image brush itself. It must not have any alternate images of its own
defaultForPrinting	Set true if this image should be the default image when printing
optionalContentDetails	The optional content details to use, or NULL if not optional

Returns

IDOMPDFAlternateImagePtr The new alternate image

getDefaultForPrinting()

```
virtual bool IDOMPDFAlternateImage::getDefaultForPrinting ( ) const  [pure virtual]
```

Get if the alternate image is marked as default for printing.

Returns

bool True if the image is default for printing, false otherwise

getImageBrush()

```
virtual IDOMImageBrushPtr IDOMPDFAlternateImage::getImageBrush ( ) const  [pure virtual]
```

Get the image brush for this alternate image.

Returns

IDOMImageBrushPtr The image, or NULL on failure

getOptionalContentDetails()

```
virtual JawsMako::IOptionalContentDetailsPtr IDOMPDFAlternateImage::getOptionalContentDetails
( ) const  [pure virtual]
```

Get the optional content properties, if present.

Returns

JawsMako::IOptionalContentDetailsPtr The optional content defails, or NULL if not optional

The documentation for this class was generated from the following file:

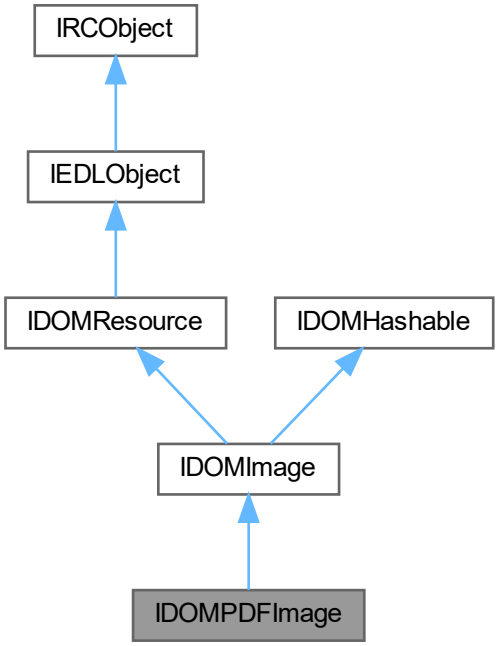
- idomimageresource.h

7.252 IDOMPDFImage Class Reference

Interface to a class representing an image extracted from a PDF file. Intended to be only used with the JawsMako APIs.

```
#include <idomimageresource.h>
```

Inheritance diagram for IDOMPDFImage:



Classes

- class CCITTFaxParams

  Class to hold filter parameters for CCITTFax-compressed image data. Please see the PDF specification for the meaning of these parameters.
- class Data

  Initialization data.
- class DCTParams

  Class to hold filter parameters for DCT-compressed image data. Please see the PDF specification for the meaning of these parameters.
- class FlateLZWParams

  Class to hold filter parameters for Flate or LZW-compressed image data. Please see the PDF specification for the meaning of these parameters.
- class IDecodeParams

  Abstract interface for per-image decoding filter parameters.
- class JBIG2Params

  Class to hold filter parameters for JBIG2-compressed image data. Please see the PDF specification for the meaning of these parameters.

Public Member Functions

- virtual CEDLVector< IDecodeParamsPtr > getDecodeParameters () const =0

  Retrieves the decode parameters. May be NULL.
- virtual IDecodeParamsPtr getDecodeParametersAtIndex (uint32 index) const

  Retrieves the decode parameters from the list with the given index. May be NULL.
- virtual CEDLVector< eDOMImageType > getImageTypes () const =0

  Retrieves the array of the image type.
- virtual CEDLVector< float > getDecode () const =0

  Retrieves the decode array used to interpret the color samples. An empty vector will be returned if there is no decode.
- virtual uint8 getBitsPerComponent () const =0

  Retrieves the bits per component of the source data. For JPEG2000, this information is available only in the image stream and this function will return 0. In this case please use the `IImageStream` interface.
- virtual CEDLVector< uint16 > getColorKey () const =0

  Retrieves the color key for mask generation. An empty vector will be returned if there is no color key.
- virtual IDOMColorSpacePtr getColorSpace () const =0

  Retrieves the color space to be used with this image. For JPEG2000, this may be NULL if the colorspace needs to be determined from the image data. In this case please use the IImageStream interface.
- virtual eImageAlpha getAlphaDetails () const =0

  Returns if the image has alpha, and if the colour samples are premultiplied.

Public Member Functions inherited from IDOMImage

- virtual IImageDecoderPtr createImageDecoder (IEDLClassFactory *factory, const IDOMImagePropertiesPtr &imageProperties)=0

  Creates a properly initialized image decoder object that reads from an inputstream that is specific to that image format.
- virtual IImageFramePtr getImageFrame (IEDLClassFactory *factory)

  Fetch the image frame; convenience.
- virtual IImageEncoderPtr createImageEncoder (const ISessionPtr &session, const IOutputStreamPtr &imageDest, const IDOMImagePropertiesPtr &imageProperties)=0

  Creates a properly initialized image encoder object that writes to an outputstream that is specific to that image format.
- virtual IDOMImagePropertiesPtr getImageProperties ()=0

Returns an object that stores the properties for this image object. The properties can then be inspected (or more added) by clients that need to manipulate the image resource.

- virtual eDOMImageType getImageType ()=0

  Retrieves the image type.
- virtual bool getIsRendered ()=0

  Determine if the image is as a result of rendering. This is indicated if the image type is eDITRendered or if the image explicitly notes this is the case (such as for IDOMPDFImage).
- virtual IDOMImagePtr getImageWithSubstitutedColorSpace (IEDLClassFactory *factory, const IDOMColor↩ SpacePtr &colorSpace)

  Obtain an image that is the same as this image, but with the colorspace substituted for another.

Public Member Functions inherited from IDOMResource

- virtual IInputStreamPtr getStream () const =0

  Retrieves the resource stream.
- virtual void setStream (const IInputStreamPtr &stream)=0

  Sets the resource stream for the node.
- virtual uint64 getStreamLength () const =0

  Retrieves the stream length, if it is available.
- virtual const EDLSysString & getUri () const =0

  Retrieves the resource URI.
- virtual void setUri (const EDLSysString &uri)=0

  Sets the resource URI.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static const CClassID & classID ()
  Retrieves class id of IDOMPDFImage.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()
  Virtual destructor.

7.252.1 Detailed Description

Interface to a class representing an image extracted from a PDF file. Intended to be only used with the JawsMako APIs.

7.252.2 Member Function Documentation

classID()

```
static const CClassID & IDOMPDFImage::classID ( )  [inline], [static]
```

Retrieves class id of IDOMPDFImage.

Returns

CClassID Class id of the element

getAlphaDetails()

```
virtual eImageAlpha IDOMPDFImage::getAlphaDetails ( ) const  [pure virtual]
```

Returns if the image has alpha, and if the colour samples are premultiplied.

Returns

eImageAlpha The image alpha details

getBitsPerComponent()

```
virtual uint8 IDOMPDFImage::getBitsPerComponent ( ) const  [pure virtual]
```

Retrieves the bits per component of the source data. For JPEG2000, this information is available only in the image stream and this function will return 0. In this case please use the `IImageStream` interface.

Returns

uint32 The number of bits per component

getColorKey()

```
virtual CEDLVector< uint16 > IDOMPDFImage::getColorKey ( ) const  [pure virtual]
```

Retrieves the color key for mask generation. An empty vector will be returned if there is no color key.

Returns

CEDLVector<**uint16**> The color key

getColorSpace()

```
virtual IDOMColorSpacePtr IDOMPDFImage::getColorSpace ( ) const  [pure virtual]
```

Retrieves the color space to be used with this image. For JPEG2000, this may be NULL if the colorspace needs to be determined from the image data. In this case please use the IImageStream interface.

Returns

IDOMColorSpacePtr The color space

getDecode()

```
virtual CEDLVector< float > IDOMPDFImage::getDecode ( ) const  [pure virtual]
```

Retrieves the decode array used to interpret the color samples. An empty vector will be returned if there is no decode.

Returns

CEDLVector<**float**> The decode array

getDecodeParameters()

```
virtual CEDLVector< IDecodeParamsPtr > IDOMPDFImage::getDecodeParameters ( ) const  [pure
virtual]
```

Retrieves the decode parameters. May be NULL.

Returns

CEDLVector<**IDecodeParamsPtr**> The array of the decode parameters

getDecodeParametersAtIndex()

```
virtual IDecodeParamsPtr IDOMPDFImage::getDecodeParametersAtIndex (
            uint32 index ) const  [inline], [virtual]
```

Retrieves the decode parameters from the list with the given index. May be NULL.

Parameters

index	Index of the desired decode parameters in the array to return (beginning at 0).

Returns

IDecodeParamsPtr The decode parameters

getImageTypes()

```
virtual CEDLVector< eDOMImageType > IDOMPDFImage::getImageTypes ( ) const  [pure virtual]
```

Retrieves the array of the image type.

Returns

CEDLVector<**eDOMImageType**> The array of the image type.

The documentation for this class was generated from the following file:

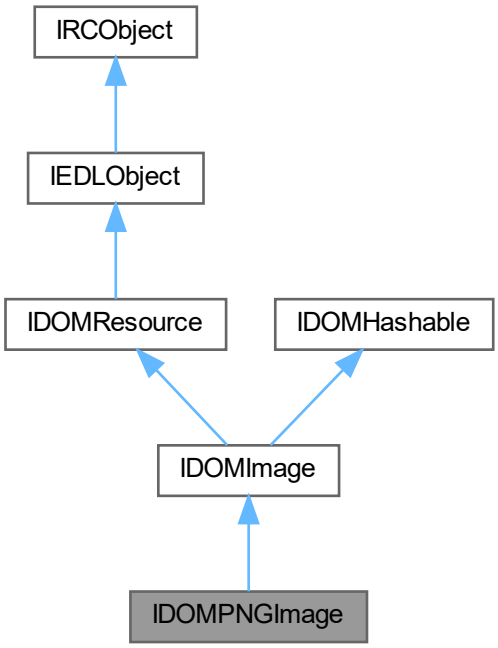
- idomimageresource.h

7.253 IDOMPNGImage Class Reference

Interface to a class representing a PNG (.png) image.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMPNGImage:



Static Public Member Functions

- static EDL_API IDOMPNGImagePtr create (IEDLClassFactory *pFactory, const IInputStreamPtr &stream)

  Create a PNG Image resource with the given PNG stream.
- static EDL_API IDOMImagePtr createWriterAndImage (const ISessionPtr &session, IImageFrameWriterPtr &frame, const IDOMColorSpacePtr &colorSpace, uint32 width, uint32 height, uint8 bitsPerComponent=8, double xResolution=96.0, double yResolution=96.0, eImageExtraChannelType extraChannel=eIECNone, const IInputStreamPtr &inStream=IInputStreamPtr(), const IOutputStreamPtr &outStream=IOutputStream↩Ptr())

  Create an IDOMPNGImage and frame that can be used to populate same.
- static EDL_API void encode (const ISessionPtr &pSession, const IDOMImagePtr &image, const IOutput↩StreamPtr &stream)

  Encode an image as a PNG stream, returning the stream. This routine may convert the image samples into a form that may be encoded as PNG if required, such as by converting to a supported color space.
- static EDL_API void encode (const ISessionPtr &pSession, const IImageFramePtr &frame, const IOutput↩StreamPtr &stream)

  Encode the contents of an IImageFrame as a PNG stream, returning the stream. This routine may convert the image samples into a form that may be encoded as PNG if required, such as by converting to a supported color space.
- static EDL_API void encode (const ISessionPtr &pSession, const IDOMColorSpacePtr &colorSpace, float dpi, uint8 bpc, void *frameBuffer, uint32 width, uint32 height, int32 stride, bool hasAlpha, const IOutputStreamPtr &stream)

  Encode a frame buffer as a PNG stream, returning the stream. The source image must be compatible with PNG.
- static const CClassID & classID ()

  Retrieves class id of IDOM.

Additional Inherited Members

Public Member Functions inherited from IDOMImage

- virtual IImageDecoderPtr createImageDecoder (IEDLClassFactory *factory, const IDOMImagePropertiesPtr &imageProperties)=0

  Creates a properly initialized image decoder object that reads from an inputstream that is specific to that image format.
- virtual IImageFramePtr getImageFrame (IEDLClassFactory *factory)

  Fetch the image frame; convenience.
- virtual IImageEncoderPtr createImageEncoder (const ISessionPtr &session, const IOutputStreamPtr &imageDest, const IDOMImagePropertiesPtr &imageProperties)=0

  Creates a properly initialized image encoder object that writes to an outputstream that is specific to that image format.
- virtual IDOMImagePropertiesPtr getImageProperties ()=0

  Returns an object that stores the properties for this image object. The properties can then be inspected (or more added) by clients that need to manipulate the image resource.
- virtual eDOMImageType getImageType ()=0

  Retrieves the image type.
- virtual bool getIsRendered ()=0

  Determine if the image is as a result of rendering. This is indicated if the image type is eDITRendered or if the image explicitly notes this is the case (such as for IDOMPDFImage).
- virtual IDOMImagePtr getImageWithSubstitutedColorSpace (IEDLClassFactory *factory, const IDOMColor↩SpacePtr &colorSpace)

  Obtain an image that is the same as this image, but with the colorspace substituted for another.

Public Member Functions inherited from IDOMResource

- virtual IInputStreamPtr getStream () const =0

  Retrieves the resource stream.
- virtual void setStream (const IInputStreamPtr &stream)=0

  Sets the resource stream for the node.
- virtual uint64 getStreamLength () const =0

  Retrieves the stream length, if it is available.
- virtual const EDLSysString & getUri () const =0

  Retrieves the resource URI.
- virtual void setUri (const EDLSysString &uri)=0

  Sets the resource URI.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.253.1 Detailed Description

Interface to a class representing a PNG (.png) image.

7.253.2 Member Function Documentation

classID()

```
static const CClassID & IDOMPNGImage::classID ( )  [inline], [static]
```

Retrieves class id of IDOM.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMPNGImagePtr IDOMPNGImage::create (
            IEDLClassFactory * pFactory,
            const IInputStreamPtr & stream )  [static]
```

Create a PNG Image resource with the given PNG stream.

Parameters

pFactory	The EDL Class factory to use.
stream	The stream containing the PNG image.

Returns

IDOMImagePtr The new image.

createWriterAndImage()

```
static EDL_API IDOMImagePtr IDOMPNGImage::createWriterAndImage (
            const ISessionPtr & session,
            IImageFrameWriterPtr & frame,
            const IDOMColorSpacePtr & colorSpace,
            uint32 width,
            uint32 height,
            uint8 bitsPerComponent = 8,
            double xResolution = 96.0,
            double yResolution = 96.0,
            eImageExtraChannelType extraChannel = eIECNone,
            const IInputStreamPtr & inStream = IInputStreamPtr(),
            const IOutputStreamPtr & outStream = IOutputStreamPtr() )  [static]
```

Create an IDOMPNGImage and frame that can be used to populate same.

Parameters

session	The session to use
frame	On exit, this is populated with a frame ready to receive image data via frame->writeScanLine(). Use frame->flushData() to complete the encoding process.
colorSpace	The color space to use. Must be compatible with the PNG format.
width	The width of the image, in pixels.
height	The height of the image, in pixels.
bitsPerComponent	The bits per component to use. Must be compatible with the PNG format.
xResolution	The x resolution, in pixels-per-inch.
yResolution	The y resolution, in pixels-per-inch.
extraChannel	The type of extra channel, if provided. Must be either eIECAlpha or eIECNone for PNG.
inStream	Optional. The first in a pair of streams used to read and write the raw image data if an external stream is desired. If NULL, a temporary store stream will be created. If non NULL, outStream must also be provided.
outStream	Optional. The second in a pair of streams used to read and write the raw image data if an external stream is desired. If NULL, a temporary store stream will be created. If non NULL, inStream must also be provided.

Returns

IDOMImagePtr The resulting image. Not valid until the frame is flushed.

encode() [1/3]

```
static EDL_API void IDOMPNGImage::encode (
            const ISessionPtr & pSession,
            const IDOMColorSpacePtr & colorSpace,
            float dpi,
            uint8 bpc,
            void * frameBuffer,
            uint32 width,
            uint32 height,
            int32 stride,
            bool hasAlpha,
            const IOutputStreamPtr & stream )  [static]
```

Encode a frame buffer as a PNG stream, returning the stream. The source image must be compatible with PNG.

Parameters

pSession	The relevant EDL session
colorSpace	The color space of the frame buffer
dpi	The image resolution in dots-per-inch
bpc	Bits per component (pixel)
frameBuffer	The frame buffer to be encoded
width	The width of the frame buffer
height	The height of the frame buffer
stride	The offset in bytes in the frameBuffer from one scanline to the next. May be negative.
hasAlpha	Set true if the frame buffer has an alpha channel.
stream	The stream to use to store the image data.

encode() [2/3]

```
static EDL_API void IDOMPNGImage::encode (
            const ISessionPtr & pSession,
            const IDOMImagePtr & image,
            const IOutputStreamPtr & stream )  [static]
```

Encode an image as a PNG stream, returning the stream. This routine may convert the image samples into a form that may be encoded as PNG if required, such as by converting to a supported color space.

Parameters

pSession	The relevant EDL session
image	The image to be encoded
stream	The stream to use to store the image data.

encode() [3/3]

```
static EDL_API void IDOMPNGImage::encode (
            const ISessionPtr & pSession,
            const IImageFramePtr & frame,
            const IOutputStreamPtr & stream )  [static]
```

Encode the contents of an IImageFrame as a PNG stream, returning the stream. This routine may convert the image samples into a form that may be encoded as PNG if required, such as by converting to a supported color space.

Parameters

pSession	The relevant EDL session
frame	The frame providing the source image data
stream	The stream to use to store the image data.

The documentation for this class was generated from the following file:

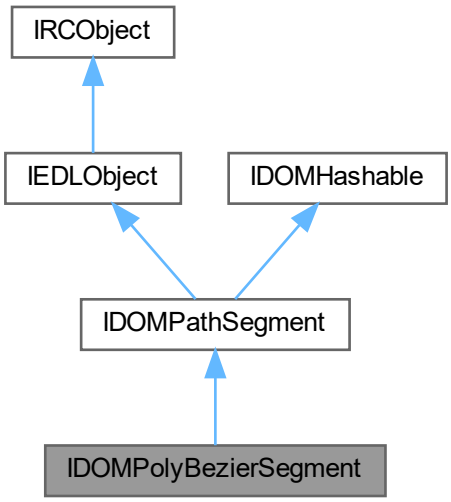
- idomimageresource.h

7.254 IDOMPolyBezierSegment Class Reference

Interface to a path segment node describing a set of cubic Bézier curves.

```
#include <idompathgeometry.h>
```

Inheritance diagram for IDOMPolyBezierSegment:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual const CFPointVect & getPoints () const =0

  Retrieves the points that make up this segment. For beziers, the number of points will be a multiple of three.
- virtual uint32 getPointsCount () const =0

  Retrieves the number of points in the points list. For beziers, the number of points will be a multiple of three.
- virtual void **clearPoints** ()=0

  Clears the points list. An exception is thrown if this segment is immutable.
- virtual void addPoint (const FPoint &point)=0

  Append a point to the list. An exception is thrown if this segment is immutable.

Public Member Functions inherited from IDOMPathSegment

- virtual bool getIsStroked () const =0

  Retrieves the value for IsStroked. IsStroked specifies whether the stroke for this segment of the path is drawn. Can be true or false.
- virtual void setIsStroked (bool isStroked)=0

  Sets the value of IsStroked. IsStroked specifies whether the stroke for this segment of the path is drawn. Can be true or false. An exception is thrown if this segment is immutable.
- virtual FRect getBounds (FPoint &startPoint) const =0

  Gets the conservative bounding box of the segment given the start point.
- virtual const FPoint & getEndPoint () const =0

Gets the end point of the segment.

- virtual bool getIsImmutable () const =0

  Determine if the segment is immutable (non-editable).

- virtual void **setImmutable** ()=0

  Force the segment to be flagged immutable.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.

- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.

- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.

- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.

- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.

- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.

- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves class id of IDOMPolyBezierSegment.

- static EDL_API IDOMPolyBezierSegmentPtr create (IEDLClassFactory *factory, bool isStroked=true, const CFPointVect &points=CFPointVect())

  Simplified creator for a bezier segment.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.254.1 Detailed Description

Interface to a path segment node describing a set of cubic Bézier curves.

```
B&eacute;zier curves are drawn from the previous point in the path figure or the previous B&eacute;zier
segment and terminate at the third point (x3n,y3n) in the Points attribute (where n is the curve being
drawn). The tangents and curvature of each B&eacute;zier curve are controlled by the first two control p
(x3n-2,y3n-2 and x3n-1,y3n-1) in the Points attribute. The Points attribute contains a multiple of three
whitespace delimited pairs of comma delimited x,y values.
```

7.254.2 Member Function Documentation

addPoint()

```
virtual void IDOMPolyBezierSegment::addPoint (
            const FPoint & point )  [pure virtual]
```

Append a point to the list. An exception is thrown if this segment is immutable.

Parameters

point	Point to append.

classID()

```
static const CClassID & IDOMPolyBezierSegment::classID ( )  [inline], [static]
```

Retrieves class id of IDOMPolyBezierSegment.

Returns

CClassID class id of the element.

create()

```
static EDL_API IDOMPolyBezierSegmentPtr IDOMPolyBezierSegment::create (
            IEDLClassFactory * factory,
            bool isStroked = true,
            const CFPointVect & points = CFPointVect() )  [static]
```

Simplified creator for a bezier segment.

Parameters

factory	The factory to use.
isStroked	Should the segment be stroked, if used in a stroking path.
points	The segment points. Must be a multiple of three in length.

Returns

IDOMPolyLineSegmentPtr The new segment.

getPoints()

```
virtual const CFPointVect & IDOMPolyBezierSegment::getPoints ( ) const  [pure virtual]
```

Retrieves the points that make up this segment. For beziers, the number of points will be a multiple of three.

Returns

CFPointVect The points.

getPointsCount()

```
virtual uint32 IDOMPolyBezierSegment::getPointsCount ( ) const  [pure virtual]
```

Retrieves the number of points in the points list. For beziers, the number of points will be a multiple of three.

Returns

uint32 The size of the points list.

The documentation for this class was generated from the following file:

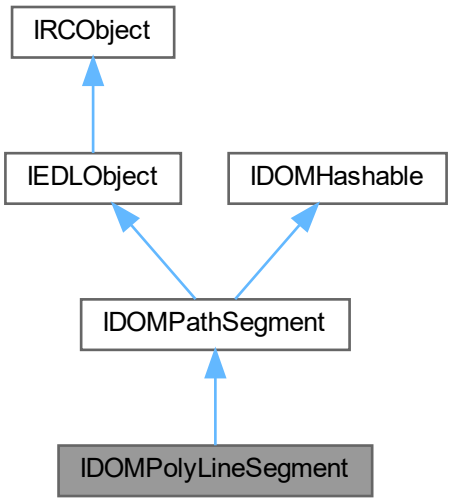
- idompathgeometry.h

7.255 IDOMPolyLineSegment Class Reference

Interface to a polyline segment node. A polyline segment describes a polygonal drawing containing an arbitrary number of individual vertices. The Points attribute defines the vertices.

```
#include <idompathgeometry.h>
```

Inheritance diagram for IDOMPolyLineSegment:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual const CFPointVect & getPoints () const =0

  Retrieves the points that make up this segment.
- virtual uint32 getPointsCount () const =0

  Retrieves the number of points in the segment.
- virtual void **clearPoints** ()=0

  Clears the points list. An exception is thrown if this segment is immutable.
- virtual void addPoint (const FPoint &point)=0

  Appends a point to the list. An exception is thrown if this segment is immutable.

Public Member Functions inherited from IDOMPathSegment

- virtual bool getIsStroked () const =0

  Retrieves the value for IsStroked. IsStroked specifies whether the stroke for this segment of the path is drawn. Can be true or false.
- virtual void setIsStroked (bool isStroked)=0

  Sets the value of IsStroked. IsStroked specifies whether the stroke for this segment of the path is drawn. Can be true or false. An exception is thrown if this segment is immutable.
- virtual FRect getBounds (FPoint &startPoint) const =0

  Gets the conservative bounding box of the segment given the start point.
- virtual const FPoint & getEndPoint () const =0

  Gets the end point of the segment.
- virtual bool getIsImmutable () const =0

  Determine if the segment is immutable (non-editable).
- virtual void **setImmutable** ()=0

  Force the segment to be flagged immutable.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves class id of IDOMPolyLineSegment.
- static EDL_API IDOMPolyLineSegmentPtr create (IEDLClassFactory *factory, bool isStroked=true, const CFPointVect &points=CFPointVect())

  Simplified creator for a line segment.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.255.1 Detailed Description

Interface to a polyline segment node. A polyline segment describes a polygonal drawing containing an arbitrary number of individual vertices. The Points attribute defines the vertices.

7.255.2 Member Function Documentation

addPoint()

```
virtual void IDOMPolyLineSegment::addPoint (
            const FPoint & point )  [pure virtual]
```

Appends a point to the list. An exception is thrown if this segment is immutable.

Parameters

point	Point to append.

classID()

```
static const CClassID & IDOMPolyLineSegment::classID ( )  [inline], [static]
```

Retrieves class id of IDOMPolyLineSegment.

Returns

CClassID class id of the element.

create()

```
static EDL_API IDOMPolyLineSegmentPtr IDOMPolyLineSegment::create (
            IEDLClassFactory * factory,
            bool isStroked = true,
            const CFPointVect & points = CFPointVect() )  [static]
```

Simplified creator for a line segment.

Parameters

factory	The factory to use.
isStroked	Should the segment be stroked, if used in a stroking path.
points	The segment points.

Returns

IDOMPolyLineSegmentPtr The new segment.

getPoints()

```
virtual const CFPointVect & IDOMPolyLineSegment::getPoints ( ) const  [pure virtual]
```

Retrieves the points that make up this segment.

Returns

CFPointVect The points.

getPointsCount()

```
virtual uint32 IDOMPolyLineSegment::getPointsCount ( ) const  [pure virtual]
```

Retrieves the number of points in the segment.

Returns

uint32 The size of the points list.

The documentation for this class was generated from the following file:

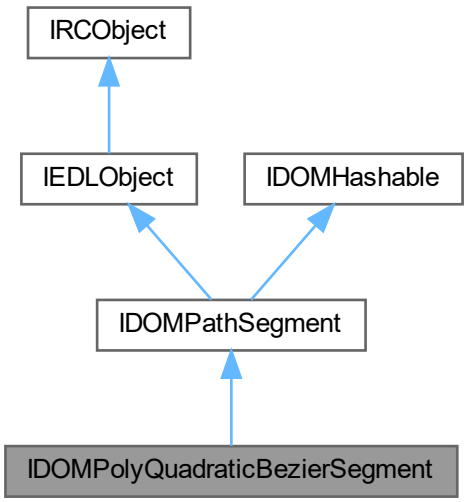
- idompathgeometry.h

7.256 IDOMPolyQuadraticBezierSegment Class Reference

Interface to a polyqualdratic Bézier segment. A polyqualdratic Bézier segment describes a set of quadratic Bézier curves from the starting point defined in the IDOMPathFigure, or from the end point of the previous segment, through a set of vertices, using specified control points. The Points attribute stores an off-curve control point (x2n-1, y2n-1) followed by the end point (x2n, y2n) for each quadratic Bézier curve (where n represents the quadratic Bézier curve).

`#include <idompathgeometry.h>`

Inheritance diagram for IDOMPolyQuadraticBezierSegment:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual const CFPointVect & getPoints () const =0

  Retrieves the points that make up this segment. For quadratic beziers, the number of points will be a multiple of two.
- virtual uint32 getPointsCount () const =0

  Retrieves the number of points in the points list. For quadratic beziers, the number of points will be a multiple of two.
- virtual void **clearPoints** ()=0

  Clears the points list. An exception is thrown if this segment is immutable.
- virtual void addPoint (const FPoint &point)=0

  Append a point to the list. An exception is thrown if this segment is immutable.
- virtual IDOMPolyBezierSegmentPtr convertToCubicBezierSegment (IEDLClassFactory *factory, const FPoint &startPoint) const =0

  Create a segment that represents this curve using a cubic bezier.

Public Member Functions inherited from IDOMPathSegment

- virtual bool getIsStroked () const =0

  Retrieves the value for IsStroked. IsStroked specifies whether the stroke for this segment of the path is drawn. Can be true or false.
- virtual void setIsStroked (bool isStroked)=0

  Sets the value of IsStroked. IsStroked specifies whether the stroke for this segment of the path is drawn. Can be true or false. An exception is thrown if this segment is immutable.
- virtual FRect getBounds (FPoint &startPoint) const =0

  Gets the conservative bounding box of the segment given the start point.
- virtual const FPoint & getEndPoint () const =0

  Gets the end point of the segment.
- virtual bool getIsImmutable () const =0

  Determine if the segment is immutable (non-editable).
- virtual void **setImmutable** ()=0

  Force the segment to be flagged immutable.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves the class id of IDOMPolyQuadraticBezierSegment.
- static EDL_API IDOMPolyQuadraticBezierSegmentPtr create (IEDLClassFactory *factory, bool isStroked=true, const CFPointVect &points=CFPointVect())

  Simplified creator for a quadratic bezier segment.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.256.1 Detailed Description

Interface to a polyqualdratic Bézier segment. A polyqualdratic Bézier segment describes a set of quadratic Bézier curves from the starting point defined in the IDOMPathFigure, or from the end point of the previous segment, through a set of vertices, using specified control points. The Points attribute stores an off-curve control point (x2n-1, y2n-1) followed by the end point (x2n, y2n) for each quadratic Bézier curve (where n represents the quadratic Bézier curve).

7.256.2 Member Function Documentation

addPoint()

```
virtual void IDOMPolyQuadraticBezierSegment::addPoint (
            const FPoint & point )  [pure virtual]
```

Append a point to the list. An exception is thrown if this segment is immutable.

Parameters

point	Point to append.

classID()

```
static const CClassID & IDOMPolyQuadraticBezierSegment::classID ( )  [inline], [static]
```

Retrieves the class id of IDOMPolyQuadraticBezierSegment.

Returns

CClassID. Returns the class id of the element.

convertToCubicBezierSegment()

```
virtual IDOMPolyBezierSegmentPtr IDOMPolyQuadraticBezierSegment::convertToCubicBezierSegment (
            IEDLClassFactory * factory,
            const FPoint & startPoint ) const  [pure virtual]
```

Create a segment that represents this curve using a cubic bezier.

Parameters

factory	Pointer to the class factory to be used.
startPoint	The start point for the curve.

Returns

IDOMPolyBezierSegmentPtr The resulting cubic bezier.

create()

```
static EDL_API IDOMPolyQuadraticBezierSegmentPtr IDOMPolyQuadraticBezierSegment::create (
            IEDLClassFactory * factory,
            bool isStroked = true,
            const CFPointVect & points = CFPointVect() )  [static]
```

Simplified creator for a quadratic bezier segment.

Parameters

factory	The factory to use.
isStroked	Should the segment be stroked, if used in a stroking path.
points	The segment points. Must be a multiple of two in length.

Returns

IDOMPolyLineSegmentPtr The new segment.

getPoints()

```
virtual const CFPointVect & IDOMPolyQuadraticBezierSegment::getPoints ( ) const  [pure virtual]
```

Retrieves the points that make up this segment. For quadratic beziers, the number of points will be a multiple of two.

Returns

CFPointVect The points.

getPointsCount()

```
virtual uint32 IDOMPolyQuadraticBezierSegment::getPointsCount ( ) const  [pure virtual]
```

Retrieves the number of points in the points list. For quadratic beziers, the number of points will be a multiple of two.

Returns

uint32 The size of the points list.

The documentation for this class was generated from the following file:

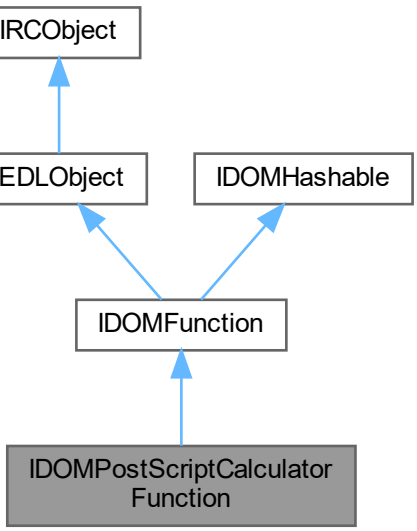
- idompathgeometry.h

7.257 IDOMPostScriptCalculatorFunction Class Reference

Interface for PostScript calculator functions. See section 3.9.4 of the PDF 1.7 Reference. Default values are as per described in that reference.

```
#include <idomfunction.h>
```

Inheritance diagram for IDOMPostScriptCalculatorFunction:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual IInputStreamPtr getCalculatorAsPostScriptStream () const =0

  Get the PostScript Calculator function as a PostScript Procedure.
- virtual JawsMako::IPDFArrayPtr getCalculator () const =0

  Get the PostScript Calculator function as an executable PS procedure.

Public Member Functions inherited from IDOMFunction

- virtual eFunctionType getFunctionType () const =0

  Retrieves function type.
- virtual uint32 getNumInputValues () const =0

  Get the number of input values that this function will operate on.
- virtual uint32 getNumOutputValues () const =0

  Get the number of output values that this function will produce.
- virtual void getInputDomain (uint32 inputNum, float &low, float &high) const =0

  Get the input domain for a given input to the function.
- virtual bool getOutputRange (uint32 outputNum, float &low, float &high) const =0

  Get the output range for a given input to the function.
- virtual void evaluate (const float *inputValues, float *outputValues) const =0

  Evaluate the input through the function and return the result.
- virtual void evaluate (const int32 *inputValues, float *outputValues) const =0

  Evaluate the input through the function and return the result.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves class id of IDOM.

Additional Inherited Members

Public Types inherited from IDOMFunction

- enum eFunctionType

  An enum for the various types of functions.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.257.1 Detailed Description

Interface for PostScript calculator functions. See section 3.9.4 of the PDF 1.7 Reference. Default values are as per described in that reference.

7.257.2 Member Function Documentation

classID()

```
static const CClassID & IDOMPostScriptCalculatorFunction::classID ( )  [inline], [static]
```

Retrieves class id of IDOM.

Returns

CClassID Class id of the element

getCalculator()

```
virtual JawsMako::IPDFArrayPtr IDOMPostScriptCalculatorFunction::getCalculator ( ) const  [pure
virtual]
```

Get the PostScript Calculator function as an executable PS procedure.

Returns

JawsMako::IPDFArrayPtr Returns the calculator procedure

getCalculatorAsPostScriptStream()

```
virtual IInputStreamPtr IDOMPostScriptCalculatorFunction::getCalculatorAsPostScriptStream ( )
const  [pure virtual]
```

Get the PostScript Calculator function as a PostScript Procedure.

Returns

IInputStreamPtr The stream

The documentation for this class was generated from the following file:

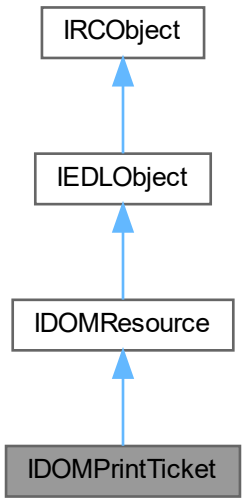
- idomfunction.h

7.258 IDOMPrintTicket Class Reference

IDOMPrintTicket interface.

```
#include <idomresources.h>
```

Inheritance diagram for IDOMPrintTicket:



Classes

- class Data

  Initialization data.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves the class id of IDOMPrintTicket.

Additional Inherited Members

Public Member Functions inherited from IDOMResource

- virtual IInputStreamPtr getStream () const =0

  Retrieves the resource stream.
- virtual void setStream (const IInputStreamPtr &stream)=0

  Sets the resource stream for the node.
- virtual uint64 getStreamLength () const =0

  Retrieves the stream length, if it is available.
- virtual const EDLSysString & getUri () const =0

  Retrieves the resource URI.
- virtual void setUri (const EDLSysString &uri)=0

  Sets the resource URI.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.258.1 Detailed Description

IDOMPrintTicket interface.

7.258.2 Member Function Documentation

classID()

```
static const CClassID & IDOMPrintTicket::classID ( )  [inline], [static]
```

Retrieves the class id of IDOMPrintTicket.

Returns

CClassID. Returns the class id of the element.

The documentation for this class was generated from the following file:

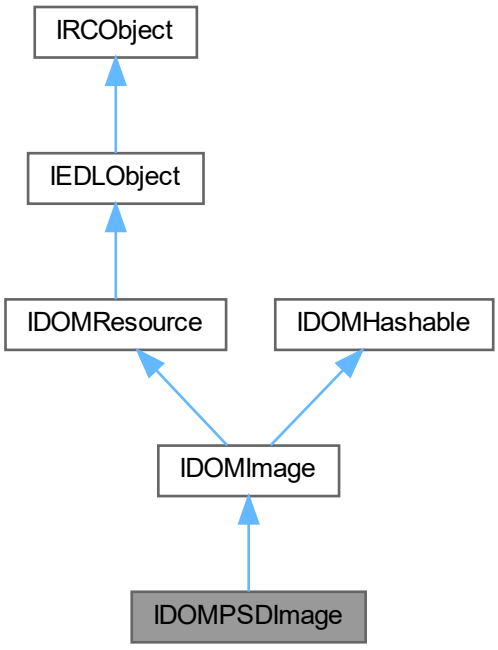
- idomresources.h

7.259 IDOMPSDImage Class Reference

IDOMPSDImage interface.

```
#include <idomimageresource.h>
```

Inheritance diagram for IDOMPSDImage:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual int16 getLayerIndex () const =0

  Get the layer index of this image from the PSD.

Public Member Functions inherited from IDOMImage

- virtual IImageDecoderPtr createImageDecoder (IEDLClassFactory *factory, const IDOMImagePropertiesPtr &imageProperties)=0

  Creates a properly initialized image decoder object that reads from an inputstream that is specific to that image format.
- virtual IImageFramePtr getImageFrame (IEDLClassFactory *factory)

  Fetch the image frame; convenience.
- virtual IImageEncoderPtr createImageEncoder (const ISessionPtr &session, const IOutputStreamPtr &imageDest, const IDOMImagePropertiesPtr &imageProperties)=0

  Creates a properly initialized image encoder object that writes to an outputstream that is specific to that image format.
- virtual IDOMImagePropertiesPtr getImageProperties ()=0

  Returns an object that stores the properties for this image object. The properties can then be inspected (or more added) by clients that need to manipulate the image resource.
- virtual eDOMImageType getImageType ()=0

  Retrieves the image type.
- virtual bool getIsRendered ()=0

  Determine if the image is as a result of rendering. This is indicated if the image type is eDITRendered or if the image explicitly notes this is the case (such as for IDOMPDFImage).
- virtual IDOMImagePtr getImageWithSubstitutedColorSpace (IEDLClassFactory *factory, const IDOMColor↩SpacePtr &colorSpace)

  Obtain an image that is the same as this image, but with the colorspace substituted for another.

Public Member Functions inherited from IDOMResource

- virtual IInputStreamPtr getStream () const =0

  Retrieves the resource stream.
- virtual void setStream (const IInputStreamPtr &stream)=0

  Sets the resource stream for the node.
- virtual uint64 getStreamLength () const =0

  Retrieves the stream length, if it is available.
- virtual const EDLSysString & getUri () const =0

  Retrieves the resource URI.
- virtual void setUri (const EDLSysString &uri)=0

  Sets the resource URI.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static EDL_API IDOMPSDImagePtr create (IEDLClassFactory *pFactory, const IInputStreamPtr &stream, int16 layerIndex=0)

  Create a PSD Image resource with the given PSD stream.
- static const CClassID & classID ()

  Retrieves class id of IDOMPSDImage.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.259.1 Detailed Description

IDOMPSDImage interface.

7.259.2 Member Function Documentation

classID()

```
static const CClassID & IDOMPSDImage::classID ( )  [inline], [static]
```

Retrieves class id of IDOMPSDImage.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMPSDImagePtr IDOMPSDImage::create (
            IEDLClassFactory * pFactory,
            const IInputStreamPtr & stream,
            int16 layerIndex = 0 )  [static]
```

Create a PSD Image resource with the given PSD stream.

Parameters

pFactory	The EDL Class factory to use.
stream	The stream containing the PSD image.
layerIndex	For multi-layer PSD files, the index of the layer to use, with 0 being the composite image.

Returns

IDOMPSDImagePtr The new image.

getLayerIndex()

```
virtual int16 IDOMPSDImage::getLayerIndex ( ) const  [pure virtual]
```

Get the layer index of this image from the PSD.

PSD files may contain multiple layers; this member returns the index (beginning at 0) of the layer that will be used from the PSD file. If 0 then this is the composite image composed of all layers.

Returns

int16 The layer index

The documentation for this class was generated from the following file:

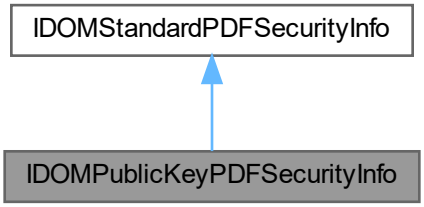
- idomimageresource.h

7.260 IDOMPublicKeyPDFSecurityInfo Class Reference

Represents security information from PDF public key (certificate based) security handling. Inherits from IDOMStandardPDFSecurityInfo, from which it diverges in a modest fashion.

`#include <idomsecurity.h>`

Inheritance diagram for IDOMPublicKeyPDFSecurityInfo:



Public Types

- enum ePublicKeyPermissionsFlags { eChangeEncryptionAllowed = 0x0002 }

Public Types inherited from IDOMStandardPDFSecurityInfo

- enum ePermissionsFlags {
  ePrintAllowed = 0x0004 , eEditingAllowed = 0x0008 , eCopyingAllowed = 0x0010 , eAnnotationEditingAllowed = 0x0020 ,
  eFormFillingAllowed = 0x0100 , eContentAccessibilityExtractionAllowed = 0x0200 , eDocumentAssemblyAllowed = 0x0400 , eHighQualityPrintAllowed = 0x0800 ,
  eEverythingAllowed = 0x0ffc }

  Bit values for each permission flag as available from getPermissionFlags.

Public Member Functions

- virtual bool isOwnerAccess () const =0

  Returns true if the input has permissions to change input encryption settings, effectively granting owner access.
- virtual EDLSysString getUserPassword () const

  The password is never returned for public key cryptography.

Public Member Functions inherited from IDOMStandardPDFSecurityInfo

- virtual int32 getHandlerRevision () const =0

  Retrieves the revision of the handler.
- virtual bool encryptMetadata () const =0

  Returns true if document level metadata stream is to be encrypted meaningful only when algorithm code = 4.
- bool isPrintingAllowed () const

  Returns true if printing is allowed.
- bool isHighQualityPrintingAllowed () const

  Returns true if high quality printing is allowed.
- bool isEditingAllowed () const

  Returns true if modification of the contents by operations other than those specified by isEditingAnnotationsAllowed(), isFillingFormAllowed() and isDocAssemblyAllowed() is allowed.
- bool isCopyingAllowed () const

  Returns true if copying of text and graphics from the document for operations other than those specified by isExtractionAllowed() is allowed.
- bool isAnnotationEditingAllowed () const

  Returns true if editing of annotations and forms is allowed.
- bool isFillingFormsAllowed () const

  Returns true if form-filling is allowed.
- bool isContentAccessibilityExtractionAllowed () const

  Returns true if extracting for content accessibility is allowed.
- bool isDocumentAssemblyAllowed () const

  Returns true if document assembly is allowed.
- virtual uint32 **getPermissionFlags** () const =0

  Retrieve the raw permissions flags according to the PDF 1.7 spec, table 3.2.0. These should be interpreted with regard to the handler revision, as some flags are not available for all security handler revisions.

7.260.1 Detailed Description

Represents security information from PDF public key (certificate based) security handling. Inherits from IDOMStandardPDFSecurityInfo, from which it diverges in a modest fashion.

7.260.2 Member Enumeration Documentation

ePublicKeyPermissionsFlags

```
enum IDOMPublicKeyPDFSecurityInfo::ePublicKeyPermissionsFlags
```

Enumerator

eChangeEncryptionAllowed	Allow encryption parameters to be changed. Effectively grants ownership access.

7.260.3 Member Function Documentation

getUserPassword()

```
virtual EDLSysString IDOMPublicKeyPDFSecurityInfo::getUserPassword ( ) const  [inline], [virtual]
```

The password is never returned for public key cryptography.

Returns

EDLSysString Always an empty string

Implements IDOMStandardPDFSecurityInfo.

isOwnerAccess()

```
virtual bool IDOMPublicKeyPDFSecurityInfo::isOwnerAccess ( ) const  [pure virtual]
```

Returns true if the input has permissions to change input encryption settings, effectively granting owner access.

Returns

bool

Implements IDOMStandardPDFSecurityInfo.

The documentation for this class was generated from the following file:

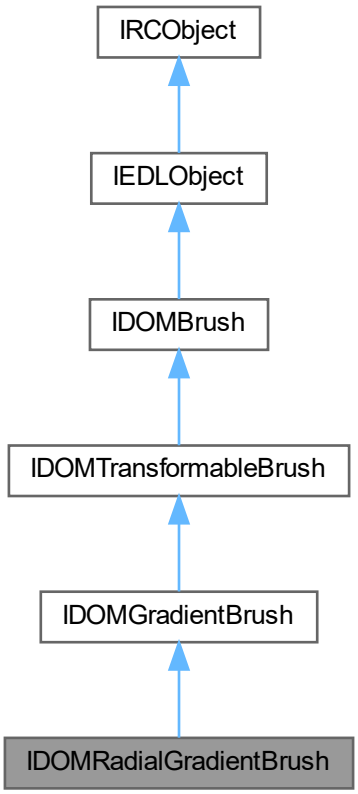
- idomsecurity.h

7.261 IDOMRadialGradientBrush Class Reference

IDOMRadialGradientBrush interface. A radial gradient brush defines an ellipse to be filled with the gradient. The ellipse is defined by its center, x radius, and y radius. Independently, a gradient origin is specified for the brush. The gradient origin defines the center of the gradient; a gradient stop with an offset at 0.0 defines the color at the gradient origin. The outer bound of the ellipse defines the end "point" of the gradient; that is, a gradient stop with an offset at 1.0 defines the color at the circumference of the ellipse, and all other gradient stops define their offsets relative to the radial distance between the gradient origin and the circumference.

```
#include <idombrush.h>
```

Inheritance diagram for IDOMRadialGradientBrush:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual const FPoint & getCenter () const =0

  Retrieves the center point of the gradient brush ellipse.
- virtual void setCenter (const FPoint &pt)=0

  Sets the center point of the gradient brush ellipse.
- virtual const FPoint & getGradientOrigin () const =0

  Retrieves the origin point of the radial gradient, corresponding to the gradient stop with an offset of 0.0.
- virtual void setGradientOrigin (const FPoint &pt)=0

  Sets the origin point of the radial gradient, corresponding to the gradient stop with an offset of 0.0.
- virtual double getRadiusX () const =0

  Retrieves the x-radius of the gradient brush ellipse.
- virtual void setRadiusX (double r)=0

  Sets the x-radius of the gradient brush ellipse.

- virtual double getRadiusY () const =0

  Retrieves the y-radius of the gradient brush ellipse.
- virtual void setRadiusY (double r)=0

  Sets the y-radius of the gradient brush ellipse.
- virtual IDOMShadingPatternType3BrushPtr createShading (IEDLClassFactory *pFactory, const FRect &fillArea, bool useFirstStopColorSpace=false)=0

  Create a Type3 Shading Pattern brush from this radial brush. All alpha information in the gradient stops will be dropped. The resulting brush will be generated to fill the given rectangle.
- virtual IDOMRadialGradientBrushPtr getSimplifiedGradient (IEDLClassFactory *pFactory, const FRect &fillArea)=0

  Create a simplified radial gradient brush, where any repeat or reflect pad mode is converted to simple padding by repeating gradient stops as required. The resulting brush will be generated to fill the given rectangle.

Public Member Functions inherited from IDOMGradientBrush

- virtual eColorInterpolationMode getColorInterpolationMode () const =0

  Retrieves the color interpolation mode value of the radial gradient brush.
- virtual void setColorInterpolationMode (eColorInterpolationMode cim)=0

  Sets the color interpolation mode value of the radial gradient brush.
- virtual eSpreadMethod getSpreadMethod () const =0

  Retrieves the spread method value of the RadialGradientBrush element.
- virtual void setSpreadMethod (eSpreadMethod sm)=0

  Sets spread method value of the RadialGradientBrush element.
- virtual void setGradientStops (const CDOMGradientStopVect &stops)=0

  Set the vector of stops in this gradient. Must not be empty.
- virtual const CDOMGradientStopVect & getGradientStops () const =0

  Retrieves the vector of stops in this gradient. An exception will be thrown if the gradient has no stops.
- virtual void addGradientStop (const IDOMGradientStopPtr &ptrGradientStop)=0

  Append a gradient stop.
- virtual void normalizeStops (IEDLClassFactory *pFactory, const IDOMColorSpacePtr &ptrColorSpace=IDOMColorSpacePtr(), eRenderingIntent intent=eRelativeColorimetric, eBlackPointCompensation bpc=eBPCDefault)=0

  Normalize the gradient stops to a single color space, sort them and ensure there are stops at 0.0 and 1.0.

Public Member Functions inherited from IDOMTransformableBrush

- virtual const FMatrix & getRenderTransform () const =0

  Retrieves the render transform matrix.
- virtual void setRenderTransform (const FMatrix &matrix)=0

  Sets the render transform matrix.

Public Member Functions inherited from IDOMBrush

- virtual eBrushType getBrushType () const =0

  Retrieves the type of the brush.
- virtual float getOpacity () const =0

  Retrieves the opacity value of the brush element.
- virtual void setOpacity (float opc)=0

  Sets the opacity value of a brush element.
- virtual IDOMBrushPtr getAdjustedForUseInTransformedNode (IEDLClassFactory *pFactory, const FMatrix &nodeTransform)

  Get a version of this brush adjusted for use inside a node with the given transform.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static EDL_API IDOMRadialGradientBrushPtr create (IEDLClassFactory *pFactory, const FPoint &center, const FPoint &gradientOrigin, double radiusX, double radiusY, const CDOMGradientStopVect &stops, float opacity=1.0f, const FMatrix &renderTransform=FMatrix(), eSpreadMethod spreadMethod=eNoSpread, eColorInterpolationMode colorInterpolationMode=eSRgbLinearInterpolation)

  Simplified radial gradient brush creation. Throws an IEDLError on failure.
- static const CClassID & classID ()

  Retrieves class id of IDOM.

Additional Inherited Members

Public Types inherited from IDOMBrush

- enum eBrushType {
  eSolidColor , eLinearGradient , eRadialGradient , eImage ,
  eMasked , eVisual , eSoftMask , eTilingPattern ,
  eType1ShadingPattern , eType2ShadingPattern , eType3ShadingPattern , eType4567ShadingPattern ,
  eNull }

  Brush type enumeration.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.261.1 Detailed Description

IDOMRadialGradientBrush interface. A radial gradient brush defines an ellipse to be filled with the gradient. The ellipse is defined by its center, x radius, and y radius. Independently, a gradient origin is specified for the brush. The gradient origin defines the center of the gradient; a gradient stop with an offset at 0.0 defines the color at the gradient origin. The outer bound of the ellipse defines the end "point" of the gradient; that is, a gradient stop with an offset at 1.0 defines the color at the circumference of the ellipse, and all other gradient stops define their offsets relative to the radial distance between the gradient origin and the circumference.

7.261.2 Member Function Documentation

classID()

```
static const CClassID & IDOMRadialGradientBrush::classID ( )  [inline], [static]
```

Retrieves class id of IDOM.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMRadialGradientBrushPtr IDOMRadialGradientBrush::create (
            IEDLClassFactory * pFactory,
            const FPoint & center,
            const FPoint & gradientOrigin,
            double radiusX,
            double radiusY,
            const CDOMGradientStopVect & stops,
            float opacity = 1.0f,
            const FMatrix & renderTransform = FMatrix(),
            eSpreadMethod spreadMethod = eNoSpread,
            eColorInterpolationMode colorInterpolationMode = eSRgbLinearInterpolation )  [static]
```

Simplified radial gradient brush creation. Throws an IEDLError on failure.

Parameters

pFactory	The EDL Class factory to use.
center	The center point of the gradient brush ellipse
gradientOrigin	The origin point of the gradient
radiusX	The x radius of the gradient brush ellipse
radiusY	The y radius of the gradient brush ellipse
stops	The gradient stops
opacity	The alpha to use.
renderTransform	The render transform to use
spreadMethod	The desired spread method
colorInterpolationMode	The desired color interpolation mode

Returns

IDOMRadialGradientBrushPtr The new brush.

createShading()

```
virtual IDOMShadingPatternType3BrushPtr IDOMRadialGradientBrush::createShading (
            IEDLClassFactory * pFactory,
            const FRect & fillArea,
            bool useFirstStopColorSpace = false )  [pure virtual]
```

Create a Type3 Shading Pattern brush from this radial brush. All alpha information in the gradient stops will be dropped. The resulting brush will be generated to fill the given rectangle.

Parameters

pFactory	The EDL class factory
fillArea	The rectangular area that needs to be filled by the resulting brush.
useFirstStopColorSpace	If true the color space in the first stop will be used for the shading. Otherwise either sRGB or scRGB will be used depending on the interpolation mode.

Returns

IDOMShadingPatternType3BrushPtr The resulting brush.

getCenter()

```
virtual const FPoint & IDOMRadialGradientBrush::getCenter ( ) const  [pure virtual]
```

Retrieves the center point of the gradient brush ellipse.

Returns

FPoint The center point of the gradient brush ellipse.

getGradientOrigin()

```
virtual const FPoint & IDOMRadialGradientBrush::getGradientOrigin ( ) const  [pure virtual]
```

Retrieves the origin point of the radial gradient, corresponding to the gradient stop with an offset of 0.0.

Returns

FPoint The origin point of the radial gradient.

getRadiusX()

```
virtual double IDOMRadialGradientBrush::getRadiusX ( ) const  [pure virtual]
```

Retrieves the x-radius of the gradient brush ellipse.

Returns

double The x-radius of the gradient brush ellipse.

getRadiusY()

```
virtual double IDOMRadialGradientBrush::getRadiusY ( ) const  [pure virtual]
```

Retrieves the y-radius of the gradient brush ellipse.

Returns

double The y-radius of the gradient brush ellipse.

getSimplifiedGradient()

```
virtual IDOMRadialGradientBrushPtr IDOMRadialGradientBrush::getSimplifiedGradient (
            IEDLClassFactory * pFactory,
            const FRect & fillArea )  [pure virtual]
```

Create a simplified radial gradient brush, where any repeat or reflect pad mode is converted to simple padding by repeating gradient stops as required. The resulting brush will be generated to fill the given rectangle.

Parameters

pFactory	The EDL class factory
fillArea	The rectangular area that needs to be filled by the resulting brush.

Returns

IDOMRadialGradientBrushPtr The simplified brush, or this brush if no action was required. This brush will be cached, so do not edit; make a clone if the returned brush needs to be changed.

setCenter()

```
virtual void IDOMRadialGradientBrush::setCenter (
            const FPoint & pt )  [pure virtual]
```

Sets the center point of the gradient brush ellipse.

Parameters

pt	Reference parameter to receive the center point of the gradient brush ellipse.

setGradientOrigin()

```
virtual void IDOMRadialGradientBrush::setGradientOrigin (
            const FPoint & pt )  [pure virtual]
```

Sets the origin point of the radial gradient, corresponding to the gradient stop with an offset of 0.0.

Parameters

pt	The origin point of the radial gradient.

setRadiusX()

```
virtual void IDOMRadialGradientBrush::setRadiusX (
            double r )  [pure virtual]
```

Sets the x-radius of the gradient brush ellipse.

Parameters

r	The x-radius of the ellipse

setRadiusY()

```
virtual void IDOMRadialGradientBrush::setRadiusY (
            double r )  [pure virtual]
```

Sets the y-radius of the gradient brush ellipse.

Parameters

r	The y-radius of the gradient brush ellipse.

The documentation for this class was generated from the following file:

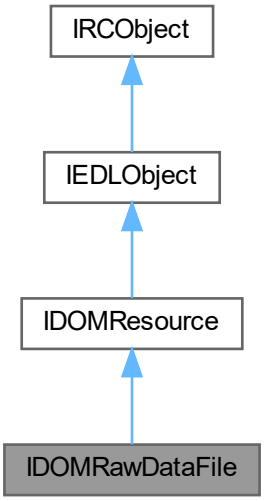
- idombrush.h

7.262 IDOMRawDataFile Class Reference

IDOMRawDataFile interface.

```
#include <idomresources.h>
```

Inheritance diagram for IDOMRawDataFile:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual const EDLSysString & **getMimeType** () const =0

  Get the mime type of this raw data file. return EDLSysString the mime type.

Public Member Functions inherited from IDOMResource

- virtual IInputStreamPtr getStream () const =0

  Retrieves the resource stream.
- virtual void setStream (const IInputStreamPtr &stream)=0

  Sets the resource stream for the node.
- virtual uint64 getStreamLength () const =0

  Retrieves the stream length, if it is available.
- virtual const EDLSysString & getUri () const =0

  Retrieves the resource URI.
- virtual void setUri (const EDLSysString &uri)=0

  Sets the resource URI.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves class id of IDOM.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.262.1 Detailed Description

IDOMRawDataFile interface.

7.262.2 Member Function Documentation

classID()

```
static const CClassID & IDOMRawDataFile::classID ( )  [inline], [static]
```

Retrieves class id of IDOM.

Returns

CClassID Class id of the element

The documentation for this class was generated from the following file:

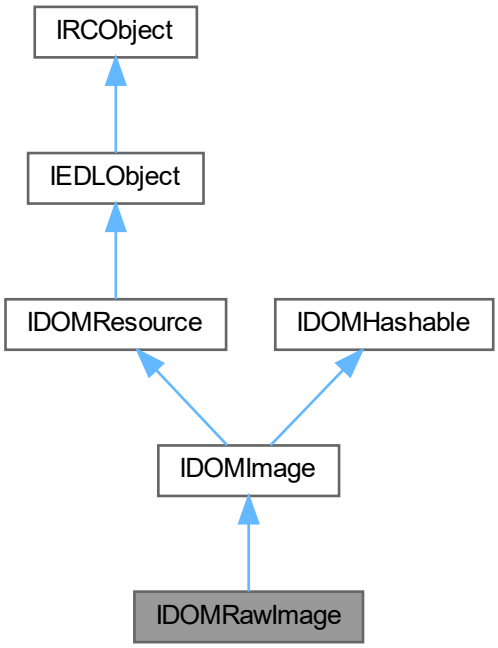
- idomresources.h

7.263 IDOMRawImage Class Reference

Interface to a class representing a raw image.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMRawImage:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual bool getSynthetic () const =0

  Returns a Boolean value indicating whether or not the image is synthetic.

Public Member Functions inherited from IDOMImage

- virtual IImageDecoderPtr createImageDecoder (IEDLClassFactory *factory, const IDOMImagePropertiesPtr &imageProperties)=0

  Creates a properly initialized image decoder object that reads from an inputstream that is specific to that image format.
- virtual IImageFramePtr getImageFrame (IEDLClassFactory *factory)

  Fetch the image frame; convenience.

- virtual IImageEncoderPtr createImageEncoder (const ISessionPtr &session, const IOutputStreamPtr &imageDest, const IDOMImagePropertiesPtr &imageProperties)=0

  Creates a properly initialized image encoder object that writes to an outputstream that is specific to that image format.
- virtual IDOMImagePropertiesPtr getImageProperties ()=0

  Returns an object that stores the properties for this image object. The properties can then be inspected (or more added) by clients that need to manipulate the image resource.
- virtual eDOMImageType getImageType ()=0

  Retrieves the image type.
- virtual bool getIsRendered ()=0

  Determine if the image is as a result of rendering. This is indicated if the image type is eDITRendered or if the image explicitly notes this is the case (such as for IDOMPDFImage).
- virtual IDOMImagePtr getImageWithSubstitutedColorSpace (IEDLClassFactory *factory, const IDOMColorSpacePtr &colorSpace)

  Obtain an image that is the same as this image, but with the colorspace substituted for another.

Public Member Functions inherited from IDOMResource

- virtual IInputStreamPtr getStream () const =0

  Retrieves the resource stream.
- virtual void setStream (const IInputStreamPtr &stream)=0

  Sets the resource stream for the node.
- virtual uint64 getStreamLength () const =0

  Retrieves the stream length, if it is available.
- virtual const EDLSysString & getUri () const =0

  Retrieves the resource URI.
- virtual void setUri (const EDLSysString &uri)=0

  Sets the resource URI.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static EDL_API IDOMRawImagePtr create (IEDLClassFactory *pFactory, const IDOMColorSpacePtr &colorSpace, const IInputStreamPtr &stream, eDOMImageType type=eDITUnknown)

  Create a "raw" image resource with the given stream.
- static const CClassID & classID ()

  Retrieves class id of IDOMRawImage.
- static EDL_API IDOMImagePtr createWriterAndImage (const ISessionPtr &session, IImageFrameWriterPtr &frame, const IDOMColorSpacePtr &colorSpace, uint32 width, uint32 height, uint8 bitsPerComponent=8, double xResolution=96.0, double yResolution=96.0, eImageExtraChannelType extraChannel=eIECNone, const IInputStreamPtr &inStream=IInputStreamPtr(), const IOutputStreamPtr &outStream=IOutputStream↩ Ptr())

  Create an IDOMRawImage and frame that can be used to populate same.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.263.1 Detailed Description

Interface to a class representing a raw image.

In EDL, a raw image is an image which is represented by raw image pixels, capable of handling common bit depths and any color space, with a trivial header.

7.263.2 Member Function Documentation

classID()

```
static const CClassID & IDOMRawImage::classID ( )  [inline], [static]
```

Retrieves class id of IDOMRawImage.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMRawImagePtr IDOMRawImage::create (
            IEDLClassFactory * pFactory,
            const IDOMColorSpacePtr & colorSpace,
            const IInputStreamPtr & stream,
            eDOMImageType type = eDITUnknown )  [static]
```

Create a "raw" image resource with the given stream.

Parameters

pFactory	The EDL Class factory to use.
colorSpace	The color space of the data
stream	The stream containing the raw image.
type	The image type

Returns

IDOMRawImagePtr The new image.

createWriterAndImage()

```
static EDL_API IDOMImagePtr IDOMRawImage::createWriterAndImage (
            const ISessionPtr & session,
            IImageFrameWriterPtr & frame,
            const IDOMColorSpacePtr & colorSpace,
            uint32 width,
            uint32 height,
            uint8 bitsPerComponent = 8,
            double xResolution = 96.0,
            double yResolution = 96.0,
            eImageExtraChannelType extraChannel = eIECNone,
            const IInputStreamPtr & inStream = IInputStreamPtr(),
            const IOutputStreamPtr & outStream = IOutputStreamPtr() )  [static]
```

Create an IDOMRawImage and frame that can be used to populate same.

Parameters

session	The session to use
frame	On exit, this is populated with a frame ready to receive image data via frame->writeScanLine(). Use frame->flushData() to complete the encoding process.
colorSpace	The color space to use. Any valid color space may be used for IDOMRawImages.
width	The width of the image, in pixels.
height	The height of the image, in pixels.
bitsPerComponent	The bits per component to use. 1, 2, 4, 8, 12 and 16 bits per sample are supported.
xResolution	The x resolution, in pixels-per-inch.
yResolution	The y resolution, in pixels-per-inch.
extraChannel	The type of extra channel, if provided.
inStream	Optional. The first in a pair of streams used to read and write the raw image data if an external stream is desired. If NULL, a temporary store stream will be created. If non NULL, outStream must also be provided.
outStream	Optional. The second in a pair of streams used to read and write the raw image data if an external stream is desired. If NULL, a temporary store stream will be created. If non NULL, inStream must also be provided.

Returns

IDOMImagePtr The resulting image. Not valid until the frame is flushed.

getSynthetic()

```
virtual bool IDOMRawImage::getSynthetic ( ) const  [pure virtual]
```

Returns a Boolean value indicating whether or not the image is synthetic.

Returns

bool Returns true if the image is synthetic, false otherwise.

The documentation for this class was generated from the following file:

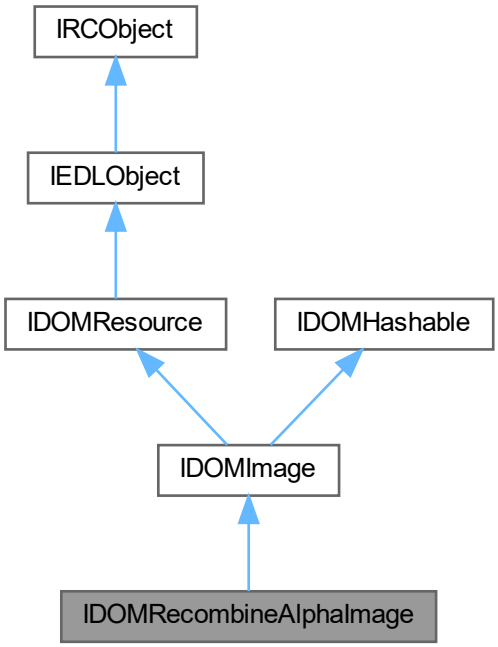
- idomimageresource.h

7.264 IDOMRecombineAlphaImage Class Reference

Similar to IDOMRecombineImage, but instead combines an image comprising the color components of the image, with a single-channel image that represents the mask or alpha channel. The images must have the same, dimensions, but may have different dimensions. The resolution information will be taken from the color image. Images with Indexed color spaces will be converted to the base spaces.

```
#include <idomimageresource.h>
```

Inheritance diagram for IDOMRecombineAlphaImage:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual bool getStream (IInputStreamPtr &stream) const

  This image type does not allow direct access to the underlying streams.
- virtual void setStream (const IInputStreamPtr &stream)

  This image type does not allow direct access to the underlying streams.

Public Member Functions inherited from IDOMImage

- virtual IImageDecoderPtr createImageDecoder (IEDLClassFactory *factory, const IDOMImagePropertiesPtr &imageProperties)=0

  Creates a properly initialized image decoder object that reads from an inputstream that is specific to that image format.
- virtual IImageFramePtr getImageFrame (IEDLClassFactory *factory)

  Fetch the image frame; convenience.
- virtual IImageEncoderPtr createImageEncoder (const ISessionPtr &session, const IOutputStreamPtr &imageDest, const IDOMImagePropertiesPtr &imageProperties)=0

  Creates a properly initialized image encoder object that writes to an outputstream that is specific to that image format.
- virtual IDOMImagePropertiesPtr getImageProperties ()=0

  Returns an object that stores the properties for this image object. The properties can then be inspected (or more added) by clients that need to manipulate the image resource.
- virtual eDOMImageType getImageType ()=0

  Retrieves the image type.
- virtual bool getIsRendered ()=0

  Determine if the image is as a result of rendering. This is indicated if the image type is eDITRendered or if the image explicitly notes this is the case (such as for IDOMPDFImage).
- virtual IDOMImagePtr getImageWithSubstitutedColorSpace (IEDLClassFactory *factory, const IDOMColor↩SpacePtr &colorSpace)

  Obtain an image that is the same as this image, but with the colorspace substituted for another.

Public Member Functions inherited from IDOMResource

- virtual IInputStreamPtr getStream () const =0

  Retrieves the resource stream.
- virtual uint64 getStreamLength () const =0

  Retrieves the stream length, if it is available.
- virtual const EDLSysString & getUri () const =0

  Retrieves the resource URI.
- virtual void setUri (const EDLSysString &uri)=0

  Sets the resource URI.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static EDL_API IDOMRecombineAlphaImagePtr create (IEDLClassFactory *pFactory, const IDOMImagePtr &colorImage, const IDOMImagePtr &extraChannel, bool extraChannelIsMask=false)

  Simplified creator for a IDOMRecombineAlphaImage.
- static const CClassID & classID ()

  Retrieves class id of IDOMRecombineAlphaImage.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.264.1 Detailed Description

Similar to IDOMRecombineImage, but instead combines an image comprising the color components of the image, with a single-channel image that represents the mask or alpha channel. The images must have the same, dimensions, but may have different dimensions. The resolution information will be taken from the color image. Images with Indexed color spaces will be converted to the base spaces.

7.264.2 Member Function Documentation

classID()

```
static const CClassID & IDOMRecombineAlphaImage::classID ( )  [inline], [static]
```

Retrieves class id of IDOMRecombineAlphaImage.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMRecombineAlphaImagePtr IDOMRecombineAlphaImage::create (
            IEDLClassFactory * pFactory,
            const IDOMImagePtr & colorImage,
            const IDOMImagePtr & extraChannel,
            bool extraChannelIsMask = false )  [static]
```

Simplified creator for a IDOMRecombineAlphaImage.

Parameters

pFactory	The EDL Class Factory
colorImage	The image with the color components
extraChannel	The image with the mask/alpha channel
extraChannelIsMask	If true, the extra channel will be noted as a mask rather than alpha

Returns

IDOMRecombineAlphaImagePtr The combined result.

getStream()

```
virtual bool IDOMRecombineAlphaImage::getStream (
            IInputStreamPtr & stream ) const  [inline], [virtual]
```

This image type does not allow direct access to the underlying streams.

Parameters

stream	A smart pointer to the stream

Returns

bool Always false

setStream()

```
virtual void IDOMRecombineAlphaImage::setStream (
            const IInputStreamPtr & stream )  [inline], [virtual]
```

This image type does not allow direct access to the underlying streams.

Parameters

stream	A smart pointer to the stream

Implements IDOMResource.

The documentation for this class was generated from the following file:

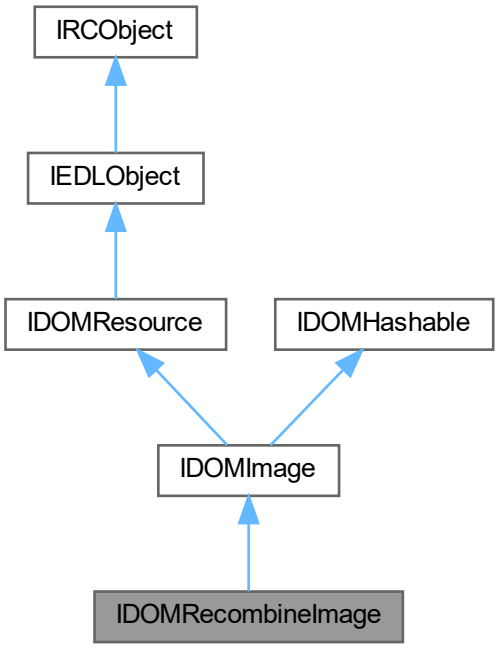
- idomimageresource.h

7.265 IDOMRecombineImage Class Reference

Interface to a class representing a image made up of separate single channel images (each with the same bps, dimensions and resolution) each representing a single component of the entire image, or a mask channel.

```
#include <idomimageresource.h>
```

Inheritance diagram for IDOMRecombineImage:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual bool getStream (IInputStreamPtr &stream) const

  This image type does not allow direct access to the underlying streams.
- virtual void setStream (const IInputStreamPtr &stream)

  This image type does not allow direct access to the underlying streams.

Public Member Functions inherited from IDOMImage

- virtual IImageDecoderPtr createImageDecoder (IEDLClassFactory *factory, const IDOMImagePropertiesPtr &imageProperties)=0

  Creates a properly initialized image decoder object that reads from an inputstream that is specific to that image format.
- virtual IImageFramePtr getImageFrame (IEDLClassFactory *factory)

  Fetch the image frame; convenience.
- virtual IImageEncoderPtr createImageEncoder (const ISessionPtr &session, const IOutputStreamPtr &imageDest, const IDOMImagePropertiesPtr &imageProperties)=0

  Creates a properly initialized image encoder object that writes to an outputstream that is specific to that image format.
- virtual IDOMImagePropertiesPtr getImageProperties ()=0

  Returns an object that stores the properties for this image object. The properties can then be inspected (or more added) by clients that need to manipulate the image resource.
- virtual eDOMImageType getImageType ()=0

  Retrieves the image type.
- virtual bool getIsRendered ()=0

  Determine if the image is as a result of rendering. This is indicated if the image type is eDITRendered or if the image explicitly notes this is the case (such as for IDOMPDFImage).
- virtual IDOMImagePtr getImageWithSubstitutedColorSpace (IEDLClassFactory *factory, const IDOMColor↩SpacePtr &colorSpace)

  Obtain an image that is the same as this image, but with the colorspace substituted for another.

Public Member Functions inherited from IDOMResource

- virtual IInputStreamPtr getStream () const =0

  Retrieves the resource stream.
- virtual uint64 getStreamLength () const =0

  Retrieves the stream length, if it is available.
- virtual const EDLSysString & getUri () const =0

  Retrieves the resource URI.
- virtual void setUri (const EDLSysString &uri)=0

  Sets the resource URI.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static EDL_API IDOMRecombineImagePtr create (IEDLClassFactory *pFactory, const IDOMColorSpace↩ Ptr &colorSpace, const CEDLVector< IDOMImagePtr > &componentImages, const IDOMImagePtr &extra↩ Channel=IDOMImagePtr(), bool extraChannelIsMask=false)

  Convenience creator for a recombine image.
- static const CClassID & classID ()

  Retrieves class id of IDOMRecombineImage.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.265.1 Detailed Description

Interface to a class representing a image made up of separate single channel images (each with the same bps, dimensions and resolution) each representing a single component of the entire image, or a mask channel.

7.265.2 Member Function Documentation

classID()

```
static const CClassID & IDOMRecombineImage::classID ( )  [inline], [static]
```

Retrieves class id of IDOMRecombineImage.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMRecombineImagePtr IDOMRecombineImage::create (
            IEDLClassFactory * pFactory,
            const IDOMColorSpacePtr & colorSpace,
            const CEDLVector< IDOMImagePtr > & componentImages,
            const IDOMImagePtr & extraChannel = IDOMImagePtr(),
            bool extraChannelIsMask = false )  [static]
```

Convenience creator for a recombine image.

Parameters

pFactory	The class factory to use
colorSpace	The color space that describes the color channels
componentImages	A vector of images each providing a single colour channel
extraChannel	The image to be used as an alpha channel or mask, if required
extraChannelIsMask	If true, the extraChannel is treated as a mask channel, not an alpha channel

Returns

IDOMRecombineImagePtr The recombined image

getStream()

```
virtual bool IDOMRecombineImage::getStream (
            IInputStreamPtr & stream ) const  [inline], [virtual]
```

This image type does not allow direct access to the underlying streams.

Parameters

stream	A smart pointer to the stream

Returns

bool Always false

setStream()

```
virtual void IDOMRecombineImage::setStream (
            const IInputStreamPtr & stream )  [inline], [virtual]
```

This image type does not allow direct access to the underlying streams.

Parameters

stream	A smart pointer to the stream

Implements IDOMResource.

The documentation for this class was generated from the following file:

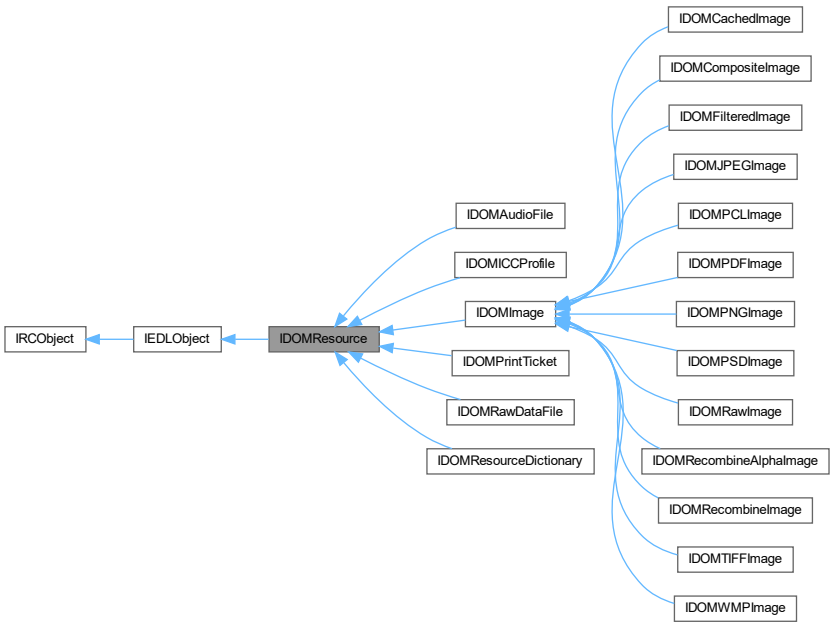
- idomimageresource.h

7.266 IDOMResource Class Reference

Provides an interface to an EDL DOM node representing a generalised resource. A resource represents non-markup document content such as images, fonts and profiles. Resources are generally stream based. This class provides the base class for interfaces to more specialized resource node types.

```
#include <idomresources.h>
```

Inheritance diagram for IDOMResource:



Public Member Functions

- virtual IInputStreamPtr getStream () const =0

  Retrieves the resource stream.
- virtual void setStream (const IInputStreamPtr &stream)=0

  Sets the resource stream for the node.
- virtual uint64 getStreamLength () const =0

  Retrieves the stream length, if it is available.
- virtual const EDLSysString & getUri () const =0

  Retrieves the resource URI.
- virtual void setUri (const EDLSysString &uri)=0

  Sets the resource URI.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.266.1 Detailed Description

Provides an interface to an EDL DOM node representing a generalised resource. A resource represents non-markup document content such as images, fonts and profiles. Resources are generally stream based. This class provides the base class for interfaces to more specialized resource node types.

7.266.2 Member Function Documentation

getStream()

```
virtual IInputStreamPtr IDOMResource::getStream ( ) const  [pure virtual]
```

Retrieves the resource stream.

Returns

IInputStreamPtr. A smart pointer to the resource stream. Can be NULL if there is no stream.

getStreamLength()

```
virtual uint64 IDOMResource::getStreamLength ( ) const  [pure virtual]
```

Retrieves the stream length, if it is available.

Returns

uint64. The stream length.

getUri()

```
virtual const EDLSysString & IDOMResource::getUri ( ) const  [pure virtual]
```

Retrieves the resource URI.

Returns

bool. Returns a constant reference to the resource URI.

setStream()

```
virtual void IDOMResource::setStream (
            const IInputStreamPtr & stream )  [pure virtual]
```

Sets the resource stream for the node.

Parameters

stream	Smart pointer to the new resource stream.

Implemented in IDOMRecombineImage, IDOMRecombineAlphaImage, IDOMCompositeImage, IDOMCachedImage, and IDOMFilteredImage.

setUri()

```
virtual void IDOMResource::setUri (
            const EDLSysString & uri )  [pure virtual]
```

Sets the resource URI.

Parameters

uri	The new resource URI.

The documentation for this class was generated from the following file:

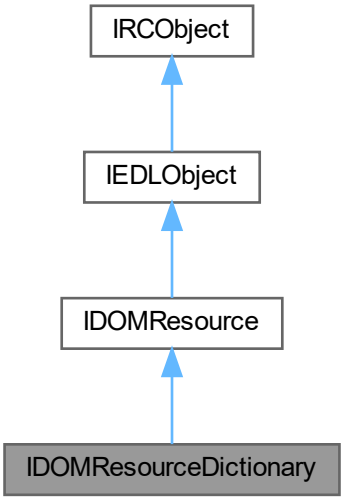
- idomresources.h

7.267 IDOMResourceDictionary Class Reference

Interface to the EDL DOM's resource dictionary. The resource dictionary is a document resource that is shared between page markups. It holds a reference list of non-markup content that is shared between multiple pages of the document.

```
#include <idomresources.h>
```

Inheritance diagram for IDOMResourceDictionary:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual IEDLObjectPtr get (const EDLSysString &name) const =0

  Retrieves a smart pointer to a resource stored in the resource dictionary.
- virtual void put (const EDLSysString &name, const IEDLObjectPtr &element)=0

  Registers a resource element in the resource dictionary.

Public Member Functions inherited from IDOMResource

- virtual IInputStreamPtr getStream () const =0

  Retrieves the resource stream.
- virtual void setStream (const IInputStreamPtr &stream)=0

  Sets the resource stream for the node.
- virtual uint64 getStreamLength () const =0

  Retrieves the stream length, if it is available.
- virtual const EDLSysString & getUri () const =0

  Retrieves the resource URI.
- virtual void setUri (const EDLSysString &uri)=0

  Sets the resource URI.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves the class id of IDOMResourceDictionary.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.267.1 Detailed Description

Interface to the EDL DOM's resource dictionary. The resource dictionary is a document resource that is shared between page markups. It holds a reference list of non-markup content that is shared between multiple pages of the document.

7.267.2 Member Function Documentation

classID()

```
static const CClassID & IDOMResourceDictionary::classID ( )  [inline], [static]
```

Retrieves the class id of IDOMResourceDictionary.

Returns

CClassID. Returns the class id of the element

get()

```
virtual IEDLObjectPtr IDOMResourceDictionary::get (
            const EDLSysString & name ) const  [pure virtual]
```

Retrieves a smart pointer to a resource stored in the resource dictionary.

Parameters

name	The name of the resource to be retrieved.

Returns

IEDLObjectPtr. A smart pointer to the resource (as an IEDLObject pointer).

put()

```
virtual void IDOMResourceDictionary::put (
            const EDLSysString & name,
            const IEDLObjectPtr & element )  [pure virtual]
```

Registers a resource element in the resource dictionary.

Parameters

name	The name of the resource to add.
element	Smart pointer to an IEDLObject representing the resource.

The documentation for this class was generated from the following file:

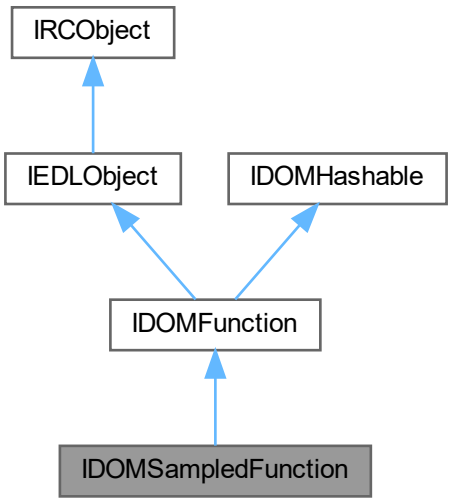
- idomresources.h

7.268 IDOMSampledFunction Class Reference

Interface for sampled functions. See section 3.9.1 of the PDF 1.7 Reference. Default values are as per described in that reference.

`#include <idomfunction.h>`

Inheritance diagram for IDOMSampledFunction:



Classes

- class Data

  Initialization data.

Public Types

- enum eInterpolationMethod

  Enum for interpolation types.

Public Types inherited from IDOMFunction

- enum eFunctionType

  An enum for the various types of functions.

Public Member Functions

- virtual uint32 getTableDimension (uint32 inputNum) const =0

  Get the dimension of the sample table for a given input number.
- virtual uint8 getBitsPerSample () const =0

  Get the bits per sample for the sample table.
- virtual eInterpolationMethod getInterpolationMethod () const =0

  Get the interpolation method used for sample lookup.
- virtual void getInputEncode (uint32 inputNum, float &low, float &high) const =0

  Get the input encode range for a given input to the function.
- virtual void getOutputDecode (uint32 outputNum, float &low, float &high) const =0

  Get the output range for a given input to the function.
- virtual const CEDLVector< uint8 > & getTableData () const =0

  Get a pointer to the sample table. Values in this table are stored in the same format as described by the PostScript and PDF references. The memory is owned by the instance and must not be modified or freed.

Public Member Functions inherited from IDOMFunction

- virtual eFunctionType getFunctionType () const =0

  Retrieves function type.
- virtual uint32 getNumInputValues () const =0

  Get the number of input values that this function will operate on.
- virtual uint32 getNumOutputValues () const =0

  Get the number of output values that this function will produce.
- virtual void getInputDomain (uint32 inputNum, float &low, float &high) const =0

  Get the input domain for a given input to the function.
- virtual bool getOutputRange (uint32 outputNum, float &low, float &high) const =0

  Get the output range for a given input to the function.
- virtual void evaluate (const float *inputValues, float *outputValues) const =0

  Evaluate the input through the function and return the result.
- virtual void evaluate (const int32 *inputValues, float *outputValues) const =0

  Evaluate the input through the function and return the result.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves class id of IDOM.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.268.1 Detailed Description

Interface for sampled functions. See section 3.9.1 of the PDF 1.7 Reference. Default values are as per described in that reference.

7.268.2 Member Function Documentation

classID()

```
static const CClassID & IDOMSampledFunction::classID ( )  [inline], [static]
```

Retrieves class id of IDOM.

Returns

CClassID class id of the element

getBitsPerSample()

```
virtual uint8 IDOMSampledFunction::getBitsPerSample ( ) const  [pure virtual]
```

Get the bits per sample for the sample table.

Returns

bitsPerSample The bits per sample

getInputEncode()

```
virtual void IDOMSampledFunction::getInputEncode (
            uint32 inputNum,
            float & low,
            float & high ) const  [pure virtual]
```

Get the input encode range for a given input to the function.

Parameters

inputNum	The 0-indexed input number
low	A reference to receive the low encode bound for the given inputNum
high	A reference to receive the high encode bound for the given inputNum

getInterpolationMethod()

```
virtual eInterpolationMethod IDOMSampledFunction::getInterpolationMethod ( ) const  [pure
virtual]
```

Get the interpolation method used for sample lookup.

Returns

eInterpolationMethod The interpolation method

getOutputDecode()

```
virtual void IDOMSampledFunction::getOutputDecode (
            uint32 outputNum,
            float & low,
            float & high ) const  [pure virtual]
```

Get the output range for a given input to the function.

Parameters

outputNum	The 0-indexed output number
low	A reference to receive the low decode bound for the given outputNum
high	A reference to receive the high decode bound for the given outputNum

getTableData()

```
virtual const CEDLVector< uint8 > & IDOMSampledFunction::getTableData ( ) const  [pure virtual]
```

Get a pointer to the sample table. Values in this table are stored in the same format as described by the PostScript and PDF references. The memory is owned by the instance and must not be modified or freed.

Returns

CEDLVector<**uint8**> A reference to the table data.

getTableDimension()

```
virtual uint32 IDOMSampledFunction::getTableDimension (
            uint32 inputNum ) const  [pure virtual]
```

Get the dimension of the sample table for a given input number.

Parameters

inputNum	The 0-indexed input number

Returns

uint32 The dimension

The documentation for this class was generated from the following file:

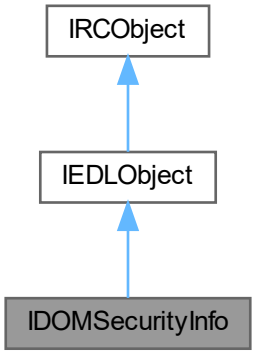
- idomfunction.h

7.269 IDOMSecurityInfo Class Reference

Base DOM security class.

`#include <idomsecurity.h>`

Inheritance diagram for IDOMSecurityInfo:



Additional Inherited Members

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.269.1 Detailed Description

Base DOM security class.

The documentation for this class was generated from the following file:

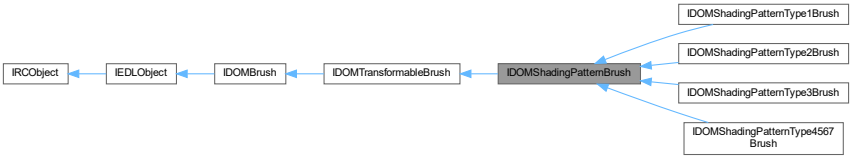
- idomsecurity.h

7.270 IDOMShadingPatternBrush Class Reference

IDOMShadingBrush provides a way of representing a PS style shading pattern.

`#include <idombrush.h>`

Inheritance diagram for IDOMShadingPatternBrush:



Public Member Functions

- virtual uint8 getShadingType () const =0

  Retrieves the shading type.
- virtual bool getBBox (FBox &bBox) const =0

  Retrieves the bounding box for the shade.
- virtual void setBBox (const FBox &bBox)=0

  Sets the bounding box for the shade.
- virtual void setBackgroundColor (const IDOMColorPtr &color)=0

  Sets the background color to use before painting the shade.
- virtual IDOMColorPtr getBackgroundColor () const =0

  Gets the background color to use before painting the shade. If the return is NULL then the background is not painted before applying the shade.
- virtual void setColorSpace (const IDOMColorSpacePtr &colorSpace)=0

  Sets the colorspace to use for painting the shade.
- virtual IDOMColorSpacePtr getColorSpace () const =0

  Gets the colorspace object to be used when painting the shading.
- virtual void setColorSpace (IEDLClassFactory *pFactory, const IDOMColorSpacePtr &colorSpace, eRenderingIntent intent, eBlackPointCompensation bpc)=0

  Set the colorspace of the shading brush, performing color conversion to that target space. This will also convert the background color if present, whereas the alternate setColorSpace() above will not.
- virtual void deIndex (IEDLClassFactory *pFactory)=0

  If the shading brush uses an Indexed color space, reduce it to its base color space.
- virtual void setAntiAlias (bool antiAlias)=0

  Sets anti aliasing flag.
- virtual bool getAntiAlias () const =0

  Gets anti aliasing flag.
- virtual void setFunction (const IDOMFunctionPtr &function)=0

  Sets the shade function.
- virtual IDOMFunctionPtr getFunction () const =0

  Gets the shade function object.

Public Member Functions inherited from IDOMTransformableBrush

- virtual const FMatrix & getRenderTransform () const =0

  Retrieves the render transform matrix.
- virtual void setRenderTransform (const FMatrix &matrix)=0

  Sets the render transform matrix.

Public Member Functions inherited from IDOMBrush

- virtual eBrushType getBrushType () const =0

  Retrieves the type of the brush.
- virtual float getOpacity () const =0

  Retrieves the opacity value of the brush element.
- virtual void setOpacity (float opc)=0

  Sets the opacity value of a brush element.
- virtual IDOMBrushPtr getAdjustedForUseInTransformedNode (IEDLClassFactory *pFactory, const FMatrix &nodeTransform)

  Get a version of this brush adjusted for use inside a node with the given transform.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Public Types inherited from IDOMBrush

- enum eBrushType {
  eSolidColor , eLinearGradient , eRadialGradient , eImage ,
  eMasked , eVisual , eSoftMask , eTilingPattern ,
  eType1ShadingPattern , eType2ShadingPattern , eType3ShadingPattern , eType4567ShadingPattern ,
  eNull }

  Brush type enumeration.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.270.1 Detailed Description

IDOMShadingBrush provides a way of representing a PS style shading pattern.

7.270.2 Member Function Documentation

deIndex()

```
virtual void IDOMShadingPatternBrush::deIndex (
            IEDLClassFactory * pFactory )  [pure virtual]
```

If the shading brush uses an Indexed color space, reduce it to its base color space.

Parameters

pFactory	A pointer to an EDL class factory

getAntiAlias()

```
virtual bool IDOMShadingPatternBrush::getAntiAlias ( ) const  [pure virtual]
```

Gets anti aliasing flag.

Returns

bool The anti-aliasing flag

getBackgroundColor()

```
virtual IDOMColorPtr IDOMShadingPatternBrush::getBackgroundColor ( ) const  [pure virtual]
```

Gets the background color to use before painting the shade. If the return is NULL then the background is not painted before applying the shade.

Returns

IDOMColorPtr The background color, or NULL if not present

getBBox()

```
virtual bool IDOMShadingPatternBrush::getBBox (
            FBox & bBox ) const  [pure virtual]
```

Retrieves the bounding box for the shade.

Parameters

bBox	Reference to receive the bounding box

b bool True if the brush has a bounding box, false otherwise.

getColorSpace()

```
virtual IDOMColorSpacePtr IDOMShadingPatternBrush::getColorSpace ( ) const  [pure virtual]
```

Gets the colorspace object to be used when painting the shading.

Returns

IDOMColorSpacePtr The colorspace

getFunction()

```
virtual IDOMFunctionPtr IDOMShadingPatternBrush::getFunction ( ) const  [pure virtual]
```

Gets the shade function object.

Returns

IDOMFunctionPtr The function, or NULL if no function is present.

getShadingType()

```
virtual uint8 IDOMShadingPatternBrush::getShadingType ( ) const  [pure virtual]
```

Retrieves the shading type.

Returns

int Returns 1 to 7 for the different PS shading types

setAntiAlias()

```
virtual void IDOMShadingPatternBrush::setAntiAlias (
            bool antiAlias )  [pure virtual]
```

Sets anti aliasing flag.

Parameters

antiAlias	New value of anti aliasing flag

setBackgroundColor()

```
virtual void IDOMShadingPatternBrush::setBackgroundColor (
            const IDOMColorPtr & color )  [pure virtual]
```

Sets the background color to use before painting the shade.

Parameters

color	The smart pointer to the color object.

setBBox()

```
virtual void IDOMShadingPatternBrush::setBBox (
            const FBox & bBox )  [pure virtual]
```

Sets the bounding box for the shade.

Parameters

bBox	The desired box

setColorSpace() [1/2]

```
virtual void IDOMShadingPatternBrush::setColorSpace (
            const IDOMColorSpacePtr & colorSpace )  [pure virtual]
```

Sets the colorspace to use for painting the shade.

Parameters

colorSpace	The smart pointer to the colorspace object.

setColorSpace() [2/2]

```
virtual void IDOMShadingPatternBrush::setColorSpace (
            IEDLClassFactory * pFactory,
            const IDOMColorSpacePtr & colorSpace,
            eRenderingIntent intent,
            eBlackPointCompensation bpc )  [pure virtual]
```

Set the colorspace of the shading brush, performing color conversion to that target space. This will also convert the background color if present, whereas the alternate setColorSpace() above will not.

Parameters

pFactory	A pointer to an EDL class factory
colorSpace	The smart pointer to the color space. Must not be a complex or composite color space such as Indexed or DeviceN.
intent	The rendering intent to use for conversion.
bpc	Black point compensation treatment. If in doubt, use eBPCDefault.

setFunction()

```
virtual void IDOMShadingPatternBrush::setFunction (
            const IDOMFunctionPtr & function )  [pure virtual]
```

Sets the shade function.

Parameters

function	Smart pointer to a function object

The documentation for this class was generated from the following file:

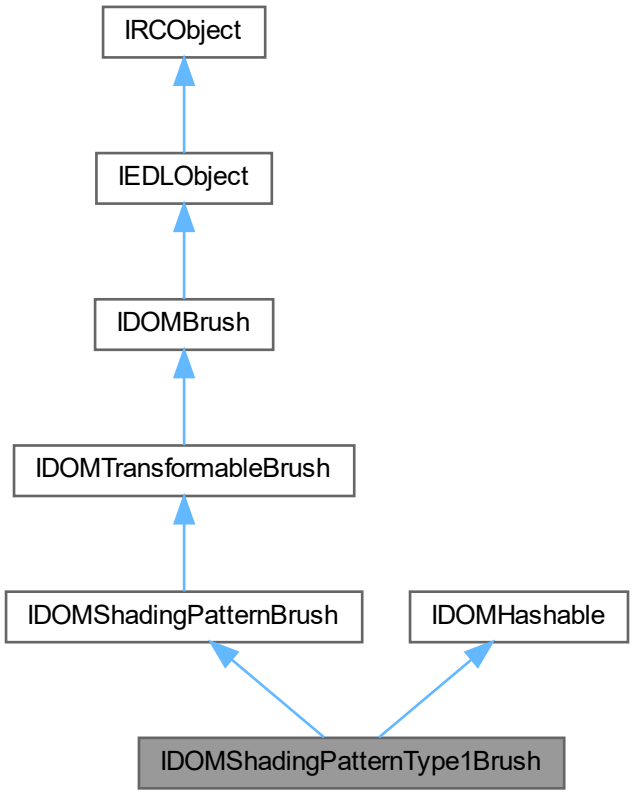
- idombrush.h

7.271 IDOMShadingPatternType1Brush Class Reference

IDOMShadingBrush provides a way of representing a PS style type 1 shading pattern.

`#include <idombrush.h>`

Inheritance diagram for IDOMShadingPatternType1Brush:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual void setDomain (float domain[4])=0

  Sets the domain range.
- virtual const float * getDomain () const =0

  Gets the domain range.
- virtual void setMatrix (const FMatrix &matrix)=0

  Sets the Type 1 shade matrix.
- virtual const FMatrix & getMatrix ()=0

  Gets the Type 1 shade matrix.

Public Member Functions inherited from IDOMShadingPatternBrush

- virtual uint8 getShadingType () const =0

  Retrieves the shading type.
- virtual bool getBBox (FBox &bBox) const =0

  Retrieves the bounding box for the shade.
- virtual void setBBox (const FBox &bBox)=0

  Sets the bounding box for the shade.
- virtual void setBackgroundColor (const IDOMColorPtr &color)=0

  Sets the background color to use before painting the shade.
- virtual IDOMColorPtr getBackgroundColor () const =0

  Gets the background color to use before painting the shade. If the return is NULL then the background is not painted before applying the shade.
- virtual void setColorSpace (const IDOMColorSpacePtr &colorSpace)=0

  Sets the colorspace to use for painting the shade.
- virtual IDOMColorSpacePtr getColorSpace () const =0

  Gets the colorspace object to be used when painting the shading.
- virtual void setColorSpace (IEDLClassFactory *pFactory, const IDOMColorSpacePtr &colorSpace, eRenderingIntent intent, eBlackPointCompensation bpc)=0

  Set the colorspace of the shading brush, performing color conversion to that target space. This will also convert the background color if present, whereas the alternate setColorSpace() above will not.
- virtual void deIndex (IEDLClassFactory *pFactory)=0

  If the shading brush uses an Indexed color space, reduce it to its base color space.
- virtual void setAntiAlias (bool antiAlias)=0

  Sets anti aliasing flag.
- virtual bool getAntiAlias () const =0

  Gets anti aliasing flag.
- virtual void setFunction (const IDOMFunctionPtr &function)=0

  Sets the shade function.
- virtual IDOMFunctionPtr getFunction () const =0

  Gets the shade function object.

Public Member Functions inherited from IDOMTransformableBrush

- virtual const FMatrix & getRenderTransform () const =0

  Retrieves the render transform matrix.
- virtual void setRenderTransform (const FMatrix &matrix)=0

  Sets the render transform matrix.

Public Member Functions inherited from IDOMBrush

- virtual eBrushType getBrushType () const =0

  Retrieves the type of the brush.
- virtual float getOpacity () const =0

  Retrieves the opacity value of the brush element.
- virtual void setOpacity (float opc)=0

  Sets the opacity value of a brush element.
- virtual IDOMBrushPtr getAdjustedForUseInTransformedNode (IEDLClassFactory *pFactory, const FMatrix &nodeTransform)

  Get a version of this brush adjusted for use inside a node with the given transform.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static EDL_API IDOMShadingPatternType1BrushPtr create (IEDLClassFactory *pFactory, const IDOMColor↩SpacePtr &colorSpace, float domain[4], const IDOMFunctionPtr &function, bool hasBBox=false, const FBox &bBox=FBox(), const IDOMColorPtr &background=IDOMColorPtr(), const FMatrix &matrix=FMatrix(), const FMatrix &renderTransform=FMatrix(), bool antiAlias=false, float opacity=1.0f)

  Simplified creator for a type 1 shading pattern brush Throws an IEDLError on failure.
- static const CClassID & classID ()

  Retrieves class id of IDOM.

Additional Inherited Members

Public Types inherited from IDOMBrush

- enum eBrushType {
  eSolidColor , eLinearGradient , eRadialGradient , eImage ,
  eMasked , eVisual , eSoftMask , eTilingPattern ,
  eType1ShadingPattern , eType2ShadingPattern , eType3ShadingPattern , eType4567ShadingPattern ,
  eNull }

  Brush type enumeration.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.271.1 Detailed Description

IDOMShadingBrush provides a way of representing a PS style type 1 shading pattern.

7.271.2 Member Function Documentation

classID()

```
static const CClassID & IDOMShadingPatternType1Brush::classID ( )  [inline], [static]
```

Retrieves class id of IDOM.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMShadingPatternType1BrushPtr IDOMShadingPatternType1Brush::create (
            IEDLClassFactory * pFactory,
            const IDOMColorSpacePtr & colorSpace,
            float domain[4],
            const IDOMFunctionPtr & function,
            bool hasBBox = false,
            const FBox & bBox = FBox(),
            const IDOMColorPtr & background = IDOMColorPtr(),
            const FMatrix & matrix = FMatrix(),
            const FMatrix & renderTransform = FMatrix(),
            bool antiAlias = false,
            float opacity = 1.0f )  [static]
```

Simplified creator for a type 1 shading pattern brush Throws an IEDLError on failure.

Parameters

pFactory	The factory to use.
colorSpace	The color space for the shading pattern
domain	The domain of the matrix x and y axis of the shade
function	The function describing the color of the shade over the range of the domain
hasBBox	true if the bBox parameter should be used
bBox	The shading bounding box
background	The background color to use, or NULL for no background
matrix	The shading matrix to use
renderTransform	The desired render transform.
antiAlias	Whether anti-aliasing should be used for this shade or not
opacity	The opacity to use

Returns

IDOMVisualBrushPtr The new visual brush.

getDomain()

```
virtual const float * IDOMShadingPatternType1Brush::getDomain ( ) const  [pure virtual]
```

Gets the domain range.

Returns

Pointer to an array of four floats representing the domain

getMatrix()

```
virtual const FMatrix & IDOMShadingPatternType1Brush::getMatrix ( )  [pure virtual]
```

Gets the Type 1 shade matrix.

Returns

FMatrix the matrix

setDomain()

```
virtual void IDOMShadingPatternType1Brush::setDomain (
            float domain[4] )  [pure virtual]
```

Sets the domain range.

Parameters

domain	Array of 4 floats

setMatrix()

```
virtual void IDOMShadingPatternType1Brush::setMatrix (
             const FMatrix & matrix )  [pure virtual]
```

Sets the Type 1 shade matrix.

Parameters

matrix	Reference to a matrix object

The documentation for this class was generated from the following file:

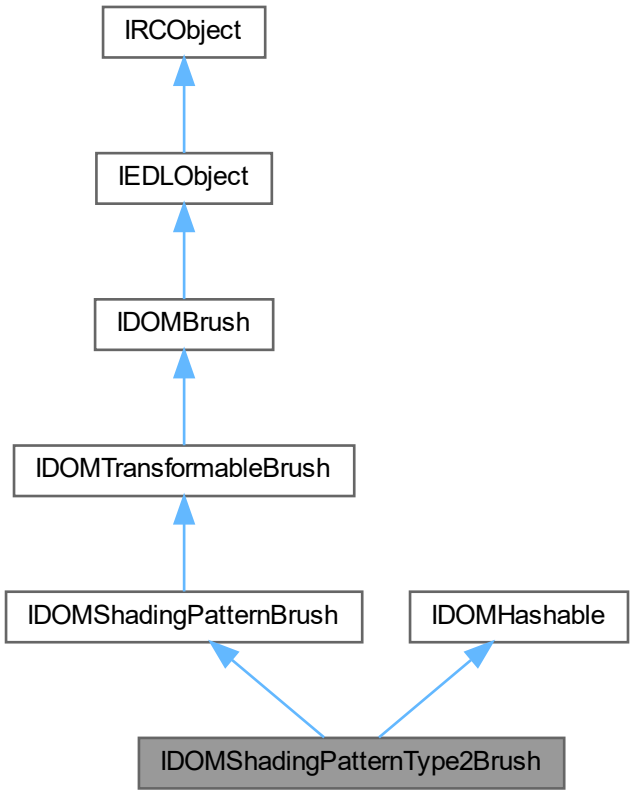
- idombrush.h

7.272 IDOMShadingPatternType2Brush Class Reference

IDOMShadingBrush provides a way of representing a PS style type 2 shading pattern.

```
#include <idombrush.h>
```

Inheritance diagram for IDOMShadingPatternType2Brush:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual void setDomain (float domain[2])=0

  Sets the domain range.
- virtual const float * getDomain () const =0

  Get the domain range.
- virtual void setStartPoint (const FPoint &point)=0

  Sets the start point.
- virtual const FPoint & getStartPoint () const =0

  Get the start point.
- virtual void setEndPoint (const FPoint &point)=0

  Set the end point.
- virtual const FPoint & getEndPoint () const =0

  Get the end point.

- virtual void setExtend (bool extendStart, bool extendEnd)=0

  Sets the shading Extend flag to represent whether or not to extend beyond the start and end points.
- virtual void getExtend (bool &extendStart, bool &extendEnd) const =0

  Gets the shading Extend flag to represent whether or not to extend beyond the start and end points for each axis.
- virtual IDOMBrushPtr getEquivalentSimpleBrush (IEDLClassFactory *pFactory, uint32 maxSamples=255)=0

  Gets an equivalent Linear gradient or Visual brush, which may involve sampling the functions. A Visual brush is required if the shading pattern has a bounding box, in order to apply a clip. This is intended to be used to generate a brush that can be expressed directly in XPS.

Public Member Functions inherited from IDOMShadingPatternBrush

- virtual uint8 getShadingType () const =0

  Retrieves the shading type.
- virtual bool getBBox (FBox &bBox) const =0

  Retrieves the bounding box for the shade.
- virtual void setBBox (const FBox &bBox)=0

  Sets the bounding box for the shade.
- virtual void setBackgroundColor (const IDOMColorPtr &color)=0

  Sets the background color to use before painting the shade.
- virtual IDOMColorPtr getBackgroundColor () const =0

  Gets the background color to use before painting the shade. If the return is NULL then the background is not painted before applying the shade.
- virtual void setColorSpace (const IDOMColorSpacePtr &colorSpace)=0

  Sets the colorspace to use for painting the shade.
- virtual IDOMColorSpacePtr getColorSpace () const =0

  Gets the colorspace object to be used when painting the shading.
- virtual void setColorSpace (IEDLClassFactory *pFactory, const IDOMColorSpacePtr &colorSpace, eRenderingIntent intent, eBlackPointCompensation bpc)=0

  Set the colorspace of the shading brush, performing color conversion to that target space. This will also convert the background color if present, whereas the alternate setColorSpace() above will not.
- virtual void deIndex (IEDLClassFactory *pFactory)=0

  If the shading brush uses an Indexed color space, reduce it to its base color space.
- virtual void setAntiAlias (bool antiAlias)=0

  Sets anti aliasing flag.
- virtual bool getAntiAlias () const =0

  Gets anti aliasing flag.
- virtual void setFunction (const IDOMFunctionPtr &function)=0

  Sets the shade function.
- virtual IDOMFunctionPtr getFunction () const =0

  Gets the shade function object.

Public Member Functions inherited from IDOMTransformableBrush

- virtual const FMatrix & getRenderTransform () const =0

  Retrieves the render transform matrix.
- virtual void setRenderTransform (const FMatrix &matrix)=0

  Sets the render transform matrix.

Public Member Functions inherited from IDOMBrush

- virtual eBrushType getBrushType () const =0

  Retrieves the type of the brush.
- virtual float getOpacity () const =0

  Retrieves the opacity value of the brush element.
- virtual void setOpacity (float opc)=0

  Sets the opacity value of a brush element.
- virtual IDOMBrushPtr getAdjustedForUseInTransformedNode (IEDLClassFactory *pFactory, const FMatrix &nodeTransform)

  Get a version of this brush adjusted for use inside a node with the given transform.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static EDL_API IDOMShadingPatternType2BrushPtr create (IEDLClassFactory *pFactory, const FPoint &startPoint, const FPoint &endPoint, const IDOMColorSpacePtr &colorSpace, float domain[2], const IDOMFunctionPtr &function, bool extendStart=false, bool extendEnd=false, bool hasBBox=false, const FBox &bBox=FBox(), const IDOMColorPtr &background=IDOMColorPtr(), const FMatrix &renderTransform=FMatrix(), bool antiAlias=false, float opacity=1.0f)

  Simplified creator for a type 2 shading pattern brush Throws an IEDLError on failure.
- static const CClassID & classID ()

  Retrieves class id of IDOM.

Additional Inherited Members

Public Types inherited from IDOMBrush

- enum eBrushType {
  eSolidColor , eLinearGradient , eRadialGradient , eImage ,
  eMasked , eVisual , eSoftMask , eTilingPattern ,
  eType1ShadingPattern , eType2ShadingPattern , eType3ShadingPattern , eType4567ShadingPattern ,
  eNull }

  Brush type enumeration.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.272.1 Detailed Description

IDOMShadingBrush provides a way of representing a PS style type 2 shading pattern.

7.272.2 Member Function Documentation

classID()

```
static const CClassID & IDOMShadingPatternType2Brush::classID ( )  [inline], [static]
```

Retrieves class id of IDOM.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMShadingPatternType2BrushPtr IDOMShadingPatternType2Brush::create (
            IEDLClassFactory * pFactory,
            const FPoint & startPoint,
            const FPoint & endPoint,
            const IDOMColorSpacePtr & colorSpace,
            float domain[2],
            const IDOMFunctionPtr & function,
            bool extendStart = false,
            bool extendEnd = false,
            bool hasBBox = false,
            const FBox & bBox = FBox(),
            const IDOMColorPtr & background = IDOMColorPtr(),
            const FMatrix & renderTransform = FMatrix(),
            bool antiAlias = false,
            float opacity = 1.0f )  [static]
```

Simplified creator for a type 2 shading pattern brush Throws an IEDLError on failure.

Parameters

pFactory	The factory to use.
startPoint	The starting point for the shade
endPoint	The end point for the shade
colorSpace	The color space for the shading pattern
domain	The domain of the shading variable
function	The function describing the color of the shade over the range of the domain
extendStart	True if the shade should be extended before the start point (as per linear gradient pad mode)
extendEnd	True if the shade should be extended beyond the end point (as per linear gradient pad mode)
hasBBox	true if the bBox parameter should be used
bBox	The shading bounding box
background	The background color to use, or NULL for no background
renderTransform	The desired render transform.
antiAlias	Whether anti-aliasing should be used for this shade or not
opacity	The opacity to use

Returns

IDOMVisualBrushPtr The new visual brush.

getDomain()

```
virtual const float * IDOMShadingPatternType2Brush::getDomain ( ) const  [pure virtual]
```

Get the domain range.

Returns

Pointer to an array of two floats representing the domain

getEndPoint()

```
virtual const FPoint & IDOMShadingPatternType2Brush::getEndPoint ( ) const  [pure virtual]
```

Get the end point.

Returns

FPoint The end point

getEquivalentSimpleBrush()

```
virtual IDOMBrushPtr IDOMShadingPatternType2Brush::getEquivalentSimpleBrush (
            IEDLClassFactory * pFactory,
            uint32 maxSamples = 255 )  [pure virtual]
```

Gets an equivalent Linear gradient or Visual brush, which may involve sampling the functions. A Visual brush is required if the shading pattern has a bounding box, in order to apply a clip. This is intended to be used to generate a brush that can be expressed directly in XPS.

Parameters

pFactory	A pointer to an EDL class factory
maxSamples	The maximum number of samples that should be taken from the function to generate gradient stops.

Returns

IDOMBrushPtr The simplified brush. This brush will be cached, so do not edit; make a clone if the returned brush needs to be changed.

getExtend()

```
virtual void IDOMShadingPatternType2Brush::getExtend (
            bool & extendStart,
            bool & extendEnd ) const  [pure virtual]
```

Gets the shading Extend flag to represent whether or not to extend beyond the start and end points for each axis.

Parameters

extendStart	Reference to bool parameter representing the flags for the start point
extendEnd	Reference to bool parameter representing the flags for the end point

getStartPoint()

```
virtual const FPoint & IDOMShadingPatternType2Brush::getStartPoint ( ) const  [pure virtual]
```

Get the start point.

Returns

FPoint The start point

setDomain()

```
virtual void IDOMShadingPatternType2Brush::setDomain (
            float domain[2] )  [pure virtual]
```

Sets the domain range.

Parameters

domain	Array of 2 floats

setEndPoint()

```
virtual void IDOMShadingPatternType2Brush::setEndPoint (
            const FPoint & point )  [pure virtual]
```

Set the end point.

Parameters

point	The end point

setExtend()

```
virtual void IDOMShadingPatternType2Brush::setExtend (
            bool extendStart,
            bool extendEnd )  [pure virtual]
```

Sets the shading Extend flag to represent whether or not to extend beyond the start and end points.

Parameters

extendStart	parameter representing the flags for the start point
extendEnd	parameter representing the flags for the end point

setStartPoint()

```
virtual void IDOMShadingPatternType2Brush::setStartPoint (
            const FPoint & point )  [pure virtual]
```

Sets the start point.

Parameters

point	The start point

The documentation for this class was generated from the following file:

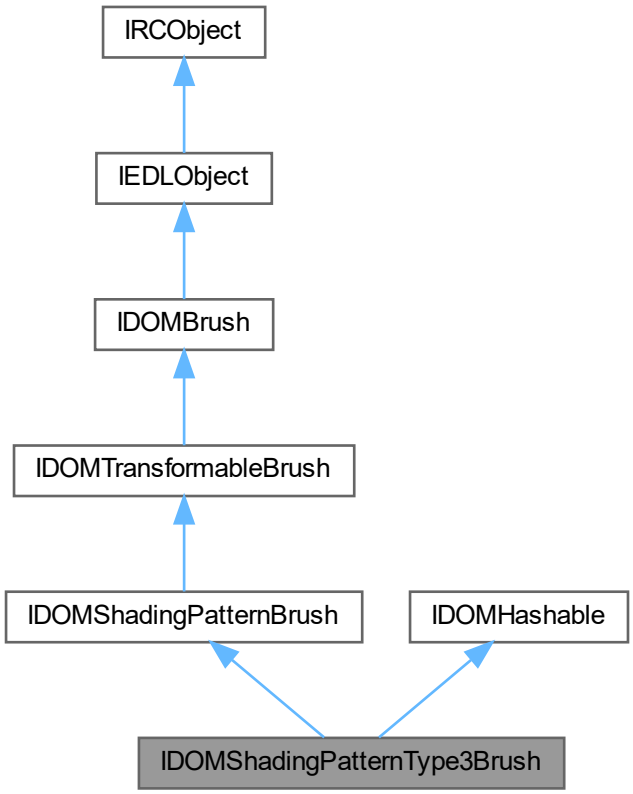
- idombrush.h

7.273 IDOMShadingPatternType3Brush Class Reference

IDOMShadingPatternType3Brush provides a way of representing a PS style type 2 shading pattern.

```
#include <idombrush.h>
```

Inheritance diagram for IDOMShadingPatternType3Brush:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual void setDomain (float domain[2])=0

  Sets the domain range.
- virtual const float * getDomain () const =0

  Get the domain range.
- virtual void setStartCircleCenter (const FPoint &point)=0

  Sets the center of the starting circle.
- virtual const FPoint & getStartCircleCenter () const =0

  Get the center of the starting circle.
- virtual void setStartCircleRadius (float radius)=0

  Set the radius of the starting circle.
- virtual float getStartCircleRadius () const =0

  Get the radius of the starting circle.

- virtual void setEndCircleCenter (const FPoint &point)=0

  Sets the center of the ending circle.
- virtual const FPoint & getEndCircleCenter () const =0

  Get the center of the ending circle.
- virtual void setEndCircleRadius (float radius)=0

  Set the radius of the ending circle.
- virtual float getEndCircleRadius () const =0

  Get the radius of the ending circle.
- virtual void setExtend (bool extendStart, bool extendEnd)=0

  Sets the shading Extend flag to represent whether or not to extend beyond the start and end circles.
- virtual void getExtend (bool &extendStart, bool &extendEnd) const =0

  Gets the shading Extend flag to represent whether or not to extend beyond the start and end points for each axis.
- virtual IDOMBrushPtr getEquivalentSimpleBrush (IEDLClassFactory *pFactory, uint32 maxSamples=255)=0

  Gets an equivalent Radial gradient or Visual brush, which may involve sampling the functions. A Visual brush is required if the shading pattern has a bounding box, in order to apply a clip. This is intended to be used to generate a brush that can be expressed directly in XPS. This is not possible for all Type 3 shading brushes.

Public Member Functions inherited from IDOMShadingPatternBrush

- virtual uint8 getShadingType () const =0

  Retrieves the shading type.
- virtual bool getBBox (FBox &bBox) const =0

  Retrieves the bounding box for the shade.
- virtual void setBBox (const FBox &bBox)=0

  Sets the bounding box for the shade.
- virtual void setBackgroundColor (const IDOMColorPtr &color)=0

  Sets the background color to use before painting the shade.
- virtual IDOMColorPtr getBackgroundColor () const =0

  Gets the background color to use before painting the shade. If the return is NULL then the background is not painted before applying the shade.
- virtual void setColorSpace (const IDOMColorSpacePtr &colorSpace)=0

  Sets the colorspace to use for painting the shade.
- virtual IDOMColorSpacePtr getColorSpace () const =0

  Gets the colorspace object to be used when painting the shading.
- virtual void setColorSpace (IEDLClassFactory *pFactory, const IDOMColorSpacePtr &colorSpace, eRenderingIntent intent, eBlackPointCompensation bpc)=0

  Set the colorspace of the shading brush, performing color conversion to that target space. This will also convert the background color if present, whereas the alternate setColorSpace() above will not.
- virtual void deIndex (IEDLClassFactory *pFactory)=0

  If the shading brush uses an Indexed color space, reduce it to its base color space.
- virtual void setAntiAlias (bool antiAlias)=0

  Sets anti aliasing flag.
- virtual bool getAntiAlias () const =0

  Gets anti aliasing flag.
- virtual void setFunction (const IDOMFunctionPtr &function)=0

  Sets the shade function.
- virtual IDOMFunctionPtr getFunction () const =0

  Gets the shade function object.

Public Member Functions inherited from IDOMTransformableBrush

- virtual const FMatrix & getRenderTransform () const =0

  Retrieves the render transform matrix.
- virtual void setRenderTransform (const FMatrix &matrix)=0

  Sets the render transform matrix.

Public Member Functions inherited from IDOMBrush

- virtual eBrushType getBrushType () const =0

  Retrieves the type of the brush.
- virtual float getOpacity () const =0

  Retrieves the opacity value of the brush element.
- virtual void setOpacity (float opc)=0

  Sets the opacity value of a brush element.
- virtual IDOMBrushPtr getAdjustedForUseInTransformedNode (IEDLClassFactory *pFactory, const FMatrix &nodeTransform)

  Get a version of this brush adjusted for use inside a node with the given transform.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static EDL_API IDOMShadingPatternType3BrushPtr create (IEDLClassFactory *pFactory, const FPoint &startCircleCenter, float startCircleRadius, const FPoint &endCircleCenter, float endCircleRadius, const IDOMColorSpacePtr &colorSpace, float domain[2], const IDOMFunctionPtr &function, bool extend↩ Start=false, bool extendEnd=false, bool hasBBox=false, const FBox &bBox=FBox(), const IDOMColor↩ Ptr &background=IDOMColorPtr(), const FMatrix &renderTransform=FMatrix(), bool antiAlias=false, float opacity=1.0f)

  Simplified creator for a type 3 shading pattern brush Throws an IEDLError on failure.
- static const CClassID & classID ()

  Retrieves class id of IDOM.

Additional Inherited Members

Public Types inherited from IDOMBrush

- enum eBrushType {
  eSolidColor , eLinearGradient , eRadialGradient , eImage ,
  eMasked , eVisual , eSoftMask , eTilingPattern ,
  eType1ShadingPattern , eType2ShadingPattern , eType3ShadingPattern , eType4567ShadingPattern ,
  eNull }

  Brush type enumeration.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.273.1 Detailed Description

IDOMShadingPatternType3Brush provides a way of representing a PS style type 2 shading pattern.

7.273.2 Member Function Documentation

classID()

```
static const CClassID & IDOMShadingPatternType3Brush::classID ( )  [inline], [static]
```

Retrieves class id of IDOM.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMShadingPatternType3BrushPtr IDOMShadingPatternType3Brush::create (
            IEDLClassFactory * pFactory,
            const FPoint & startCircleCenter,
            float startCircleRadius,
            const FPoint & endCircleCenter,
            float endCircleRadius,
            const IDOMColorSpacePtr & colorSpace,
            float domain[2],
            const IDOMFunctionPtr & function,
            bool extendStart = false,
            bool extendEnd = false,
            bool hasBBox = false,
            const FBox & bBox = FBox(),
            const IDOMColorPtr & background = IDOMColorPtr(),
            const FMatrix & renderTransform = FMatrix(),
            bool antiAlias = false,
            float opacity = 1.0f )  [static]
```

Simplified creator for a type 3 shading pattern brush Throws an IEDLError on failure.

Parameters

pFactory	The factory to use.
startCircleCenter	The center of the starting circle for the shade
startCircleRadius	The radius of the starting circle for the shade
endCircleCenter	The center of the starting circle for the shade
endCircleRadius	The radius of the starting circle for the shade
colorSpace	The color space for the shading pattern
domain	The domain of the shading variable
function	The function describing the color of the shade over the range of the domain
extendStart	True if the shade should be extended before the starting circle
extendEnd	True if the shade should be extended beyond the endinc circle
hasBBox	true if the bBox parameter should be used
bBox	The shading bounding box
background	The background color to use, or NULL for no background
renderTransform	The desired render transform.
antiAlias	Whether anti-aliasing should be used for this shade or not
opacity	The opacity to use

Returns

IDOMVisualBrushPtr The new visual brush.

getDomain()

```
virtual const float * IDOMShadingPatternType3Brush::getDomain ( ) const  [pure virtual]
```

Get the domain range.

Returns

Pointer to an array of two floats representing the domain

getEndCircleCenter()

```
virtual const FPoint & IDOMShadingPatternType3Brush::getEndCircleCenter ( ) const  [pure virtual]
```

Get the center of the ending circle.

Returns

FPoint The ending circle center

getEndCircleRadius()

```
virtual float IDOMShadingPatternType3Brush::getEndCircleRadius ( ) const  [pure virtual]
```

Get the radius of the ending circle.

Returns

float The ending circle radius

getEquivalentSimpleBrush()

```
virtual IDOMBrushPtr IDOMShadingPatternType3Brush::getEquivalentSimpleBrush (
            IEDLClassFactory * pFactory,
            uint32 maxSamples = 255 )  [pure virtual]
```

Gets an equivalent Radial gradient or Visual brush, which may involve sampling the functions. A Visual brush is required if the shading pattern has a bounding box, in order to apply a clip. This is intended to be used to generate a brush that can be expressed directly in XPS. This is not possible for all Type 3 shading brushes.

Parameters

pFactory	A pointer to an EDL class factory
maxSamples	The maximum number of samples that should be taken from the function to generate gradient stops.

Returns

IDOMBrushPtr The simplified brush, or NULL if this brush cannot be represented as a simpler brush. This brush will be cached, so do not edit; make a clone if the returned brush needs to be changed.

getExtend()

```
virtual void IDOMShadingPatternType3Brush::getExtend (
            bool & extendStart,
            bool & extendEnd ) const  [pure virtual]
```

Gets the shading Extend flag to represent whether or not to extend beyond the start and end points for each axis.

Parameters

extendStart	Reference to boolean parameter representing the flags for the starting circle
extendEnd	Reference to boolean parameter representing the flags for the ending circle

getStartCircleCenter()

```
virtual const FPoint & IDOMShadingPatternType3Brush::getStartCircleCenter ( ) const  [pure
virtual]
```

Get the center of the starting circle.

Returns

FPoint The starting circle center

getStartCircleRadius()

```
virtual float IDOMShadingPatternType3Brush::getStartCircleRadius ( ) const  [pure virtual]
```

Get the radius of the starting circle.

Returns

float The staring circle radius

setDomain()

```
virtual void IDOMShadingPatternType3Brush::setDomain (
            float domain[2] )  [pure virtual]
```

Sets the domain range.

Parameters

domain	Array of 2 floats

setEndCircleCenter()

```
virtual void IDOMShadingPatternType3Brush::setEndCircleCenter (
            const FPoint & point )  [pure virtual]
```

Sets the center of the ending circle.

Parameters

point	The ending circle center

setEndCircleRadius()

```
virtual void IDOMShadingPatternType3Brush::setEndCircleRadius (
            float radius )  [pure virtual]
```

Set the radius of the ending circle.

Parameters

radius	The ending circle radius

setExtend()

```
virtual void IDOMShadingPatternType3Brush::setExtend (
            bool extendStart,
            bool extendEnd )  [pure virtual]
```

Sets the shading Extend flag to represent whether or not to extend beyond the start and end circles.

Parameters

extendStart	Boolean parameter representing the flags for the starting circle
extendEnd	Boolean parameter representing the flags for the ending circle

setStartCircleCenter()

```
virtual void IDOMShadingPatternType3Brush::setStartCircleCenter (
            const FPoint & point )  [pure virtual]
```

Sets the center of the starting circle.

Parameters

point	The starting circle center

setStartCircleRadius()

```
virtual void IDOMShadingPatternType3Brush::setStartCircleRadius (
            float radius )  [pure virtual]
```

Set the radius of the starting circle.

Parameters

radius	The staring circle radius

The documentation for this class was generated from the following file:

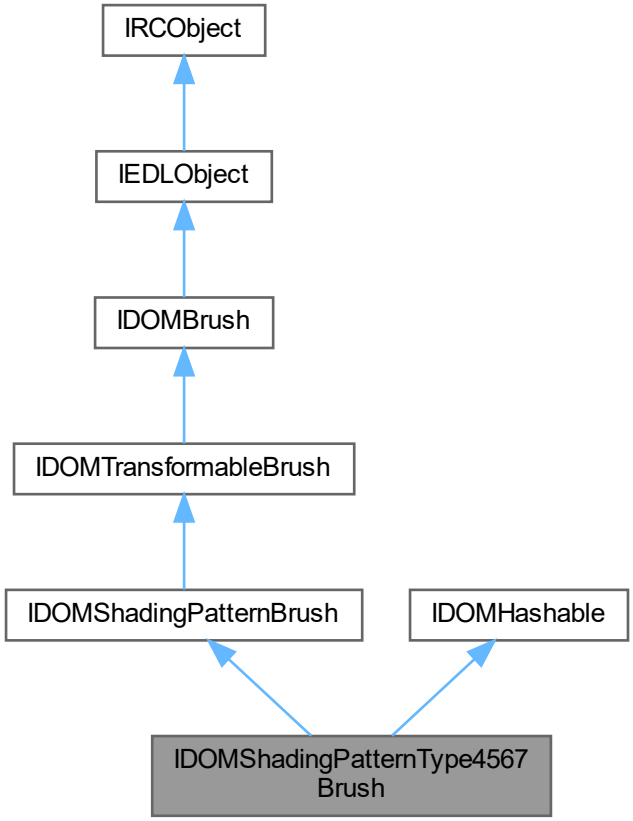
- idombrush.h

7.274 IDOMShadingPatternType4567Brush Class Reference

IDOMShadingPatternType4567Brush provides a way of representing a PS style type 4 shading pattern.

`#include <idombrush.h>`

Inheritance diagram for IDOMShadingPatternType4567Brush:



Classes

- class CMeshEntry

  A entry in the shading pattern's mesh. The interpretation of each entry depends on the shading type, and potentially on per-entry flags. Please see the PDF specification for details.
- class Data

  Initialization data.

Public Member Functions

- virtual void setShadingType (uint8 shadingType)=0

  Sets the shading type.
- virtual void setDataSource (const JawsMako::IPDFStringPtr &dataSource)=0

  Sets the data source property.
- virtual JawsMako::IPDFStringPtr getDataSource () const =0

  Gets the data source property.
- virtual void setBitsPerCoordinate (uint8 bitsPerCoordinate)=0

  Sets the bits per coordinate.
- virtual uint8 getBitsPerCoordinate () const =0

  Gets the bits per coordinate parameter.
- virtual void setBitsPerComponent (uint8 bitsPerComponent)=0

  Sets the bits per component.
- virtual uint8 getBitsPerComponent () const =0

  Gets the bits per component parameter.
- virtual void setBitsPerFlag (uint8 bitsPerFlag)=0

  Sets the bits per flag.
- virtual uint8 getBitsPerFlag () const =0

  Gets the bits per flag parameter.
- virtual void setVerticesPerRow (int32 verticesPerRow)=0

  Sets the vertices per row flag.
- virtual int32 getVerticesPerRow () const =0

  Gets the vertices per row parameter.
- virtual void setDecode (const CEDLVector< float > &decode)=0

  Sets the decode array.
- virtual const CEDLVector< float > & getDecode () const =0

  Gets the decode array.
- virtual CMeshEntryVect getMeshEntries () const =0

  Get the vector of mesh entries that describe the appearance of this shading pattern. This will return a copy that may be edited freely.
- virtual void setMeshEntries (const CMeshEntryVect &meshEntries)=0

  Set the vector of mesh entries that describe the appearance of this shading pattern.

Public Member Functions inherited from IDOMShadingPatternBrush

- virtual uint8 getShadingType () const =0

  Retrieves the shading type.
- virtual bool getBBox (FBox &bBox) const =0

  Retrieves the bounding box for the shade.
- virtual void setBBox (const FBox &bBox)=0

  Sets the bounding box for the shade.
- virtual void setBackgroundColor (const IDOMColorPtr &color)=0

  Sets the background color to use before painting the shade.
- virtual IDOMColorPtr getBackgroundColor () const =0

  Gets the background color to use before painting the shade. If the return is NULL then the background is not painted before applying the shade.
- virtual void setColorSpace (const IDOMColorSpacePtr &colorSpace)=0

  Sets the colorspace to use for painting the shade.
- virtual IDOMColorSpacePtr getColorSpace () const =0

  Gets the colorspace object to be used when painting the shading.

- virtual void setColorSpace (IEDLClassFactory *pFactory, const IDOMColorSpacePtr &colorSpace, eRenderingIntent intent, eBlackPointCompensation bpc)=0

  Set the colorspace of the shading brush, performing color conversion to that target space. This will also convert the background color if present, whereas the alternate setColorSpace() above will not.
- virtual void deIndex (IEDLClassFactory *pFactory)=0

  If the shading brush uses an Indexed color space, reduce it to its base color space.
- virtual void setAntiAlias (bool antiAlias)=0

  Sets anti aliasing flag.
- virtual bool getAntiAlias () const =0

  Gets anti aliasing flag.
- virtual void setFunction (const IDOMFunctionPtr &function)=0

  Sets the shade function.
- virtual IDOMFunctionPtr getFunction () const =0

  Gets the shade function object.

Public Member Functions inherited from IDOMTransformableBrush

- virtual const FMatrix & getRenderTransform () const =0

  Retrieves the render transform matrix.
- virtual void setRenderTransform (const FMatrix &matrix)=0

  Sets the render transform matrix.

Public Member Functions inherited from IDOMBrush

- virtual eBrushType getBrushType () const =0

  Retrieves the type of the brush.
- virtual float getOpacity () const =0

  Retrieves the opacity value of the brush element.
- virtual void setOpacity (float opc)=0

  Sets the opacity value of a brush element.
- virtual IDOMBrushPtr getAdjustedForUseInTransformedNode (IEDLClassFactory *pFactory, const FMatrix &nodeTransform)

  Get a version of this brush adjusted for use inside a node with the given transform.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static EDL_API IDOMShadingPatternType4567BrushPtr create (IEDLClassFactory *pFactory, uint8 shadingType, const IDOMColorSpacePtr &colorSpace, const JawsMako::IPDFStringPtr &dataSource, const uint8 bitsPerCoordinate=0, const uint8 bitsPerComponent=0, const uint8 bitsPerFlag=0, const int32 verticesPerRow=0, const CEDLVector< float > &decode=CEDLVector< float >(), const IDOMFunction↩Ptr &function=IDOMFunctionPtr(), bool hasBBox=false, const FBox &bBox=FBox(), const IDOMColor↩Ptr &background=IDOMColorPtr(), const FMatrix &renderTransform=FMatrix(), bool antiAlias=false, float opacity=1.0f)

  Simplified creator for a type 4,5,6 or 7 shading pattern brush Throws an IEDLError on failure.
- static EDL_API IDOMShadingPatternType4567BrushPtr create (IEDLClassFactory *pFactory, uint8 shadingType, const IDOMColorSpacePtr &colorSpace, const IDOMFunctionPtr &function, const CMesh↩EntryVect &meshEntries, const uint8 bitsPerCoordinate=16, const uint8 bitsPerComponent=8, const uint8 bitsPerFlag=8, const int32 verticesPerRow=0, const CEDLVector< float > &decode=CEDLVector< float >(), bool hasBBox=false, const FBox &bBox=FBox(), const IDOMColorPtr &background=IDOMColorPtr(), const FMatrix &renderTransform=FMatrix(), bool antiAlias=false, float opacity=1.0f)

  Simplified creator for a type 4,5,6 or 7 shading pattern brush, providing a mesh as data source Throws an IEDLError on failure. When using a mesh source, the bitsPerComponent is locked to 8.
- static const CClassID & classID ()

  Retrieves class id of IDOM.

Additional Inherited Members

Public Types inherited from IDOMBrush

- enum eBrushType {
  eSolidColor , eLinearGradient , eRadialGradient , eImage ,
  eMasked , eVisual , eSoftMask , eTilingPattern ,
  eType1ShadingPattern , eType2ShadingPattern , eType3ShadingPattern , eType4567ShadingPattern ,
  eNull }

  Brush type enumeration.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.274.1 Detailed Description

IDOMShadingPatternType4567Brush provides a way of representing a PS style type 4 shading pattern.

7.274.2 Member Function Documentation

classID()

```
static const CClassID & IDOMShadingPatternType4567Brush::classID ( )  [inline], [static]
```

Retrieves class id of IDOM.

Returns

CClassID class id of the element

create() [1/2]

```
static EDL_API IDOMShadingPatternType4567BrushPtr IDOMShadingPatternType4567Brush::create (
            IEDLClassFactory * pFactory,
            uint8 shadingType,
            const IDOMColorSpacePtr & colorSpace,
            const IDOMFunctionPtr & function,
            const CMeshEntryVect & meshEntries,
            const uint8 bitsPerCoordinate = 16,
            const uint8 bitsPerComponent = 8,
            const uint8 bitsPerFlag = 8,
            const int32 verticesPerRow = 0,
            const CEDLVector< float > & decode = CEDLVector< float >(),
            bool hasBBox = false,
            const FBox & bBox = FBox(),
            const IDOMColorPtr & background = IDOMColorPtr(),
            const FMatrix & renderTransform = FMatrix(),
            bool antiAlias = false,
            float opacity = 1.0f )  [static]
```

Simplified creator for a type 4,5,6 or 7 shading pattern brush, providing a mesh as data source Throws an IEDLError on failure. When using a mesh source, the bitsPerComponent is locked to 8.

Parameters

pFactory	The factory to use.
shadingType	The shading pattern type.
colorSpace	The color space for the shading pattern.
function	If the mesh entries specify a parametric value instead of a color, the function that produces the final output color. If the mesh specifies actual colors, this should be null.
meshEntries	The mesh entries to use to construct the shade.
bitsPerCoordinate	The number of bits used to represent each coordinate.
bitsPerComponent	The number of bits used to the function or color samples at each coordinate. If the color space is indexed, and the colors are stored in the mesh, then this is ignored and the depth will be forced to 8 bits per component.
bitsPerFlag	The number of bits used to represent the edge flag for each patch or vertex (depends on shade type). Not used for all shade types.
verticesPerRow	The vertices per row parameter. Not used for all shade types.
decode	A vector representing the decode array of the shade. if the data source provides vertex color information.
hasBBox	true if the bBox parameter should be used

Parameters

bBox	The shading bounding box
background	The background color to use, or NULL for no background
renderTransform	The desired render transform.
antiAlias	Whether anti-aliasing should be used for this shade or not
opacity	The opacity to use

Returns

IDOMVisualBrushPtr The new visual brush.

create() [2/2]

```
static EDL_API IDOMShadingPatternType4567BrushPtr IDOMShadingPatternType4567Brush::create (
            IEDLClassFactory * pFactory,
            uint8 shadingType,
            const IDOMColorSpacePtr & colorSpace,
            const JawsMako::IPDFStringPtr & dataSource,
            const uint8 bitsPerCoordinate = 0,
            const uint8 bitsPerComponent = 0,
            const uint8 bitsPerFlag = 0,
            const int32 verticesPerRow = 0,
            const CEDLVector< float > & decode = CEDLVector< float >(),
            const IDOMFunctionPtr & function = IDOMFunctionPtr(),
            bool hasBBox = false,
            const FBox & bBox = FBox(),
            const IDOMColorPtr & background = IDOMColorPtr(),
            const FMatrix & renderTransform = FMatrix(),
            bool antiAlias = false,
            float opacity = 1.0f )  [static]
```

Simplified creator for a type 4,5,6 or 7 shading pattern brush Throws an IEDLError on failure.

Parameters

pFactory	The factory to use.
shadingType	The shading pattern type.
colorSpace	The color space for the shading pattern.
dataSource	The data source used for vertex/patch data.
bitsPerCoordinate	The number of bits used to represent each coordinate.
bitsPerComponent	The number of bits used to represent each color component.
bitsPerFlag	The number of bits used to represent the edge flag for each patch or vertex (depends on shade type). Not used for all shade types.
verticesPerRow	The vertices per row parameter. Not used for all shade types.
decode	A vector representing the decode array of the shade.
function	The optional shading function describing the color depending on the parametric shade variable. Must be NULL if the data source provides vertex color information.
hasBBox	true if the bBox parameter should be used.
bBox	The shading bounding box.
background	The background color to use, or NULL for no background.
renderTransform	The desired render transform.
antiAlias	Whether anti-aliasing should be used for this shade or not.
opacity	The opacity to use.

Returns

IDOMVisualBrushPtr The new visual brush.

getBitsPerComponent()

```
virtual uint8 IDOMShadingPatternType4567Brush::getBitsPerComponent ( ) const  [pure virtual]
```

Gets the bits per component parameter.

Returns

uint8 The bits per component parameter

getBitsPerCoordinate()

```
virtual uint8 IDOMShadingPatternType4567Brush::getBitsPerCoordinate ( ) const  [pure virtual]
```

Gets the bits per coordinate parameter.

Returns

uint8 The bits per coordinate parameter

getBitsPerFlag()

```
virtual uint8 IDOMShadingPatternType4567Brush::getBitsPerFlag ( ) const  [pure virtual]
```

Gets the bits per flag parameter.

Returns

uint8 The bits per flag

getDataSource()

```
virtual JawsMako::IPDFStringPtr IDOMShadingPatternType4567Brush::getDataSource ( ) const  [pure
virtual]
```

Gets the data source property.

Returns

IPDFStringPtr The data source

getDecode()

```
virtual const CEDLVector< float > & IDOMShadingPatternType4567Brush::getDecode ( ) const
[pure virtual]
```

Gets the decode array.

Returns

CEDLVector<**float**> The decode array

getMeshEntries()

```
virtual CMeshEntryVect IDOMShadingPatternType4567Brush::getMeshEntries ( ) const  [pure virtual]
```

Get the vector of mesh entries that describe the appearance of this shading pattern. This will return a copy that may be edited freely.

Returns

CMeshEntryVect The mesh entries.

getVerticesPerRow()

```
virtual int32 IDOMShadingPatternType4567Brush::getVerticesPerRow ( ) const  [pure virtual]
```

Gets the vertices per row parameter.

Returns

int32 The vertices per row parameter

setBitsPerComponent()

```
virtual void IDOMShadingPatternType4567Brush::setBitsPerComponent (
            uint8 bitsPerComponent )  [pure virtual]
```

Sets the bits per component.

Parameters

bitsPerComponent	Bits per component value to set

setBitsPerCoordinate()

```
virtual void IDOMShadingPatternType4567Brush::setBitsPerCoordinate (
            uint8 bitsPerCoordinate )  [pure virtual]
```

Sets the bits per coordinate.

Parameters

bitsPerCoordinate	Bits per coordinate value to set

setBitsPerFlag()

```
virtual void IDOMShadingPatternType4567Brush::setBitsPerFlag (
            uint8 bitsPerFlag )  [pure virtual]
```

Sets the bits per flag.

Parameters

bitsPerFlag	Bits per flag value to set

setDataSource()

```
virtual void IDOMShadingPatternType4567Brush::setDataSource (
            const JawsMako::IPDFStringPtr & dataSource )  [pure virtual]
```

Sets the data source property.

Parameters

dataSource	A reference to the data source encoded as a PDF string object

setDecode()

```
virtual void IDOMShadingPatternType4567Brush::setDecode (
            const CEDLVector< float > & decode )  [pure virtual]
```

Sets the decode array.

Parameters

decode	The decode array

setMeshEntries()

```
virtual void IDOMShadingPatternType4567Brush::setMeshEntries (
            const CMeshEntryVect & meshEntries )  [pure virtual]
```

Set the vector of mesh entries that describe the appearance of this shading pattern.

Parameters

meshEntries	The mesh entries.

setShadingType()

```
virtual void IDOMShadingPatternType4567Brush::setShadingType (
            uint8 shadingType )  [pure virtual]
```

Sets the shading type.

Parameters

shadingType	An integer which represents the PS style shading type (one of 4, 5, 6 or 7)

setVerticesPerRow()

```
virtual void IDOMShadingPatternType4567Brush::setVerticesPerRow (
            int32 verticesPerRow )  [pure virtual]
```

Sets the vertices per row flag.

Parameters

verticesPerRow	Vertices per row value to set. Must be greater than two.

The documentation for this class was generated from the following file:

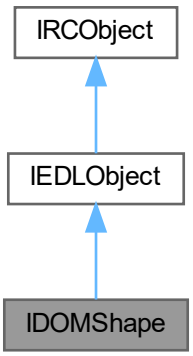
- idombrush.h

7.275 IDOMShape Class Reference

Interface to an IDOMShape.

```
#include <idomshape.h>
```

Inheritance diagram for IDOMShape:



Classes

- class CShapeDetails

  Provides a view into the regions that comprise the shape. A region consists of a series of spans, each representing a series of rectangles that share the same y span.
- class Data

  Initialization data.

Public Member Functions

- virtual IDOMImagePtr getAsImage (const ISessionPtr &session, bool bUseTempFileForImage) const =0

  Return in `image` an 1-bit-per-pixel DeviceGray image representation of this shape.
- virtual IDOMPathGeometryPtr getAsGeometry (IEDLClassFactory *factory) const =0

  Get the shape as path geometry.
- virtual FRect getBounds () const =0

  Get the bounds (in pixels) of the shape.
- virtual float getResolution () const =0

  Get the resolution of the shape.
- virtual bool getIsEmpty () const =0

  Detect an empty shape.
- virtual bool getIsRect () const =0

  Detect if a shape is a rectangle.
- virtual bool isEqualTo (const IDOMShapePtr &ptrShape) const =0

  Check this shape for equality to another shape.
- virtual void unite (const IDOMShapePtr &ptrShape)=0

  Unite this shape with another.
- virtual void intersect (const IDOMShapePtr &ptrShape)=0

  Reduce this shape to its intersection with another shape.
- virtual void difference (const IDOMShapePtr &ptrShape)=0

  Reduce this shape by subtracting another shape.
- virtual void offset (int32 offsetX, int32 offsetY)=0

Offset the shape by the given numbers of pixels.

- virtual bool intersects (const IDOMShapePtr &ptrShape) const =0

  Detect if this shape intersects with another.
- virtual bool completelyContainsShape (const IDOMShapePtr &ptrShape) const =0

  Detect if another shape fits completely inside this shape.
- virtual CShapeDetails getShapeDetails () const =0

  Fetch the regions that comprise the shape.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves class id of IDOMShape.
- static EDL_API IDOMShapePtr **createRect** (IEDLClassFactory *pFactory, float resolution, const FRect &rect)

  Create a rectangular shape.
- static EDL_API IDOMShapePtr **createRectPixels** (IEDLClassFactory *pFactory, float resolution, const Int↩ Rect &rect)

  As per createRect(), but the rectangle is pixels. The shape will be tagged with the given resolution, however the dimensions of the shape will not change.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.275.1 Detailed Description

Interface to an IDOMShape.

A shape describes a scan-converted path. That is, a path that has been scanned to find which pixels would be affected by the path.

This is useful for a number of things. Once a shape has been made from a path it may be compared to other shapes to test for things like intersection and occulsion using higher accuracy than can be obtained by simply comparing bounding boxes.

A shape may be compared only with another shape scan-converted at the same resolution.

As of current this interface does not allow access to the low level details of the scan converted result. Rather it is intended to allow for simple comparisons.

7.275.2 Member Function Documentation

classID()

```
static const CClassID & IDOMShape::classID ( )  [inline], [static]
```

Retrieves class id of IDOMShape.

Returns

CClassID Class id of the element

completelyContainsShape()

```
virtual bool IDOMShape::completelyContainsShape (
            const IDOMShapePtr & ptrShape ) const  [pure virtual]
```

Detect if another shape fits completely inside this shape.

Parameters

in	*ptrShape*	The shape to test.

Returns

bool True if the given shape fits completely in this shape.

difference()

```
virtual void IDOMShape::difference (
            const IDOMShapePtr & ptrShape )  [pure virtual]
```

Reduce this shape by subtracting another shape.

Parameters

in	*ptrShape*	The shape to subtract.

getAsGeometry()

```
virtual IDOMPathGeometryPtr IDOMShape::getAsGeometry (
            IEDLClassFactory * factory ) const  [pure virtual]
```

Get the shape as path geometry.

```
Take care when using this API as it may generate very complex
geometry, as the generated geometry will consist of rectangles
representing the spans encoded in the shape, For complex shapes
with many spans, this geometry can become very large.
```

Parameters

factory	The factory to use.

Returns

IDOMPathGeometryPtr The resulting geometry.

getAsImage()

```
virtual IDOMImagePtr IDOMShape::getAsImage (
            const ISessionPtr & session,
            bool bUseTempFileForImage ) const  [pure virtual]
```

Return in `image` an 1-bit-per-pixel DeviceGray image representation of this shape.

Parameters

in	*session*	The current session.
in	*bUseTempFileForImage*	If true, the method creates a temporary file (the lifecycle of which Mako manages itself) to store the underlying image data in. If false, it uses an in-memory stream to store the image data. Client code can determine the size of the image that will be created using this class's `getBounds()` method, and choose which value to send for this flag. Note that as the image returned is 1 bit per pixel, the number of bytes required to store an image is the product of the bounds divided by the size of a byte.

Returns

IDOMImagePtr The resulting image

getBounds()

```
virtual FRect IDOMShape::getBounds ( ) const  [pure virtual]
```

Get the bounds (in pixels) of the shape.

Returns

FRect The bounds of the shape.

getIsEmpty()

```
virtual bool IDOMShape::getIsEmpty ( ) const  [pure virtual]
```

Detect an empty shape.

Returns

bool True if the shape is empty.

getIsRect()

```
virtual bool IDOMShape::getIsRect ( ) const  [pure virtual]
```

Detect if a shape is a rectangle.

Returns

bool True if the shape is rectangular.

getResolution()

```
virtual float IDOMShape::getResolution ( ) const  [pure virtual]
```

Get the resolution of the shape.

Returns

float Returns the resolution of the shape.

getShapeDetails()

```
virtual CShapeDetails IDOMShape::getShapeDetails ( ) const  [pure virtual]
```

Fetch the regions that comprise the shape.

Returns

CShapeDetails The details of the shape. See #CShapeDetails

intersect()

```
virtual void IDOMShape::intersect (
            const IDOMShapePtr & ptrShape )  [pure virtual]
```

Reduce this shape to its intersection with another shape.

Parameters

in	*ptrShape*	The shape with which to intersect.

intersects()

```
virtual bool IDOMShape::intersects (
            const IDOMShapePtr & ptrShape ) const  [pure virtual]
```

Detect if this shape intersects with another.

Parameters

in	*ptrShape*	The shape to test with.

Returns

bool True if the shapes intersect, false otherwise.

isEqualTo()

```
virtual bool IDOMShape::isEqualTo (
            const IDOMShapePtr & ptrShape ) const  [pure virtual]
```

Check this shape for equality to another shape.

Parameters

in	*ptrShape*	The shape to compare.

Returns

bool True if the shapes are equivalent.

offset()

```
virtual void IDOMShape::offset (
            int32 offsetX,
            int32 offsetY )  [pure virtual]
```

Offset the shape by the given numbers of pixels.

Parameters

offsetX	The number of pixels to offset in x
offsetY	The number of pixels to offset in y

unite()

```
virtual void IDOMShape::unite (
            const IDOMShapePtr & ptrShape )  [pure virtual]
```

Unite this shape with another.

Parameters

in	*ptrShape*	The shape with which to unite.

The documentation for this class was generated from the following file:

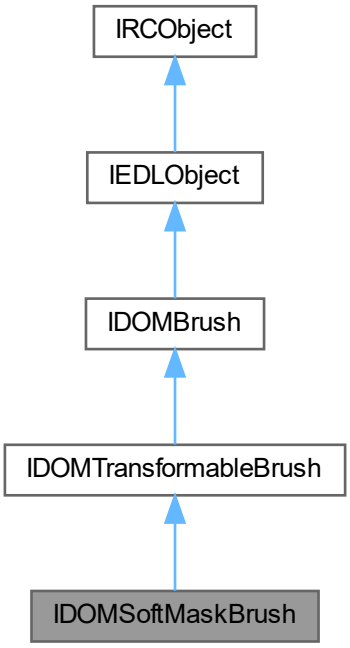
- idomshape.h

7.276 IDOMSoftMaskBrush Class Reference

IDOMSoftMaskBrush provides a way of representing a PDF style soft mask in it's entirity. The soft mask brush contains a suitable IDOMTransparency group, as well as the necessary soft mask details. See section 7.5.4 of the PDF 1.7 specification. These are only allowed for OpacityMask entries.

```
#include <idombrush.h>
```

Inheritance diagram for IDOMSoftMaskBrush:



Classes

- class Data

  Initialization data.

Public Types

- enum eSoftMaskType

  Enum for soft mask interpretation.

Public Types inherited from IDOMBrush

- enum eBrushType {
  eSolidColor , eLinearGradient , eRadialGradient , eImage ,
  eMasked , eVisual , eSoftMask , eTilingPattern ,
  eType1ShadingPattern , eType2ShadingPattern , eType3ShadingPattern , eType4567ShadingPattern ,
  eNull }

  Brush type enumeration.

Public Member Functions

- virtual eSoftMaskType getSoftMaskType () const =0

  Retrieves the soft mask type.
- virtual IDOMTransparencyGroupPtr getGroup () const =0

  Retrieves the transparency group used for the mask.
- virtual IDOMColorPtr getBackdropColor () const =0

  Retrieves the color used for luminosity backdrop composition.
- virtual IDOMFunctionPtr getTransferFunction () const =0

  Retrieves the function used for luminosity used for deriving mask values.

Public Member Functions inherited from IDOMTransformableBrush

- virtual const FMatrix & getRenderTransform () const =0

  Retrieves the render transform matrix.
- virtual void setRenderTransform (const FMatrix &matrix)=0

  Sets the render transform matrix.

Public Member Functions inherited from IDOMBrush

- virtual eBrushType getBrushType () const =0

  Retrieves the type of the brush.
- virtual float getOpacity () const =0

  Retrieves the opacity value of the brush element.
- virtual void setOpacity (float opc)=0

  Sets the opacity value of a brush element.
- virtual IDOMBrushPtr getAdjustedForUseInTransformedNode (IEDLClassFactory *pFactory, const FMatrix &nodeTransform)

  Get a version of this brush adjusted for use inside a node with the given transform.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static EDL_API IDOMSoftMaskBrushPtr create (IEDLClassFactory *pFactory, const IDOMTransparency↩ GroupPtr &group, eSoftMaskType type, const FMatrix &renderTransform=FMatrix(), const IDOMColorPtr &backdropColor=IDOMColorPtr(), const IDOMFunctionPtr &transfer=IDOMFunctionPtr())

  Simplified creator for a soft mask brush. Throws an IEDLError on failure.
- static const CClassID & classID ()

  Retrieves class id of IDOM.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.276.1 Detailed Description

IDOMSoftMaskBrush provides a way of representing a PDF style soft mask in it's entirity. The soft mask brush contains a suitable IDOMTransparency group, as well as the necessary soft mask details. See section 7.5.4 of the PDF 1.7 specification. These are only allowed for OpacityMask entries.

7.276.2 Member Function Documentation

classID()

```
static const CClassID & IDOMSoftMaskBrush::classID ( )  [inline], [static]
```

Retrieves class id of IDOM.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMSoftMaskBrushPtr IDOMSoftMaskBrush::create (
            IEDLClassFactory * pFactory,
            const IDOMTransparencyGroupPtr & group,
            eSoftMaskType type,
            const FMatrix & renderTransform = FMatrix(),
            const IDOMColorPtr & backdropColor = IDOMColorPtr(),
            const IDOMFunctionPtr & transfer = IDOMFunctionPtr() )  [static]
```

Simplified creator for a soft mask brush. Throws an IEDLError on failure.

Parameters

pFactory	The factory to use.
group	The group to form the content of the soft mask.
type	The type of the soft mask.
renderTransform	The desired render transform.
backdropColor	The backdrop color (must have the same color space as the group). Optional.
transfer	The soft mask transfer function. Optional.

Returns

IDOMSoftMaskBrushPtr The new brush.

getBackdropColor()

```
virtual IDOMColorPtr IDOMSoftMaskBrush::getBackdropColor ( ) const  [pure virtual]
```

Retrieves the color used for luminosity backdrop composition.

Parameters

b IDOMColorPtr The backdrop color, or NULL if no backdrop color is present.

getGroup()

```
virtual IDOMTransparencyGroupPtr IDOMSoftMaskBrush::getGroup ( ) const  [pure virtual]
```

Retrieves the transparency group used for the mask.

Returns

IDOMTransparencyGroupPtr The group

getSoftMaskType()

```
virtual eSoftMaskType IDOMSoftMaskBrush::getSoftMaskType ( ) const  [pure virtual]
```

Retrieves the soft mask type.

Returns

eSoftMaskType The soft mask type.

getTransferFunction()

```
virtual IDOMFunctionPtr IDOMSoftMaskBrush::getTransferFunction ( ) const  [pure virtual]
```

Retrieves the function used for luminosity used for deriving mask values.

Returns

IDOMFunctionPtr The transform function, or NULL if there is no transfer function

The documentation for this class was generated from the following file:

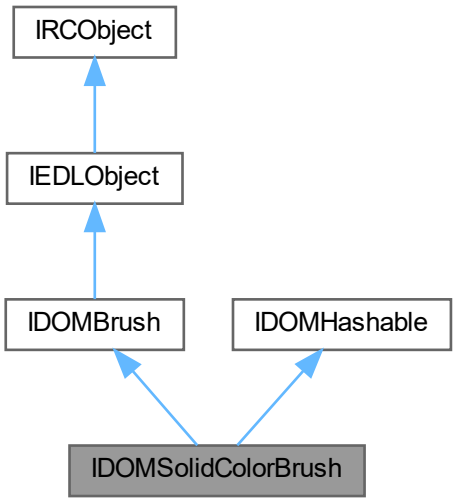
- idombrush.h

7.277 IDOMSolidColorBrush Class Reference

A solid color brush is used to fill defined geometric regions with a solid color. If there is an alpha component of the color, it is combined in a multiplicative way with the corresponding opacity attribute.

```
#include <idombrush.h>
```

Inheritance diagram for IDOMSolidColorBrush:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual IDOMColorPtr getColor () const =0

  Retrieves the color value of the solid color brush.
- virtual void setColor (const IDOMColorPtr &color)=0

  Sets color value of the solid color brush.

Public Member Functions inherited from IDOMBrush

- virtual eBrushType getBrushType () const =0

  Retrieves the type of the brush.
- virtual float getOpacity () const =0

  Retrieves the opacity value of the brush element.
- virtual void setOpacity (float opc)=0

  Sets the opacity value of a brush element.
- virtual IDOMBrushPtr getAdjustedForUseInTransformedNode (IEDLClassFactory *pFactory, const FMatrix &nodeTransform)

  Get a version of this brush adjusted for use inside a node with the given transform.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static EDL_API IDOMSolidColorBrushPtr create (IEDLClassFactory *pFactory, const IDOMColorPtr &color, float opacity=1.0f)

  Simplified solid color brush creation. Throws an IEDLError on failure.
- static IDOMSolidColorBrushPtr createSolidGray (IEDLClassFactory *pFactory, float gray=0.0f)

  Simplified solid color brush creation for DeviceGray colors. Default parameters will yield an opaque DeviceGray black.
- static IDOMSolidColorBrushPtr createSolidRgb (IEDLClassFactory *pFactory, float r=0.0f, float g=0.0f, float b=0.0f)

  Simplified solid color brush creation for DeviceRGB colors. Default parameters will yield an opaque DeviceRGB black.
- static IDOMSolidColorBrushPtr createSolidCmyk (IEDLClassFactory *pFactory, float c=0.0f, float m=0.0f, float y=0.0f, float k=0.0f)

  Simplified solid color brush creation for DeviceCMYK colors. Default parameters will yield an opaque DeviceCMYK white.
- static EDL_API IDOMSolidColorBrushPtr createWithSpaceAndComponents (IEDLClassFactory *pFactory, const IDOMColorSpacePtr &space, double opacity, double component1, double component2=0.0, double component3=0.0, double component4=0.0, double component5=0.0, double component6=0.0, double component7=0.0, double component8=0.0, double component9=0.0, double component10=0.0, double component11=0.0, double component12=0.0, double component13=0.0, double component14=0.0, double component15=0.0, double component16=0.0, double component17=0.0, double component18=0.0, double component19=0.0, double component20=0.0, double component21=0.0, double component22=0.0, double component23=0.0, double component24=0.0, double component25=0.0, double component26=0.0, double component27=0.0, double component28=0.0, double component29=0.0, double component30=0.0, double component31=0.0, double component32=0.0)

  Simplified solid color brush creation routine which takes a color space and components directly. Throws an IEDLError on failure.
- static EDL_API IDOMSolidColorBrushPtr createWithSpaceAndComponentsFromVect (IEDLClassFactory *pFactory, const IDOMColorSpacePtr &space, float opacity, const CEDLVector< float > &components)

  Simplified solid color brush creation routine which takes a color space and components directly. Throws an IEDLError on failure.
- static EDL_API IDOMSolidColorBrushPtr createWithSpaceAndComponentsFromArray (IEDLClassFactory *pFactory, const IDOMColorSpacePtr &space, float opacity, const float *components)

  Simplified solid color brush creation routine which takes a color space and components directly. Throws an IEDLError on failure.
- static const CClassID & classID ()

  Retrieves class id of IDOMIDOMSolidColorBrush.

Additional Inherited Members

Public Types inherited from IDOMBrush

- enum eBrushType {
  eSolidColor , eLinearGradient , eRadialGradient , eImage ,
  eMasked , eVisual , eSoftMask , eTilingPattern ,
  eType1ShadingPattern , eType2ShadingPattern , eType3ShadingPattern , eType4567ShadingPattern ,
  eNull }

  Brush type enumeration.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.277.1 Detailed Description

A solid color brush is used to fill defined geometric regions with a solid color. If there is an alpha component of the color, it is combined in a multiplicative way with the corresponding opacity attribute.

7.277.2 Member Function Documentation

classID()

```
static const CClassID & IDOMSolidColorBrush::classID ( )  [inline], [static]
```

Retrieves class id of IDOMIDOMSolidColorBrush.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMSolidColorBrushPtr IDOMSolidColorBrush::create (
            IEDLClassFactory * pFactory,
            const IDOMColorPtr & color,
            float opacity = 1.0f )  [static]
```

Simplified solid color brush creation. Throws an IEDLError on failure.

Parameters

pFactory	The EDL Class factory to use.
color	The color to use.
opacity	The alpha to use.

Returns

IDOMSolidColorBrushPtr The new brush.

createSolidCmyk()

```
static IDOMSolidColorBrushPtr IDOMSolidColorBrush::createSolidCmyk (
            IEDLClassFactory * pFactory,
            float c = 0.0f,
            float m = 0.0f,
            float y = 0.0f,
            float k = 0.0f )  [inline], [static]
```

Simplified solid color brush creation for DeviceCMYK colors. Default parameters will yield an opaque DeviceCMYK white.

Parameters

pFactory	The EDL Class factory to use.
c	The cyan level to use.
m	The magenta level to use.
y	The yellow level to use.
k	The black level to use.

Returns

IDOMSolidColorBrushPtr The new brush.

createSolidGray()

```
static IDOMSolidColorBrushPtr IDOMSolidColorBrush::createSolidGray (
            IEDLClassFactory * pFactory,
            float gray = 0.0f )  [inline], [static]
```

Simplified solid color brush creation for DeviceGray colors. Default parameters will yield an opaque DeviceGray black.

Parameters

pFactory	The EDL Class factory to use.
gray	The gray level to use (0 is black).

Returns

IDOMSolidColorBrushPtr The new brush.

createSolidRgb()

```
static IDOMSolidColorBrushPtr IDOMSolidColorBrush::createSolidRgb (
            IEDLClassFactory * pFactory,
            float r = 0.0f,
            float g = 0.0f,
            float b = 0.0f )  [inline], [static]
```

Simplified solid color brush creation for DeviceRGB colors. Default parameters will yield an opaque DeviceRGB black.

Parameters

pFactory	The EDL Class factory to use.
r	The red level to use.
g	The green level to use.
b	The blue level to use.

Returns

IDOMSolidColorBrushPtr The new brush.

createWithSpaceAndComponents()

```
static EDL_API IDOMSolidColorBrushPtr IDOMSolidColorBrush::createWithSpaceAndComponents (
            IEDLClassFactory * pFactory,
            const IDOMColorSpacePtr & space,
            double opacity,
            double component1,
            double component2 = 0.0,
            double component3 = 0.0,
            double component4 = 0.0,
            double component5 = 0.0,
            double component6 = 0.0,
            double component7 = 0.0,
            double component8 = 0.0,
            double component9 = 0.0,
            double component10 = 0.0,
            double component11 = 0.0,
            double component12 = 0.0,
            double component13 = 0.0,
            double component14 = 0.0,
            double component15 = 0.0,
            double component16 = 0.0,
            double component17 = 0.0,
            double component18 = 0.0,
            double component19 = 0.0,
            double component20 = 0.0,
            double component21 = 0.0,
            double component22 = 0.0,
            double component23 = 0.0,
            double component24 = 0.0,
            double component25 = 0.0,
            double component26 = 0.0,
            double component27 = 0.0,
            double component28 = 0.0,
            double component29 = 0.0,
            double component30 = 0.0,
            double component31 = 0.0,
            double component32 = 0.0 )  [static]
```

Simplified solid color brush creation routine which takes a color space and components directly. Throws an IEDLError on failure.

Parameters

pFactory	The EDL Class factory to use.
space	The color space to use.
opacity	The opacity to use (assigned to the brush, not the color)
component1	The first color component to use.
component2	The second color component to use (if applicable).
component3	The third color component to use (if applicable).
component4	The fourth color component to use. (if applicable)
component5	The fifth color component to use (if applicable).

Parameters

component6	The sixth color component to use (if applicable).
component7	The seventh color component to use.
component8	The eighth color component to use (if applicable).
component9	The ninth color component to use (if applicable).
component10	The tenth color component to use (if applicable).
component11	The eleventh color component to use (if applicable).
component12	The twelth color component to use (if applicable).
component13	The thirteenth color component to use (if applicable).
component14	The fourteenth color component to use (if applicable).
component15	The fifteenth color component to use (if applicable).
component16	The sixteenth color component to use (if applicable).
component17	The seventeenth color component to use (if applicable).
component18	The eightteenth color component to use (if applicable).
component19	The nineteenth color component to use (if applicable).
component20	The twentieth color component to use (if applicable).
component21	The twenty-first color component to use (if applicable).
component22	The twenty-second color component to use (if applicable).
component23	The twenty-third color component to use (if applicable).
component24	The twenty-fourth color component to use (if applicable).
component25	The twenty-fifth color component to use (if applicable).
component26	The twenty-sixth color component to use (if applicable).
component27	The twenty-seventh color component to use (if applicable).
component28	The twenty-eighth color component to use (if applicable).
component29	The twenty-ninth color component to use (if applicable).
component30	The thirtieth color component to use (if applicable).
component31	The thirty-first color component to use (if applicable).
component32	The thirty-second color component to use (if applicable).

Returns

IDOMColorPtr The new color.

IDOMColorPtr The new color.

createWithSpaceAndComponentsFromArray()

```
static EDL_API IDOMSolidColorBrushPtr IDOMSolidColorBrush::createWithSpaceAndComponentsFrom↩
Array (
            IEDLClassFactory * pFactory,
            const IDOMColorSpacePtr & space,
            float opacity,
            const float * components )  [static]
```

Simplified solid color brush creation routine which takes a color space and components directly. Throws an IEDLError on failure.

Parameters

pFactory	The EDL Class factory to use.
space	The color space to use.
opacity	The opacity to use (assigned to the brush, not the color)
components	An array of color components to use, as floats. The number of components MUST match the number of components expected of the color space.

Returns

IDOMColorPtr The new color.

createWithSpaceAndComponentsFromVect()

```
static EDL_API IDOMSolidColorBrushPtr IDOMSolidColorBrush::createWithSpaceAndComponentsFrom↩
Vect (
            IEDLClassFactory * pFactory,
            const IDOMColorSpacePtr & space,
            float opacity,
            const CEDLVector< float > & components )  [static]
```

Simplified solid color brush creation routine which takes a color space and components directly. Throws an IEDLError on failure.

Parameters

pFactory	The EDL Class factory to use.
space	The color space to use.
opacity	The opacity to use (assigned to the brush, not the color)
components	A vector of color components to use, as floats. The number of components MUST match the number of components expected of the color space.

Returns

IDOMColorPtr The new color.

getColor()

```
virtual IDOMColorPtr IDOMSolidColorBrush::getColor ( ) const  [pure virtual]
```

Retrieves the color value of the solid color brush.

Returns

IDOMColorPtr The color

setColor()

```
virtual void IDOMSolidColorBrush::setColor (
            const IDOMColorPtr & color )  [pure virtual]
```

Sets color value of the solid color brush.

Parameters

color	The color value

The documentation for this class was generated from the following file:

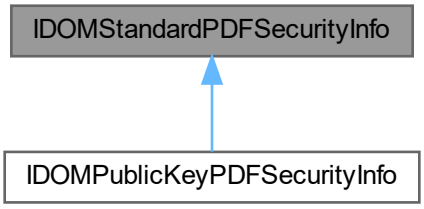
- idombrush.h

7.278 IDOMStandardPDFSecurityInfo Class Reference

Represents security information from PDF Standard encryption handler.

`#include <idomsecurity.h>`

Inheritance diagram for IDOMStandardPDFSecurityInfo:



Public Types

- enum ePermissionsFlags {
  ePrintAllowed = 0x0004 , eEditingAllowed = 0x0008 , eCopyingAllowed = 0x0010 , eAnnotationEditingAllowed = 0x0020 ,
  eFormFillingAllowed = 0x0100 , eContentAccessibilityExtractionAllowed = 0x0200 , eDocumentAssemblyAllowed = 0x0400 , eHighQualityPrintAllowed = 0x0800 ,
  eEverythingAllowed = 0x0ffc }

  Bit values for each permission flag as available from getPermissionFlags.

Public Member Functions

- virtual int32 getHandlerRevision () const =0

  Retrieves the revision of the handler.
- virtual EDLSysString getUserPassword () const =0

  Retrieves the user access password.
- virtual bool encryptMetadata () const =0

  Returns true if document level metadata stream is to be encrypted meaningful only when algorithm code = 4.
- virtual bool isOwnerAccess () const =0

  Returns true if the owner password was used to open the PDF.

- bool isPrintingAllowed () const

  Returns true if printing is allowed.
- bool isHighQualityPrintingAllowed () const

  Returns true if high quality printing is allowed.
- bool isEditingAllowed () const

  Returns true if modification of the contents by operations other than those specified by isEditingAnnotationsAllowed(), isFillingFormAllowed() and isDocAssemblyAllowed() is allowed.
- bool isCopyingAllowed () const

  Returns true if copying of text and graphics from the document for operations other than those specified by isExtractionAllowed() is allowed.
- bool isAnnotationEditingAllowed () const

  Returns true if editing of annotations and forms is allowed.
- bool isFillingFormsAllowed () const

  Returns true if form-filling is allowed.
- bool isContentAccessibilityExtractionAllowed () const

  Returns true if extracting for content accessibility is allowed.
- bool isDocumentAssemblyAllowed () const

  Returns true if document assembly is allowed.
- virtual uint32 **getPermissionFlags** () const =0

  Retrieve the raw permissions flags according to the PDF 1.7 spec, table 3.2.0. These should be interpreted with regard to the handler revision, as some flags are not available for all security handler revisions.

7.278.1 Detailed Description

Represents security information from PDF Standard encryption handler.

7.278.2 Member Function Documentation

encryptMetadata()

```
virtual bool IDOMStandardPDFSecurityInfo::encryptMetadata ( ) const  [pure virtual]
```

Returns true if document level metadata stream is to be encrypted meaningful only when algorithm code = 4.

Returns

bool The metadata encryption setting

getHandlerRevision()

```
virtual int32 IDOMStandardPDFSecurityInfo::getHandlerRevision ( ) const  [pure virtual]
```

Retrieves the revision of the handler.

Returns

int32 The handler revision

getUserPassword()

```
virtual EDLSysString IDOMStandardPDFSecurityInfo::getUserPassword ( ) const  [pure virtual]
```

Retrieves the user access password.

Returns

EDLSysString The user password

Implemented in IDOMPublicKeyPDFSecurityInfo.

isAnnotationEditingAllowed()

```
bool IDOMStandardPDFSecurityInfo::isAnnotationEditingAllowed ( ) const  [inline]
```

Returns true if editing of annotations and forms is allowed.

Returns

bool Is annotation editing allowed?

isContentAccessibilityExtractionAllowed()

```
bool IDOMStandardPDFSecurityInfo::isContentAccessibilityExtractionAllowed ( ) const  [inline]
```

Returns true if extracting for content accessibility is allowed.

Returns

bool Is content accessibility extraction allowed?

isCopyingAllowed()

```
bool IDOMStandardPDFSecurityInfo::isCopyingAllowed ( ) const  [inline]
```

Returns true if copying of text and graphics from the document for operations other than those specified by isExtractionAllowed() is allowed.

Returns

bool Is copying allowed?

isDocumentAssemblyAllowed()

```
bool IDOMStandardPDFSecurityInfo::isDocumentAssemblyAllowed ( ) const  [inline]
```

Returns true if document assembly is allowed.

Returns

bool Is document assembly allowed?

isEditingAllowed()

```
bool IDOMStandardPDFSecurityInfo::isEditingAllowed ( ) const  [inline]
```

Returns true if modification of the contents by operations other than those specified by isEditingAnnotationsAllowed(), isFillingFormAllowed() and isDocAssemblyAllowed() is allowed.

Returns

bool Is editing allowed?

isFillingFormsAllowed()

```
bool IDOMStandardPDFSecurityInfo::isFillingFormsAllowed ( ) const  [inline]
```

Returns true if form-filling is allowed.

Returns

bool Is filling of forms allowed?

isHighQualityPrintingAllowed()

```
bool IDOMStandardPDFSecurityInfo::isHighQualityPrintingAllowed ( ) const  [inline]
```

Returns true if high quality printing is allowed.

Returns

bool Is high quality printing allowed?

isOwnerAccess()

```
virtual bool IDOMStandardPDFSecurityInfo::isOwnerAccess ( ) const  [pure virtual]
```

Returns true if the owner password was used to open the PDF.

Returns

bool Owner password used to access?

Implemented in IDOMPublicKeyPDFSecurityInfo.

isPrintingAllowed()

```
bool IDOMStandardPDFSecurityInfo::isPrintingAllowed ( ) const  [inline]
```

Returns true if printing is allowed.

Returns

bool Is printing allowed?

The documentation for this class was generated from the following file:

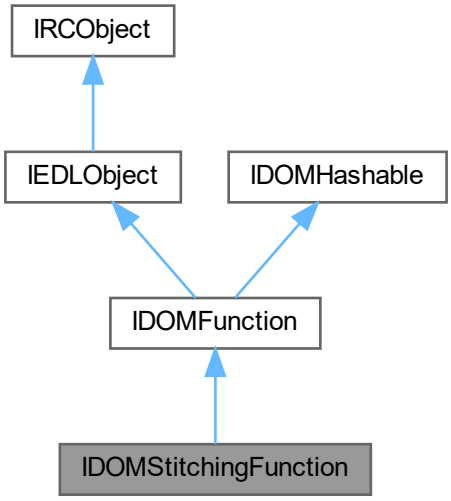
- idomsecurity.h

7.279 IDOMStitchingFunction Class Reference

Interface for stitching functions. See section 3.9.3 of the PDF 1.7 Reference. Default values are as per described in that reference. There can only be one input for this function, and the functions contained therein must also handle one input.

```
#include <idomfunction.h>
```

Inheritance diagram for IDOMStitchingFunction:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual uint32 getNumFunctions () const =0

  Get the number of functions that are stitched.
- virtual IDOMFunctionPtr getFunctionAtIndex (uint32 index) const =0

  Get the function for a given function index.
- virtual const CEDLVector< float > & getBoundsVector () const =0

  Get a reference to the bounds vector for this function.
- virtual const CEDLVector< float > & getEncodeVector () const =0

  Get a reference to the encode vector for this function.

Public Member Functions inherited from IDOMFunction

- virtual eFunctionType getFunctionType () const =0

  Retrieves function type.
- virtual uint32 getNumInputValues () const =0

  Get the number of input values that this function will operate on.
- virtual uint32 getNumOutputValues () const =0

  Get the number of output values that this function will produce.
- virtual void getInputDomain (uint32 inputNum, float &low, float &high) const =0

  Get the input domain for a given input to the function.
- virtual bool getOutputRange (uint32 outputNum, float &low, float &high) const =0

  Get the output range for a given input to the function.
- virtual void evaluate (const float *inputValues, float *outputValues) const =0

  Evaluate the input through the function and return the result.
- virtual void evaluate (const int32 *inputValues, float *outputValues) const =0

  Evaluate the input through the function and return the result.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves class id of IDOM.

Additional Inherited Members

Public Types inherited from IDOMFunction

- enum eFunctionType

  An enum for the various types of functions.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.279.1 Detailed Description

Interface for stitching functions. See section 3.9.3 of the PDF 1.7 Reference. Default values are as per described in that reference. There can only be one input for this function, and the functions contained therein must also handle one input.

7.279.2 Member Function Documentation

classID()

```
static const CClassID & IDOMStitchingFunction::classID ( )  [inline], [static]
```

Retrieves class id of IDOM.

Returns

CClassID class id of the element

getBoundsVector()

```
virtual const CEDLVector< float > & IDOMStitchingFunction::getBoundsVector ( ) const  [pure
virtual]
```

Get a reference to the bounds vector for this function.

Returns

CEDLVector<**float**> Reference to a constant CEDLBuffer<float> containing the bounds.

getEncodeVector()

```
virtual const CEDLVector< float > & IDOMStitchingFunction::getEncodeVector ( ) const  [pure
virtual]
```

Get a reference to the encode vector for this function.

Returns

CEDLVector<**float**> Reference to a constant CEDLBuffer<float> containing the encode.

getFunctionAtIndex()

```
virtual IDOMFunctionPtr IDOMStitchingFunction::getFunctionAtIndex (
            uint32 index ) const  [pure virtual]
```

Get the function for a given function index.

Parameters

index	A zero based index for the desired function.

Returns

DOMFunctionPtr The function

getNumFunctions()

```
virtual uint32 IDOMStitchingFunction::getNumFunctions ( ) const  [pure virtual]
```

Get the number of functions that are stitched.

Returns

uint32 The number of functions

The documentation for this class was generated from the following file:

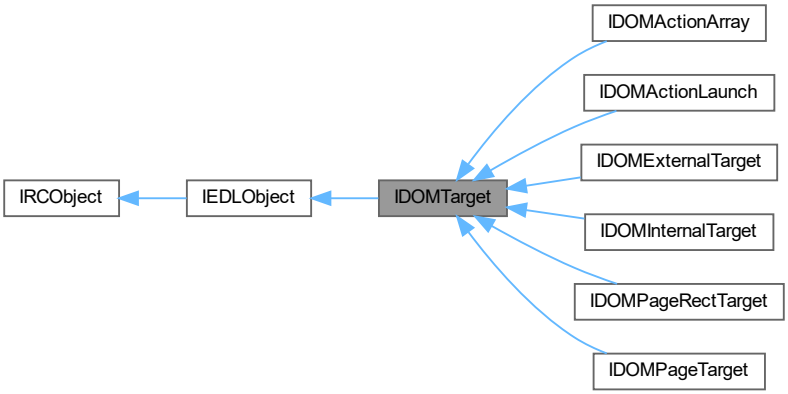
- idomfunction.h

7.280 IDOMTarget Class Reference

Base class for defining hyperlink targets in a document.

`#include <idomtarget.h>`

Inheritance diagram for IDOMTarget:



Public Types

- enum eTargetType {
  eExternal , eInternal , ePage , ePageRect ,
  eActionGoToR , eActionGoToE , eActionLaunch , eActionThread ,
  eActionSound , eActionMovie , eActionHide , eActionNamed ,
  eActionSubmitForm , eActionResetForm , eActionImportData , eActionJavaScript ,
  eActionSetOCGState , eActionRendition , eActionTrans , eActionGoTo3DView ,
  eActionArray }

  An enumeration of target types.

Public Member Functions

- virtual eTargetType getTargetType () const =0

  Retrieves the target type.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.280.1 Detailed Description

Base class for defining hyperlink targets in a document.

Several subclasses exist, defining more specialized types of target. For example, internal targets appear in the same document as the working page, whereas external targets appear either in a different document, or on the web.

7.280.2 Member Function Documentation

getTargetType()

```
virtual eTargetType IDOMTarget::getTargetType ( ) const  [pure virtual]
```

Retrieves the target type.

Returns

eTargetType The target type

Implemented in IDOMExternalTarget, IDOMInternalTarget, IDOMPageTarget, IDOMPageRectTarget, IDOMActionLaunch, and IDOMActionArray.

The documentation for this class was generated from the following file:

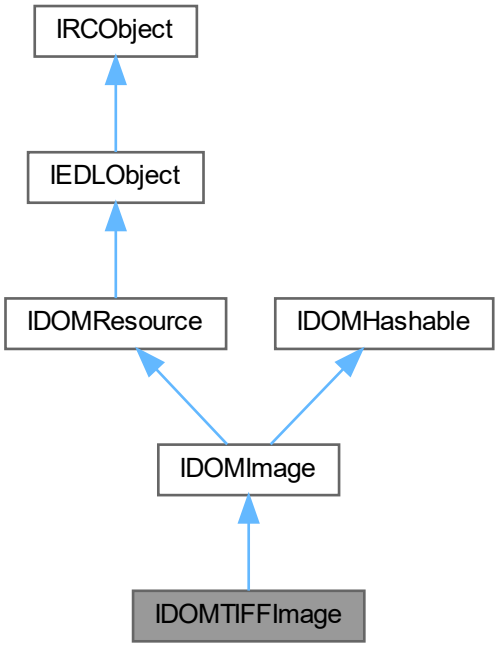
- idomtarget.h

7.281 IDOMTIFFImage Class Reference

IDOMTIFFImage interface.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMTIFFImage:



Classes

- class Data
  Initialization data.

Public Types

- enum eTIFFCompression { }
  Available TIFF compression schemes.
- enum eTIFFPrediction
  Available TIFF prediction schemes.

Public Member Functions

- virtual uint16 getImageIndex () const =0
  Get the image index of this TIFF to use.

Public Member Functions inherited from IDOMImage

- virtual IImageDecoderPtr createImageDecoder (IEDLClassFactory *factory, const IDOMImagePropertiesPtr &imageProperties)=0

  Creates a properly initialized image decoder object that reads from an inputstream that is specific to that image format.
- virtual IImageFramePtr getImageFrame (IEDLClassFactory *factory)

  Fetch the image frame; convenience.
- virtual IImageEncoderPtr createImageEncoder (const ISessionPtr &session, const IOutputStreamPtr &imageDest, const IDOMImagePropertiesPtr &imageProperties)=0

  Creates a properly initialized image encoder object that writes to an outputstream that is specific to that image format.
- virtual IDOMImagePropertiesPtr getImageProperties ()=0

  Returns an object that stores the properties for this image object. The properties can then be inspected (or more added) by clients that need to manipulate the image resource.
- virtual eDOMImageType getImageType ()=0

  Retrieves the image type.
- virtual bool getIsRendered ()=0

  Determine if the image is as a result of rendering. This is indicated if the image type is eDITRendered or if the image explicitly notes this is the case (such as for IDOMPDFImage).
- virtual IDOMImagePtr getImageWithSubstitutedColorSpace (IEDLClassFactory *factory, const IDOMColor↩SpacePtr &colorSpace)

  Obtain an image that is the same as this image, but with the colorspace substituted for another.

Public Member Functions inherited from IDOMResource

- virtual IInputStreamPtr getStream () const =0

  Retrieves the resource stream.
- virtual void setStream (const IInputStreamPtr &stream)=0

  Sets the resource stream for the node.
- virtual uint64 getStreamLength () const =0

  Retrieves the stream length, if it is available.
- virtual const EDLSysString & getUri () const =0

  Retrieves the resource URI.
- virtual void setUri (const EDLSysString &uri)=0

  Sets the resource URI.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static EDL_API IDOMTIFFImagePtr create (IEDLClassFactory *pFactory, const IInputStreamPtr &stream, uint16 imageIndex=0, bool allowMultiChannel=false)

  Create a TIFF Image resource with the given TIFF stream.
- static EDL_API IDOMImagePtr createWriterAndImage (const ISessionPtr &session, IImageFrameWriterPtr &frame, const IDOMColorSpacePtr &colorSpace, uint32 width, uint32 height, uint8 bitsPerComponent=8, double xResolution=96.0, double yResolution=96.0, eTIFFCompression compresssionType=eTCAuto, eTIFFPrediction predictionType=eTPNone, eImageExtraChannelType extraChannel=eIECNone, bool bigTiff=false, const IInputStreamPtr &inStream=IInputStreamPtr(), const IOutputStreamPtr &outStream=IOutputStreamPtr())

  Create an IDOMTIFFImage and frame that can be used to populate same.
- static EDL_API void encode (const ISessionPtr &pSession, const IDOMImagePtr &image, const IOutputStreamPtr &stream, eTIFFCompression scheme=eTCAuto, eTIFFPrediction predictionType=eTPNone, bool bigTiff=false)

  Encode an image as a TIFF stream, returning the stream. This routine may convert the image samples into a form that may be encoded as TIFF if required, such as by converting to a supported color space.
- static EDL_API void encode (const ISessionPtr &pSession, const IImageFramePtr &frame, const IOutputStreamPtr &stream, eTIFFCompression scheme=eTCAuto, eTIFFPrediction predictionType=eTPNone, bool bigTiff=false)

  Encode the contents of an IImageFrame as a TIFF stream, returning the stream. This routine may convert the image samples into a form that may be encoded as TIFF if required, such as by converting to a supported color space.
- static const CClassID & classID ()

  Retrieves class id of IDOMTIFFImage.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.281.1 Detailed Description

IDOMTIFFImage interface.

7.281.2 Member Enumeration Documentation

eTIFFCompression

```
enum IDOMTIFFImage::eTIFFCompression
```

Available TIFF compression schemes.

Enumerator

eTCNone	Choose an appropriate scheme for the incoming image.

7.281.3 Member Function Documentation

classID()

```
static const CClassID & IDOMTIFFImage::classID ( )  [inline], [static]
```

Retrieves class id of IDOMTIFFImage.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMTIFFImagePtr IDOMTIFFImage::create (
            IEDLClassFactory * pFactory,
            const IInputStreamPtr & stream,
            uint16 imageIndex = 0,
            bool allowMultiChannel = false )  [static]
```

Create a TIFF Image resource with the given TIFF stream.

Parameters

pFactory	The EDL Class factory to use.
stream	The stream containing the TIFF image.
imageIndex	For multi-image TIFF files, the index of the image to use, with 0 being the first image in the TIFF file.
allowMultiChannel	If true, the TIFF decoder will allow Separated (CMYK) tiff files containing extra color samples. If found, a DeviceN color space will be used. Note however that this DeviceN color space will not likely be useful out of the box, and some external knowledge of the color channels may be required. Please contact GlobalGraphics support if you have specialised needs.

Returns

IDOMTIFFImagePtr The new image.

createWriterAndImage()

```
static EDL_API IDOMImagePtr IDOMTIFFImage::createWriterAndImage (
            const ISessionPtr & session,
            IImageFrameWriterPtr & frame,
```

```
            const IDOMColorSpacePtr & colorSpace,
            uint32 width,
            uint32 height,
            uint8 bitsPerComponent = 8,
            double xResolution = 96.0,
            double yResolution = 96.0,
            eTIFFCompression compresssionType = eTCAuto,
            eTIFFPrediction predictionType = eTPNone,
            eImageExtraChannelType extraChannel = eIECNone,
            bool bigTiff = false,
            const IInputStreamPtr & inStream = IInputStreamPtr(),
            const IOutputStreamPtr & outStream = IOutputStreamPtr() )  [static]
```

Create an IDOMTIFFImage and frame that can be used to populate same.

Parameters

session	The session to use
frame	On exit, this is populated with a frame ready to receive image data via frame->writeScanLine(). Use frame->flushData() to complete the encoding process.
colorSpace	The color space to use. Must be compatible with the TIFF format.
width	The width of the image, in pixels.
height	The height of the image, in pixels.
bitsPerComponent	The bits per component to use. Must be compatible with the TIFF format.
xResolution	The x resolution, in pixels-per-inch.
yResolution	The y resolution, in pixels-per-inch.
extraChannel	The type of extra channel, if provided. Must be either eIECAlpha or eIECNone for TIFF.
compresssionType	The type of compression to use.
predictionType	The type of prediction to use. Ignored if the predictor is not compatible with the compressor.
bigTiff	Whether or not to use the bigTIFF extensions
inStream	Optional. The first in a pair of streams used to read and write the raw image data if an external stream is desired. If NULL, a temporary store stream will be created. If non NULL, outStream must also be provided.
outStream	Optional. The second in a pair of streams used to read and write the raw image data if an external stream is desired. If NULL, a temporary store stream will be created. If non NULL, inStream must also be provided.

Returns

IDOMImagePtr The resulting image. Not valid until the frame is flushed.

encode() [1/2]

```
static EDL_API void IDOMTIFFImage::encode (
            const ISessionPtr & pSession,
            const IDOMImagePtr & image,
            const IOutputStreamPtr & stream,
            eTIFFCompression scheme = eTCAuto,
            eTIFFPrediction predictionType = eTPNone,
            bool bigTiff = false )  [static]
```

Encode an image as a TIFF stream, returning the stream. This routine may convert the image samples into a form that may be encoded as TIFF if required, such as by converting to a supported color space.

Parameters

pSession	The relevant EDL session
image	The image to be encoded
scheme	The TIFF compression scheme to be used.
predictionType	The type of prediction to use. Ignored if the predictor is not compatible with the compressor.
bigTiff	If true, use bigtff format, allowing for very large images. Not all TIFF-consuming software will support this format.
stream	The stream to use to store the image data. For best performance, this should be a random-access stream (IRAStream).

encode() [2/2]

```
static EDL_API void IDOMTIFFImage::encode (
            const ISessionPtr & pSession,
            const IImageFramePtr & frame,
            const IOutputStreamPtr & stream,
            eTIFFCompression scheme = eTCAuto,
            eTIFFPrediction predictionType = eTPNone,
            bool bigTiff = false )  [static]
```

Encode the contents of an IImageFrame as a TIFF stream, returning the stream. This routine may convert the image samples into a form that may be encoded as TIFF if required, such as by converting to a supported color space.

Parameters

pSession	The relevant EDL session
frame	The frame providing the source image data
scheme	The TIFF compression scheme to be used.
predictionType	The type of prediction to use. Ignored if the predictor is not compatible with the compressor.
bigTiff	If true, use bigtff format, allowing for very large images. Not all TIFF-consuming software will support this format.
stream	The stream to use to store the image data. For best performance, this should be a random-access stream (IRAStream).

getImageIndex()

```
virtual uint16 IDOMTIFFImage::getImageIndex ( ) const  [pure virtual]
```

Get the image index of this TIFF to use.

TIFF files may contain multiple images; this member returns the index (beginning at 0) of the image that will be used from the TIFF file.

Returns

uint16 The image index

The documentation for this class was generated from the following file:

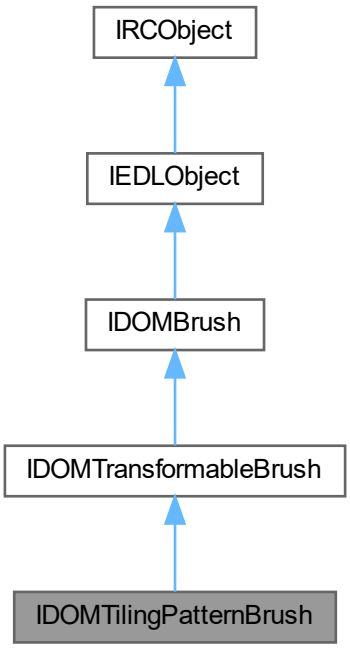
- idomimageresource.h

7.282 IDOMTilingPatternBrush Class Reference

IDOMTilingPatternBrush provides a way of representing a PS style tiling pattern.

`#include <idombrush.h>`

Inheritance diagram for IDOMTilingPatternBrush:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual uint8 getPatternType () const =0

  Retrieves the pattern type.
- virtual const FBox & getBBox () const =0

  Retrieves the bounding box for the pattern.
- virtual void setBBox (const FBox &bBox)=0

  Sets the bounding box for the pattern.
- virtual void getTilingStep (float &xstep, float &ystep) const =0

  Retrieves the tiling step for the pattern.
- virtual void setTilingStep (float xstep, float ystep)=0

  Sets the tiling step for the pattern.

- virtual void setPaintType (uint8 paintType)=0

  Sets the paint type for the pattern.
- virtual uint8 getPaintType () const =0

  Gets the paint type for the pattern.
- virtual void setTilingType (uint8 tilingType)=0

  Sets the tiling type for the pattern.
- virtual uint8 getTilingType () const =0

  Gets the tiling type for the pattern.
- virtual void setPatternColor (const IDOMColorPtr &color)=0

  Sets the pattern color for uncolored pattern (paint type is 2)
- virtual IDOMColorPtr getPatternColor () const =0

  Gets the pattern color for uncolored pattern (paint type is 2)
- virtual IDOMNodePtr getVisual () const =0

  Retrieves smart pointer to visual (path, glyphs, group, canvas) node.
- virtual void setVisual (const IDOMNodePtr &ptrVisual)=0

  Sets visual node.
- virtual IDOMVisualBrushPtr getEquivalentVisualBrush (IEDLClassFactory *pFactory)=0

  Gets an equivalent IDOMVisualBrush brush. If the brush has overlapping tiles, this cannot be done.

Public Member Functions inherited from IDOMTransformableBrush

- virtual const FMatrix & getRenderTransform () const =0

  Retrieves the render transform matrix.
- virtual void setRenderTransform (const FMatrix &matrix)=0

  Sets the render transform matrix.

Public Member Functions inherited from IDOMBrush

- virtual eBrushType getBrushType () const =0

  Retrieves the type of the brush.
- virtual float getOpacity () const =0

  Retrieves the opacity value of the brush element.
- virtual void setOpacity (float opc)=0

  Sets the opacity value of a brush element.
- virtual IDOMBrushPtr getAdjustedForUseInTransformedNode (IEDLClassFactory *pFactory, const FMatrix &nodeTransform)

  Get a version of this brush adjusted for use inside a node with the given transform.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static EDL_API IDOMTilingPatternBrushPtr create (IEDLClassFactory *pFactory, const IDOMNodePtr &visual, const FRect &bBox, uint8 paintType, const IDOMColorPtr &color, float xStep, float yStep, uint8 tiling↩Type=1, const FMatrix &renderTransform=FMatrix())

  Simplified creator for a tiling brush. Throws an IEDLError on failure.
- static const CClassID & classID ()

  Retrieves class id of IDOM.

Additional Inherited Members

Public Types inherited from IDOMBrush

- enum eBrushType {
  eSolidColor , eLinearGradient , eRadialGradient , eImage ,
  eMasked , eVisual , eSoftMask , eTilingPattern ,
  eType1ShadingPattern , eType2ShadingPattern , eType3ShadingPattern , eType4567ShadingPattern ,
  eNull }

  Brush type enumeration.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.282.1 Detailed Description

IDOMTilingPatternBrush provides a way of representing a PS style tiling pattern.

7.282.2 Member Function Documentation

classID()

```
static const CClassID & IDOMTilingPatternBrush::classID ( )  [inline], [static]
```

Retrieves class id of IDOM.

Returns

CClassID Class id of the element

create()

```
static EDL_API IDOMTilingPatternBrushPtr IDOMTilingPatternBrush::create (
            IEDLClassFactory * pFactory,
            const IDOMNodePtr & visual,
            const FRect & bBox,
            uint8 paintType,
            const IDOMColorPtr & color,
            float xStep,
            float yStep,
            uint8 tilingType = 1,
            const FMatrix & renderTransform = FMatrix() )  [static]
```

Simplified creator for a tiling brush. Throws an IEDLError on failure.

Parameters

pFactory	The factory to use.
visual	The node or node tree representing the pattern's content.
bBox	the bounding box of the pattern's content.
paintType	The paint type, either 1 (colored) or 2 (uncolored)
color	The color to use. Must be NULL for paintType 1. For paintType 2, if NULL, A DeviceGray black will be used.
xStep	The tiling step in X
yStep	The tiling step in Y
tilingType	The tiling type; 1 for constant spacing, 2 for no distortion, 3 for constant spacing and faster tiling.
renderTransform	The desired render transform.

Returns

IDOMTilingPatternBrushPtr The new brush.

getBBox()

```
virtual const FBox & IDOMTilingPatternBrush::getBBox ( ) const  [pure virtual]
```

Retrieves the bounding box for the pattern.

Returns

FBox The bounding box

getEquivalentVisualBrush()

```
virtual IDOMVisualBrushPtr IDOMTilingPatternBrush::getEquivalentVisualBrush (
            IEDLClassFactory * pFactory )  [pure virtual]
```

Gets an equivalent IDOMVisualBrush brush. If the brush has overlapping tiles, this cannot be done.

Parameters

pFactory	A pointer to an EDL class factory

Returns

IDOMVisualBrushPtr The resulting visual brush, or NULL if this brush cannot be expressed as a visual brush. This brush will be cached, so do not edit; make a clone if the returned brush needs to be changed.

getPaintType()

```
virtual uint8 IDOMTilingPatternBrush::getPaintType ( ) const  [pure virtual]
```

Gets the paint type for the pattern.

Returns

int32 The paint type (1 for colored tiling pattern, 2 for uncolored)

getPatternColor()

```
virtual IDOMColorPtr IDOMTilingPatternBrush::getPatternColor ( ) const  [pure virtual]
```

Gets the pattern color for uncolored pattern (paint type is 2)

Returns

IDOMColorPtr The color, or NULL for paint type 1

getPatternType()

```
virtual uint8 IDOMTilingPatternBrush::getPatternType ( ) const  [pure virtual]
```

Retrieves the pattern type.

Returns

int Returns pattern type, 1 for tiling pattern

getTilingStep()

```
virtual void IDOMTilingPatternBrush::getTilingStep (
            float & xstep,
            float & ystep ) const  [pure virtual]
```

Retrieves the tiling step for the pattern.

Parameters

xstep	Reference to xstep
ystep	Reference to ystep

getTilingType()

```
virtual uint8 IDOMTilingPatternBrush::getTilingType ( ) const  [pure virtual]
```

Gets the tiling type for the pattern.

Returns

int32 The tiling type (1 for constant spacing, 2 for no distortion, 3 for constant spacing and faster tiling)

getVisual()

```
virtual IDOMNodePtr IDOMTilingPatternBrush::getVisual ( ) const  [pure virtual]
```

Retrieves smart pointer to visual (path, glyphs, group, canvas) node.

Returns

IDOMNodePtr The visual

setBBox()

```
virtual void IDOMTilingPatternBrush::setBBox (
            const FBox & bBox )  [pure virtual]
```

Sets the bounding box for the pattern.

Parameters

bBox	The box to use

setPaintType()

```
virtual void IDOMTilingPatternBrush::setPaintType (
            uint8 paintType )  [pure virtual]
```

Sets the paint type for the pattern.

Parameters

paintType	The paint type (1 for colored tiling pattern, 2 for uncolored)

setPatternColor()

```
virtual void IDOMTilingPatternBrush::setPatternColor (
            const IDOMColorPtr & color )  [pure virtual]
```

Sets the pattern color for uncolored pattern (paint type is 2)

Parameters

color	Reference to the desired color to use.

setTilingStep()

```
virtual void IDOMTilingPatternBrush::setTilingStep (
            float xstep,
            float ystep )  [pure virtual]
```

Sets the tiling step for the pattern.

Parameters

xstep	Reference to xstep
ystep	Reference to ystep

setTilingType()

```
virtual void IDOMTilingPatternBrush::setTilingType (
            uint8 tilingType )  [pure virtual]
```

Sets the tiling type for the pattern.

Parameters

tilingType	The tiling type (1 for constant spacing, 2 for no distortion, 3 for constant spacing and faster tiling)

setVisual()

```
virtual void IDOMTilingPatternBrush::setVisual (
            const IDOMNodePtr & ptrVisual )  [pure virtual]
```

Sets visual node.

Parameters

ptrVisual	Smart pointer to the visual node

The documentation for this class was generated from the following file:

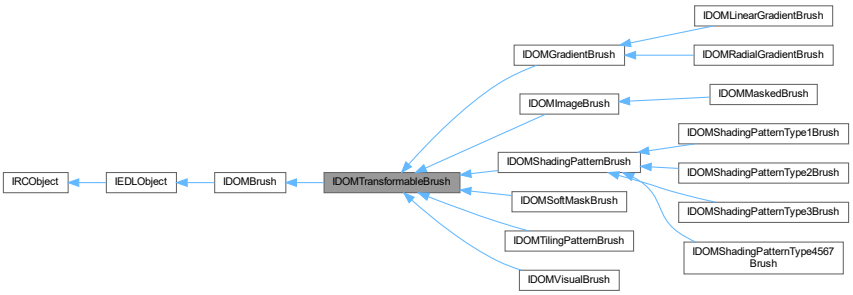
- idombrush.h

7.283 IDOMTransformableBrush Class Reference

Abstract interface for a brush to which a render transform may be applied.

`#include <idombrush.h>`

Inheritance diagram for IDOMTransformableBrush:



Public Member Functions

- virtual const FMatrix & getRenderTransform () const =0

  Retrieves the render transform matrix.
- virtual void setRenderTransform (const FMatrix &matrix)=0

  Sets the render transform matrix.

Public Member Functions inherited from IDOMBrush

- virtual eBrushType getBrushType () const =0

  Retrieves the type of the brush.
- virtual float getOpacity () const =0

  Retrieves the opacity value of the brush element.
- virtual void setOpacity (float opc)=0

  Sets the opacity value of a brush element.
- virtual IDOMBrushPtr getAdjustedForUseInTransformedNode (IEDLClassFactory *pFactory, const FMatrix &nodeTransform)

  Get a version of this brush adjusted for use inside a node with the given transform.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Public Types inherited from IDOMBrush

- enum eBrushType {
  eSolidColor , eLinearGradient , eRadialGradient , eImage ,
  eMasked , eVisual , eSoftMask , eTilingPattern ,
  eType1ShadingPattern , eType2ShadingPattern , eType3ShadingPattern , eType4567ShadingPattern ,
  eNull }

  Brush type enumeration.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.283.1 Detailed Description

Abstract interface for a brush to which a render transform may be applied.

IDOMTransformableBrush and its descendants can store a render transform and are therefore capable of being used meaningfully within a node that itself has a render transform.
A brush that is to be used within such a node has the node's render transform passed in to it, where it is used to modify any existing render transform that the brush may already have, such that using the brush in a transformed node will always provide the expected results.

7.283.2 Member Function Documentation

getRenderTransform()

```
virtual const FMatrix & IDOMTransformableBrush::getRenderTransform ( ) const  [pure virtual]
```

Retrieves the render transform matrix.

Returns

FMatrix The render transform.

setRenderTransform()

```
virtual void IDOMTransformableBrush::setRenderTransform (
            const FMatrix & matrix )  [pure virtual]
```

Sets the render transform matrix.

Parameters

matrix	Render transform matrix

The documentation for this class was generated from the following file:

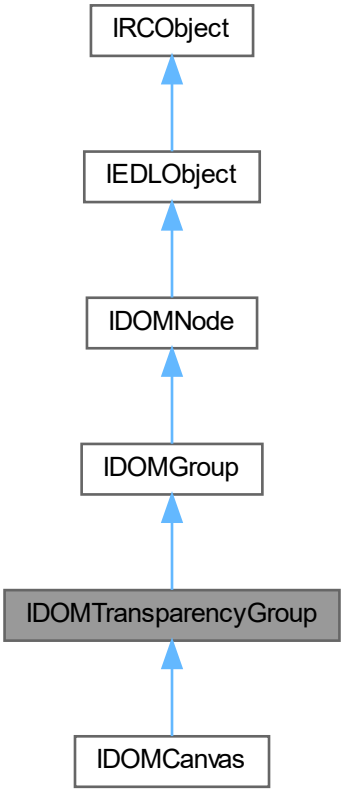
- idombrush.h

7.284 IDOMTransparencyGroup Class Reference

IDOMTransparencyGroup interface. Analogous to PDF Transparency groups.

`#include <idomgroup.h>`

Inheritance diagram for IDOMTransparencyGroup:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual float getOpacity () const =0

  Get the group alpha/opacity.
- virtual void setOpacity (float opacity)=0

  Set the group opacity.
- virtual eBlendMode getBlendMode () const =0

  Get the blend mode to be used for compositing this group with the backdrop.
- virtual void setBlendMode (eBlendMode blendMode)=0

  Set the blend mode to be used for compositing this group with the backdrop.
- virtual IDOMColorSpacePtr getColorSpace () const =0

  Get the group colorspace.
- virtual void setColorSpace (const IDOMColorSpacePtr &colorSpace)=0

  Set the group colorspace.
- virtual bool getIsIsolated () const =0

  Is the group an isolated group? See section 7.5.5 of the PDF 1.7 spec for details.
- virtual void setIsIsolated (bool isolated)=0

  Set whether the group is isolated. See section 7.5.5 of the PDF 1.7 spec for details.
- virtual bool getIsKnockout () const =0

  Is the group a knockout group? See section 7.5.5 of the PDF 1.7 spec for details.
- virtual void setIsKnockout (bool knockout)=0

  Set whether the group is a knockout group. See section 7.5.5 of the PDF 1.7 spec for details.
- virtual IDOMBrushPtr getOpacityMask () const =0

  Retrieves smart pointer to opacity mask.
- virtual void setOpacityMask (const IDOMBrushPtr &ptrOpacityMask)=0

  Sets opacity mask.

Public Member Functions inherited from IDOMGroup

- virtual const FMatrix & getRenderTransform () const =0

  Retrieves render transform matrix of the Group and its children.
- virtual void setRenderTransform (const FMatrix &matrix)=0

  Sets render transform matrix of the Group and its children.
- virtual IDOMPathGeometryPtr getClip () const =0

  Retrieves smart pointer to the clip.
- virtual void setClip (const IDOMPathGeometryPtr &ptrClip)=0

  Sets clip.
- virtual JawsMako::IOptionalContentDetailsPtr getOptionalContentDetails () const =0

  Get the JawsMako Optional Content details, or NULL if the group is not subject to optional content.
- virtual void setOptionalContentDetails (const JawsMako::IOptionalContentDetailsPtr &details)=0

  Set the JawsMako Optional Content details for the group, or NULL to remove. Note that this is only allowed for bare IDOMGroup objects and not for subclasses. Any attempt to set optional content details on any object that is not an IDOMGroup (getNodeType() is eDOMGroupNode) will result in an exception. Note that an IDOMGroup may have optional content details or marked content details, but not both.
- virtual JawsMako::IMarkedContentDetailsPtr getMarkedContentDetails () const =0

  Get the JawsMako Marked Content details for this group, or NULL if the group is not marked.
- virtual void setMarkedContentDetails (const JawsMako::IMarkedContentDetailsPtr &details)=0

  Set the JawsMako Marked Content details for this group, or NULL to remove. Note that this is only allowed for bare IDOMGroup objects and not for subclasses. Any attempt to set marked content details on any object that is not an IDOMGroup (getNodeType() is eDOMGroupNode) will result in an exception. Note that an IDOMGroup may have optional content details or marked content details, but not both.

Public Member Functions inherited from IDOMNode

- virtual ~**IDOMNode** ()

  virtual destructor
- virtual DOMid getDOMid () const =0

  Retrieves the node ID.
- virtual void setDOMid (DOMid id)=0

  Sets the node ID.
- virtual eDOMNodeType getNodeType () const =0

  Retrieves the DOM node type.
- virtual bool getProperty (const EDLSysString &propertyName, PValue &propertyValue) const =0

  Retrieves the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.
- virtual void setProperty (const EDLSysString &propertyName, const PValue &propertyValue)=0

  Sets the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.
- virtual void removeProperty (const EDLSysString &propertyName)=0

  Removes property.
- virtual IEDLSysStringCollectionEnumPtr getPropertyCollectionEnum ()=0

  Retrieves a navigable list of the property names stored on this node.
- virtual bool hasChildNodes () const =0

  Function that indicates whether this node is a parent to other nodes.
- virtual IDOMNodePtr getParentNode () const =0

  Gets the parent node of this node.
- virtual IDOMNodePtr getFirstChild () const =0

  Gets the first child node of this node.
- virtual IDOMNodePtr getLastChild () const =0

  Gets the last child node of this node.
- virtual IDOMNodePtr getNextChild (const IDOMNodePtr &child) const =0

  Gets the child node which follows the node passed in.
- virtual IDOMNodePtr getPreviousChild (const IDOMNodePtr &child) const =0

  Gets the child node which precedes the node passed in.
- virtual IDOMNodePtr getPreviousSibling () const =0

  Retrieves the node's previous sibling node.
- virtual IDOMNodePtr getNextSibling () const =0

  Retrieves node's next sibling node.
- virtual void appendChild (const IDOMNodePtr &child)=0

  Appends a node to the end of the node's child list.
- virtual void insertChild (const IDOMNodePtr &ptrPreviousSibling, const IDOMNodePtr &child, bool bCheck↩ Complete=true)=0

  Insert a child node after ptrPreviousSibling.
- virtual IDOMNodePtr extractChild (const IDOMNodePtr &child)=0

  Extracts (that is, finds and removes) a child node from the node children. After extraction the child node is no longer a part of the DOM. If no node is specified, the first available node will be extracted from the node's children.
- virtual void replaceChild (const IDOMNodePtr &oldChild, const IDOMNodePtr &newChild)=0

  Replaces the child node with another.
- virtual bool isComplete () const =0

  Signals the completeness of the node.
  A complete node is one that has no more children to be added to it.
- virtual void **setComplete** ()=0

Sets the node's completeness status to "true".

- virtual IDOMNodeFlags * getFlags ()=0

  Retrieves the node's flags property.
- virtual void setParentNode (const IDOMNodePtr &ptrParent)=0

  Sets the parent node.
- virtual void setPreviousSibling (const IDOMNodePtr &ptrPreviousSibling)=0

  Sets the previous sibling node.
- virtual void setNextSibling (const IDOMNodePtr &ptrNextSibling)=0

  Sets the next sibling node.
- virtual bool isAncestor (const IDOMNodePtr &ptrCandidate)=0

  Function tests whether a candidate node is a descendant of the node.
- virtual FRect getBounds (bool applyTransform=true, bool applyClip=true)

  Find the conservative bounding box of the marking content of the node.
- virtual bool copyNodeData (IDOMNode *pSourceNode)=0

  Copy the properties collection, the flags and the DOM ID from the given source node to this one.
- virtual IDOMNodePtr cloneNode (IEDLClassFactory *pFactory) const =0

  Simpified node cloning. An exception of type IEDLError will be thrown on failure.
- virtual IDOMNodePtr cloneTree (IEDLClassFactory *pFactory) const =0

  Clone the tree of nodes beginning at this node. An exception of type IEDLError will be thrown on failure.
- virtual void cloneTreeAndAppend (IEDLClassFactory *pFactory, const IDOMNodePtr &dest) const =0

  Clone the tree of nodes beginning at this node, and append the result to the destination tree.
- virtual void **completeTree** ()=0

  Mark the entire tree from this point as complete. You should not ordinarily need to call this function.
- virtual void **removeCompleteFlagFromTree** ()=0

  Mark the entire tree from this point as complete.
- virtual void findChildrenOfType (eDOMNodeType type, CDOMNodeVect &nodes, bool searchForms=false)=0

  Find all children of this node with the given type, appending to the given vector. Does not descend into brushes.
- virtual void walkTree (WalkTreeFunc func, void *priv, bool descendIntoBrushes=false, bool descendInto↩ Forms=false)=0

  Walk through the DOM calling a given function on each node. The function is allowed to:
- virtual void notifyOnDestruct (NodeDeleteFunc func, void *priv)=0

  Register interest in being told when this node is about to be destroyed.
- virtual void unregisterNotify (NodeDeleteFunc func, void *priv)=0

  Unregister interest in being told when this node is about to be destroyed.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves class id of IDOMTransparencyGroup.
- static EDL_API IDOMTransparencyGroupPtr create (IEDLClassFactory *pFactory, const FMatrix &transform=FMatrix(), const IDOMPathGeometryPtr &clip=IDOMPathGeometryPtr(), float opacity=1.0f, const IDOMBrushPtr &opacityMask=IDOMBrushPtr(), eBlendMode blendMode=eBlendModeNormal, const IDOMColorSpacePtr &colorSpace=IDOMColorSpacePtr(), bool isolated=false, bool knockout=false)

  Simplified creation function for IDOMGroup. Throws an IEDLError exception on failure.

Static Public Member Functions inherited from IDOMGroup

- static EDL_API IDOMGroupPtr create (IEDLClassFactory *pFactory, const FMatrix &transform=FMatrix(), const IDOMPathGeometryPtr &clip=IDOMPathGeometryPtr())

  Simplified creation function for IDOMGroup. Throws an IEDLError exception on failure.
- static const CClassID & classID ()

  Retrieves class id of IDOMGroup.

Static Public Member Functions inherited from IDOMNode

- static EDL_API FMatrix effectiveTransformationOfNode (const IDOMNodePtr &node)

  Attempt to find the effective transformation matrix external to the specified node relative to either a containing page or ultimate parent.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.284.1 Detailed Description

IDOMTransparencyGroup interface. Analogous to PDF Transparency groups.

7.284.2 Member Function Documentation

classID()

```
static const CClassID & IDOMTransparencyGroup::classID ( )  [inline], [static]
```

Retrieves class id of IDOMTransparencyGroup.

Returns

CClassID class id of the element

create()

```
static EDL_API IDOMTransparencyGroupPtr IDOMTransparencyGroup::create (
            IEDLClassFactory * pFactory,
            const FMatrix & transform = FMatrix(),
            const IDOMPathGeometryPtr & clip = IDOMPathGeometryPtr(),
            float opacity = 1.0f,
            const IDOMBrushPtr & opacityMask = IDOMBrushPtr(),
            eBlendMode blendMode = eBlendModeNormal,
            const IDOMColorSpacePtr & colorSpace = IDOMColorSpacePtr(),
            bool isolated = false,
            bool knockout = false )  [static]
```

Simplified creation function for IDOMGroup. Throws an IEDLError exception on failure.

Parameters

pFactory	The EDL class factory
transform	The desired render transform
clip	The desired clip
opacity	The desired opacity
opacityMask	The desired opacity mask
blendMode	The desired blend mode
colorSpace	The desired group color space
isolated	Group isolated parameter
knockout	Group knockout parameter

Returns

IDOMTransparencyGroupPtr A smart pointer to the new transprency group

getBlendMode()

```
virtual eBlendMode IDOMTransparencyGroup::getBlendMode ( ) const  [pure virtual]
```

Get the blend mode to be used for compositing this group with the backdrop.

Returns

eBlendMode The blend mode

getColorSpace()

```
virtual IDOMColorSpacePtr IDOMTransparencyGroup::getColorSpace ( ) const  [pure virtual]
```

Get the group colorspace.

Returns

IDOMColorSpacePtr The group color space or NULL of no colour space is present.

getIsIsolated()

```
virtual bool IDOMTransparencyGroup::getIsIsolated ( ) const  [pure virtual]
```

Is the group an isolated group? See section 7.5.5 of the PDF 1.7 spec for details.

Returns

bool True if the group is isolated.

getIsKnockout()

```
virtual bool IDOMTransparencyGroup::getIsKnockout ( ) const  [pure virtual]
```

Is the group a knockout group? See section 7.5.5 of the PDF 1.7 spec for details.

Returns

bool True if the group is isolated.

getOpacity()

```
virtual float IDOMTransparencyGroup::getOpacity ( ) const  [pure virtual]
```

Get the group alpha/opacity.

Returns

float The group opacity.

getOpacityMask()

```
virtual IDOMBrushPtr IDOMTransparencyGroup::getOpacityMask ( ) const  [pure virtual]
```

Retrieves smart pointer to opacity mask.

Returns

IDOMBrushPtr The opacity mask or NULL of no opacity mask is present.

setBlendMode()

```
virtual void IDOMTransparencyGroup::setBlendMode (
            eBlendMode blendMode )  [pure virtual]
```

Set the blend mode to be used for compositing this group with the backdrop.

Parameters

blendMode	The blend mode for the group.

setColorSpace()

```
virtual void IDOMTransparencyGroup::setColorSpace (
            const IDOMColorSpacePtr & colorSpace )  [pure virtual]
```

Set the group colorspace.

This is an optional entry in most circumstances, and may not be used in all situations. This is only used for the following situations:

- If the group is isolated (optional)
- If the group is used in a SoftMask Brush, with the soft mask using the luminosity of the group.

Not all color spaces can be used. See section 7.5.5 of the PDF 1.7 specification for further information.

Parameters

colorSpace	The desired color space

setIsIsolated()

```
virtual void IDOMTransparencyGroup::setIsIsolated (
            bool isolated )  [pure virtual]
```

Set whether the group is isolated. See section 7.5.5 of the PDF 1.7 spec for details.

Parameters

isolated	New value of isolated

setIsKnockout()

```
virtual void IDOMTransparencyGroup::setIsKnockout (
            bool knockout )  [pure virtual]
```

Set whether the group is a knockout group. See section 7.5.5 of the PDF 1.7 spec for details.

Parameters

knockout	New value of knockout

setOpacity()

```
virtual void IDOMTransparencyGroup::setOpacity (
            float opacity )  [pure virtual]
```

Set the group opacity.

Parameters

opacity	The desired opacity.

setOpacityMask()

```
virtual void IDOMTransparencyGroup::setOpacityMask (
            const IDOMBrushPtr & ptrOpacityMask )  [pure virtual]
```

Sets opacity mask.

Parameters

ptrOpacityMask	Smart pointer to brush

The documentation for this class was generated from the following file:

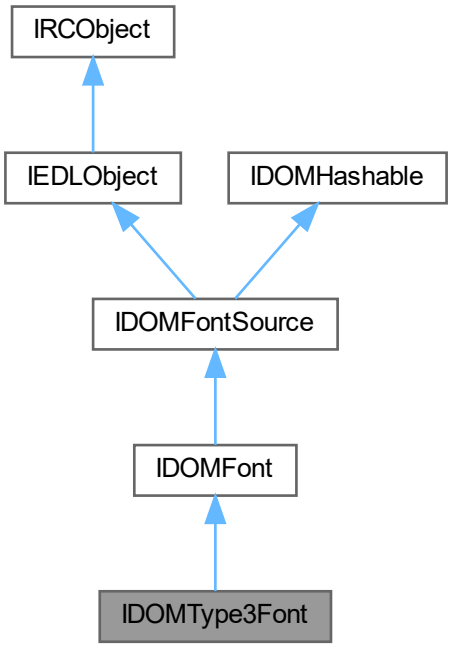
- idomgroup.h

7.285 IDOMType3Font Class Reference

Representation of a PostScript/PDF Type 3 Font. At present, the stream cannot be set, only retrieved.

```
#include <idomfont.h>
```

Inheritance diagram for IDOMType3Font:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual DOMid getId () const =0

  Retrieves the unique ID of this font. This ID is allocated on creation. Clones of this font will receive a new id.
- virtual bool hasGlyph (const EDLSysString &glyphName) const =0

  Determines if the font already has a glyph with the given name.
- virtual IDOMGlyphPtr getGlyph (const EDLSysString &glyphName) const =0

  Retrieves a glyph of the given name. An exception with code EDL_ERR_TYPE3_GLYPH_NOT_FOUND will be thrown if no such glyph is present.
- virtual bool hasGlyph (uint32 codePoint) const =0

  Determines if the font already has a glyph with the given unicode codepoint.
- virtual IDOMGlyphPtr getGlyph (uint32 codePoint) const =0

  Retrieves a glyph of the given unicode codepoint, if it exists. An exception with code EDL_ERR_TYPE3_GLYPH_↩ NOT_FOUND will be thrown if no such glyph is present.
- virtual bool hasGlyph (IDOMGlyph::GlyphID glyphID) const =0

  Determines if the font already has a glyph with the given glyph ID.
- virtual IDOMGlyphPtr getGlyph (IDOMGlyph::GlyphID glyphID) const =0

  Retrieves a glyph of the given GlyphID, if it exists. An exception with code EDL_ERR_TYPE3_GLYPH_NOT_FOUND will be thrown if no such glyph is present.

- virtual CDOMGlyphVect getGlyphs () const =0

  Obtain a vector of all of the glyphs present in this font.
- virtual void **deleteGlyphs** ()=0

  Delete the glyph collection from the font.
- virtual void addGlyph (const IDOMGlyphPtr &glyph)=0

  Adds a glyph of the given name to the font.
- virtual FRect getBBox ()=0

  Returns font's bounding box.
- virtual JawsMako::IPDFDictionaryPtr getFontDictionary () const =0

  Fetch the font dictionary. Do not edit the returned dictionary.

Public Member Functions inherited from IDOMFont

- virtual eFontType getFontType () const =0

  Gets the font type. See eFontType for more information on font types.
- virtual IDOMFontSourcePtr getFontSource () const =0

  Get the font source of this font.
- virtual void setFontSource (const IDOMFontSourcePtr &fontSource)=0

  Sets the font source for this font.
- virtual IInputStreamPtr getFontBaseStream () const =0

  Return the base stream for this font, obtaining it from the font source.
- virtual void getCharacterMap (CCharacterMap &characterMap, uint32 fontIndex=0U)=0

  Get the character map for this font.

Public Member Functions inherited from IDOMFontSource

- virtual eFontSourceType getFontSourceType () const =0

  Gets the font source type.
- virtual const EDLSysString & determineUri () const =0

  Determines the URI based on the font source (underlying font sources may be searched)

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves class id of IDOM.

Static Public Member Functions inherited from IDOMFont

- static const CClassID & classID ()

  Retrieves class id of IDOMFont.

Static Public Member Functions inherited from IDOMFontSource

- static const CClassID & classID ()

  Retrieves the class id of IDOMFontSource.

Additional Inherited Members

Public Types inherited from IDOMFont

- enum eFontType {
  eFontTypeUnknown , eFontTypeOpenType , eFontType3 , eFontTypePCLXL ,
  eFontTypePCL5 }

  type used to uniquely identify the type of font
- typedef uint32 **CharCode**

  type used to uniquely identify a character code
- typedef std::map< CharCode, IDOMGlyph::GlyphID > **CCharacterMap**

  Map type used for storing character code to glyph ID mappings.

Public Types inherited from IDOMFontSource

- enum eFontSourceType { eFontSourceTypeNone , eFontSourceTypeStreamFilter , eFontSourceTypeStream , eFontSourceTypeFont }

  type used to uniquely identify the source type of a font

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.285.1 Detailed Description

Representation of a PostScript/PDF Type 3 Font. At present, the stream cannot be set, only retrieved.

7.285.2 Member Function Documentation

addGlyph()

```
virtual void IDOMType3Font::addGlyph (
            const IDOMGlyphPtr & glyph )  [pure virtual]
```

Adds a glyph of the given name to the font.

Parameters

glyph	The glyph

classID()

```
static const CClassID & IDOMType3Font::classID ( )  [inline], [static]
```

Retrieves class id of IDOM.

Returns

CClassID class id of the element

getBBox()

```
virtual FRect IDOMType3Font::getBBox ( )  [pure virtual]
```

Returns font's bounding box.

Returns

FRect The bounding box

getFontDictionary()

```
virtual JawsMako::IPDFDictionaryPtr IDOMType3Font::getFontDictionary ( ) const  [pure virtual]
```

Fetch the font dictionary. Do not edit the returned dictionary.

Returns

IPDFDictionaryPtr The font dictionary

getGlyph() [1/3]

```
virtual IDOMGlyphPtr IDOMType3Font::getGlyph (
            const EDLSysString & glyphName ) const  [pure virtual]
```

Retrieves a glyph of the given name. An exception with code EDL_ERR_TYPE3_GLYPH_NOT_FOUND will be thrown if no such glyph is present.

Parameters

glyphName	The glyph name

Returns

IDOMGlyphPtr The glyph

getGlyph() [2/3]

```
virtual IDOMGlyphPtr IDOMType3Font::getGlyph (
            IDOMGlyph::GlyphID glyphID ) const  [pure virtual]
```

Retrieves a glyph of the given GlyphID, if it exists. An exception with code EDL_ERR_TYPE3_GLYPH_NOT_↩ FOUND will be thrown if no such glyph is present.

Parameters

glyphID	The glyphID

Returns

IDOMGlyphPtr The glyph

getGlyph() [3/3]

```
virtual IDOMGlyphPtr IDOMType3Font::getGlyph (
            uint32 codePoint ) const  [pure virtual]
```

Retrieves a glyph of the given unicode codepoint, if it exists. An exception with code EDL_ERR_TYPE3_GLYPH↩ _NOT_FOUND will be thrown if no such glyph is present.

Parameters

codePoint	The unicode codepoint

Returns

IDOMGlyphPtr The glyph

getGlyphs()

```
virtual CDOMGlyphVect IDOMType3Font::getGlyphs ( ) const  [pure virtual]
```

Obtain a vector of all of the glyphs present in this font.

Returns

CDOMGlyphVect The glyphs

getId()

```
virtual DOMid IDOMType3Font::getId ( ) const  [pure virtual]
```

Retrieves the unique ID of this font. This ID is allocated on creation. Clones of this font will receive a new id.

Returns

DOMid The DOM ID

hasGlyph() [1/3]

```
virtual bool IDOMType3Font::hasGlyph (
            const EDLSysString & glyphName ) const  [pure virtual]
```

Determines if the font already has a glyph with the given name.

Parameters

glyphName	The glyph name

Returns

bool true if the glyph is present, false otherwise

hasGlyph() [2/3]

```
virtual bool IDOMType3Font::hasGlyph (
            IDOMGlyph::GlyphID glyphID ) const  [pure virtual]
```

Determines if the font already has a glyph with the given glyph ID.

Parameters

glyphID	The glyphID

Returns

bool true if the glyph is present, false otherwise

hasGlyph() [3/3]

```
virtual bool IDOMType3Font::hasGlyph (
            uint32 codePoint ) const  [pure virtual]
```

Determines if the font already has a glyph with the given unicode codepoint.

Parameters

codePoint	The unicode codepoint

Returns

bool true if the glyph is present, false otherwise

The documentation for this class was generated from the following file:

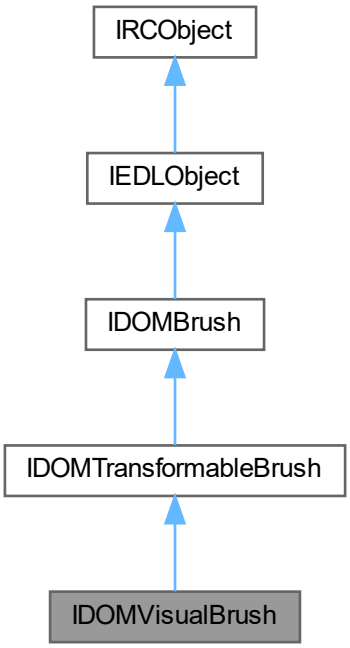
- idomfont.h

7.286 IDOMVisualBrush Class Reference

A visual brush is used to fill a region with a vector drawing.

`#include <idombrush.h>`

Inheritance diagram for IDOMVisualBrush:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual eTilingMode getTileMode () const =0

  Retrieves the tiling mode value of the visual brush.
- virtual void setTileMode (eTilingMode tm)=0

  Sets tiling mode value of the visual brush.
- virtual eViewUnits getViewBoxUnits () const =0

  Retrieves the viewbox units used by the image brush. Currently, only absolute units are supported.
- virtual void setViewBoxUnits (eViewUnits vu)=0

  Sets the viewbox units value of the image brush. Currently, only absolute units are supported.
- virtual eViewUnits getViewPortUnits () const =0

  Retrieves the viewport units value of the image brush. Currently, only absolute units are supported.
- virtual void setViewPortUnits (eViewUnits vu)=0

  Sets the viewport units used for the image brush. Currently, only absolute units are supported.
- virtual const FRect & getViewBox () const =0

  Retrieves the viewbox rectangle.
- virtual void setViewBox (const FRect &vb)=0

  Sets viewbox rectangle.
- virtual const FRect & getViewPort () const =0

  Retrieves the viewport rectangle.
- virtual void setViewPort (const FRect &vp)=0

  Sets the viewport rectangle.
- virtual IDOMNodePtr getVisual () const =0

  Retrieves smart pointer to the visual node (path, glyphs, group or canvas node) used to specify the source for the visual brush.
- virtual void setVisual (const IDOMNodePtr &ptrVisual)=0

  Sets the visual node (path, glyphs or canvas node) used to specify the source for the visual brush.
- virtual IDOMTilingPatternBrushPtr getEquivalentTilingBrush (IEDLClassFactory *pFactory)=0

  Gets an equivalent IDOMTilingPattern brush. If the receiver has a tile mode of eNoTile, this call will fail.
- virtual IDOMVisualBrushPtr getEquivalentSimpleVisualBrush (IEDLClassFactory *pFactory)=0

  Gets an equivalent visual brush where any flip tile mode is simplified to simple tiling. Useful for situations where flipping cannot be handled, but simple visual brushes can.

Public Member Functions inherited from IDOMTransformableBrush

- virtual const FMatrix & getRenderTransform () const =0

  Retrieves the render transform matrix.
- virtual void setRenderTransform (const FMatrix &matrix)=0

  Sets the render transform matrix.

Public Member Functions inherited from IDOMBrush

- virtual eBrushType getBrushType () const =0

  Retrieves the type of the brush.
- virtual float getOpacity () const =0

  Retrieves the opacity value of the brush element.
- virtual void setOpacity (float opc)=0

  Sets the opacity value of a brush element.
- virtual IDOMBrushPtr getAdjustedForUseInTransformedNode (IEDLClassFactory *pFactory, const FMatrix &nodeTransform)

  Get a version of this brush adjusted for use inside a node with the given transform.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static EDL_API IDOMVisualBrushPtr create (IEDLClassFactory *pFactory, const IDOMNodePtr &visual, const FRect &viewBox, const FRect &viewPort, const FMatrix &renderTransform=FMatrix(), float opacity=1.0f, eTilingMode tileMode=eNoTile)

  Simplified creator for a visual brush. Throws an IEDLError on failure.
- static const CClassID & classID ()

  Retrieves the class id of IDOMVisualBrush.

Additional Inherited Members

Public Types inherited from IDOMBrush

- enum eBrushType {
  eSolidColor , eLinearGradient , eRadialGradient , eImage ,
  eMasked , eVisual , eSoftMask , eTilingPattern ,
  eType1ShadingPattern , eType2ShadingPattern , eType3ShadingPattern , eType4567ShadingPattern ,
  eNull }

  Brush type enumeration.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.286.1 Detailed Description

A visual brush is used to fill a region with a vector drawing.

The drawing may be specified as either a visual brush property or as a resource reference. The drawing content may include exactly one Canvas, Path, or Glyphs node and that node's child and descendant nodes.

Visual brushes share a number of tile-related properties with image brushes.

See also

IDOMImageBrush

7.286.2 Member Function Documentation

classID()

```
static const CClassID & IDOMVisualBrush::classID ( )  [inline], [static]
```

Retrieves the class id of IDOMVisualBrush.

Returns

CClassID The class id of IDOMVisualBrush.

create()

```
static EDL_API IDOMVisualBrushPtr IDOMVisualBrush::create (
            IEDLClassFactory * pFactory,
            const IDOMNodePtr & visual,
            const FRect & viewBox,
            const FRect & viewPort,
            const FMatrix & renderTransform = FMatrix(),
            float opacity = 1.0f,
            eTilingMode tileMode = eNoTile )  [static]
```

Simplified creator for a visual brush. Throws an IEDLError on failure.

Parameters

pFactory	The factory to use.
visual	The node to use as the visual.
viewBox	The desired view box.
viewPort	The desired view port.
renderTransform	The desired render transform.
opacity	The desired brush opacity.
tileMode	The desired tile mode.

Returns

IDOMVisualBrushPtr The new visual brush.

getEquivalentSimpleVisualBrush()

```
virtual IDOMVisualBrushPtr IDOMVisualBrush::getEquivalentSimpleVisualBrush (
            IEDLClassFactory * pFactory )  [pure virtual]
```

Gets an equivalent visual brush where any flip tile mode is simplified to simple tiling. Useful for situations where flipping cannot be handled, but simple visual brushes can.

Parameters

pFactory	A pointer to an EDL class factory.

Returns

IDOMVisualBrushPtr The equivalent visual brush, or this brush if no changes are required. This brush will be cached, so do not edit; make a clone if the returned brush needs to be changed.

getEquivalentTilingBrush()

```
virtual IDOMTilingPatternBrushPtr IDOMVisualBrush::getEquivalentTilingBrush (
            IEDLClassFactory * pFactory )  [pure virtual]
```

Gets an equivalent IDOMTilingPattern brush. If the receiver has a tile mode of eNoTile, this call will fail.

Parameters

pFactory	A pointer to an EDL class factory

Returns

IDOMTilingPatternBrushPtr The equivalent tiling brush, or NULL if a tiling brush cannot be created for this brush. This brush will be cached, so do not edit; make a clone if the returned brush needs to be changed.

getTileMode()

```
virtual eTilingMode IDOMVisualBrush::getTileMode ( ) const  [pure virtual]
```

Retrieves the tiling mode value of the visual brush.

Returns

eTilingMode The tiling mode

See also

eTilingMode

getViewBox()

```
virtual const FRect & IDOMVisualBrush::getViewBox ( ) const  [pure virtual]
```

Retrieves the viewbox rectangle.

The viewbox specifies the portion of a source image or visual to be rendered to the page as a tile, whose size and location are determined by the image brush's viewport. The tile is then used to fill the geometry specified by the parent element according to the image brush's tile mode. The ViewBox can specify a region larger than the image itself, including negative values. The view box specifies the position and dimension of the brush's source content. It is specified by four comma-separated real numbers (x, y, width, height) where width and height are non-negative.

Returns

FRect The viewbox

getViewBoxUnits()

```
virtual eViewUnits IDOMVisualBrush::getViewBoxUnits ( ) const  [pure virtual]
```

Retrieves the viewbox units used by the image brush. Currently, only absolute units are supported.

See also

getViewBox()

setViewBox()

Returns

eViewUnits The viewbox units.

getViewPort()

```
virtual const FRect & IDOMVisualBrush::getViewPort ( ) const  [pure virtual]
```

Retrieves the viewport rectangle.

The viewport specifies the dimensions and location of the initial tile that will be filled with the specified image or visual fragment. It is defined in the current effective coordinate space. It is specified by four comma separated real numbers (x, y, width, height) where width and height are non-negative.

Returns

FRect The viewport rectangle

getViewPortUnits()

```
virtual eViewUnits IDOMVisualBrush::getViewPortUnits ( ) const  [pure virtual]
```

Retrieves the viewport units value of the image brush. Currently, only absolute units are supported.

See also

getViewPort()

setViewPort()

Returns

eViewUnits The viewport units.

getVisual()

```
virtual IDOMNodePtr IDOMVisualBrush::getVisual ( ) const  [pure virtual]
```

Retrieves smart pointer to the visual node (path, glyphs, group or canvas node) used to specify the source for the visual brush.

Returns

IDOMNodePtr The visual

setTileMode()

```
virtual void IDOMVisualBrush::setTileMode (
            eTilingMode tm )  [pure virtual]
```

Sets tiling mode value of the visual brush.

Parameters

tm	The tiling mode value

See also

eTilingMode

setViewBox()

```
virtual void IDOMVisualBrush::setViewBox (
            const FRect & vb )  [pure virtual]
```

Sets viewbox rectangle.

The viewbox specifies the portion of a source image or visual to be rendered to the page as a tile. The tile is then used to fill the geometry specified by the parent element according to the TileMode() function. The viewbox can specify a region larger than the image itself, including negative values. The viewbox specifies the position and dimension of the brush's source content. It is specified by four comma-separated real numbers (x, y, width, height) where width and height are non-negative.

Parameters

vb	The viewbox rectangle

setViewBoxUnits()

```
virtual void IDOMVisualBrush::setViewBoxUnits (
            eViewUnits vu )  [pure virtual]
```

Sets the viewbox units value of the image brush. Currently, only absolute units are supported.

See also

getViewBox()

setViewBox()

Parameters

vu	The viewbox units value

setViewPort()

```
virtual void IDOMVisualBrush::setViewPort (
            const FRect & vp )  [pure virtual]
```

Sets the viewport rectangle.

The viewport specifies the dimensions and location of the initial tile that will be filled with the specified image or visual fragment. It is defined in the current effective coordinate space. It is specified by four comma separated real numbers (x, y, width, height) where width and height are non-negative.

Parameters

vp	The viewport rectangle

setViewPortUnits()

```
virtual void IDOMVisualBrush::setViewPortUnits (
            eViewUnits vu )  [pure virtual]
```

Sets the viewport units used for the image brush. Currently, only absolute units are supported.

See also

getViewPort()

setViewPort()

Parameters

vu	The new viewport units value

setVisual()

```
virtual void IDOMVisualBrush::setVisual (
            const IDOMNodePtr & ptrVisual )  [pure virtual]
```

Sets the visual node (path, glyphs or canvas node) used to specify the source for the visual brush.

Parameters

ptrVisual	Pointer to the visual node (path, glyphs or canvas node) used to specify the source for the visual brush.

The documentation for this class was generated from the following file:

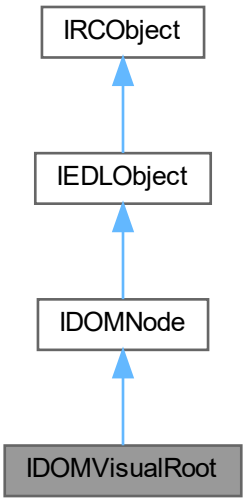
- idombrush.h

7.287 IDOMVisualRoot Class Reference

IDOMVisualRoot interface.

`#include <idombrush.h>`

Inheritance diagram for IDOMVisualRoot:



Static Public Member Functions

- static const CClassID & classID ()

  Retrieves the class ID of IDOMVisualRoot.

Static Public Member Functions inherited from IDOMNode

- static EDL_API FMatrix effectiveTransformationOfNode (const IDOMNodePtr &node)

  Attempt to find the effective transformation matrix external to the specified node relative to either a containing page or ultimate parent.

Additional Inherited Members

Public Member Functions inherited from IDOMNode

- virtual ~**IDOMNode** ()

  virtual destructor
- virtual DOMid getDOMid () const =0

  Retrieves the node ID.
- virtual void setDOMid (DOMid id)=0

  Sets the node ID.
- virtual eDOMNodeType getNodeType () const =0

  Retrieves the DOM node type.
- virtual bool getProperty (const EDLSysString &propertyName, PValue &propertyValue) const =0

  Retrieves the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.
- virtual void setProperty (const EDLSysString &propertyName, const PValue &propertyValue)=0

  Sets the value of a property. The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.
- virtual void removeProperty (const EDLSysString &propertyName)=0

  Removes property.
- virtual IEDLSysStringCollectionEnumPtr getPropertyCollectionEnum ()=0

  Retrieves a navigable list of the property names stored on this node.
- virtual bool hasChildNodes () const =0

  Function that indicates whether this node is a parent to other nodes.
- virtual IDOMNodePtr getParentNode () const =0

  Gets the parent node of this node.
- virtual IDOMNodePtr getFirstChild () const =0

  Gets the first child node of this node.
- virtual IDOMNodePtr getLastChild () const =0

  Gets the last child node of this node.
- virtual IDOMNodePtr getNextChild (const IDOMNodePtr &child) const =0

  Gets the child node which follows the node passed in.
- virtual IDOMNodePtr getPreviousChild (const IDOMNodePtr &child) const =0

  Gets the child node which precedes the node passed in.
- virtual IDOMNodePtr getPreviousSibling () const =0

  Retrieves the node's previous sibling node.
- virtual IDOMNodePtr getNextSibling () const =0

  Retrieves node's next sibling node.

- virtual void appendChild (const IDOMNodePtr &child)=0

  Appends a node to the end of the node's child list.
- virtual void insertChild (const IDOMNodePtr &ptrPreviousSibling, const IDOMNodePtr &child, bool bCheck↩ Complete=true)=0

  Insert a child node after ptrPreviousSibling.
- virtual IDOMNodePtr extractChild (const IDOMNodePtr &child)=0

  Extracts (that is, finds and removes) a child node from the node children. After extraction the child node is no longer a part of the DOM. If no node is specified, the first available node will be extracted from the node's children.
- virtual void replaceChild (const IDOMNodePtr &oldChild, const IDOMNodePtr &newChild)=0

  Replaces the child node with another.
- virtual bool isComplete () const =0

  Signals the completeness of the node.
  A complete node is one that has no more children to be added to it.
- virtual void **setComplete** ()=0

  Sets the node's completeness status to "true".
- virtual IDOMNodeFlags ∗ getFlags ()=0

  Retrieves the node's flags property.
- virtual void setParentNode (const IDOMNodePtr &ptrParent)=0

  Sets the parent node.
- virtual void setPreviousSibling (const IDOMNodePtr &ptrPreviousSibling)=0

  Sets the previous sibling node.
- virtual void setNextSibling (const IDOMNodePtr &ptrNextSibling)=0

  Sets the next sibling node.
- virtual bool isAncestor (const IDOMNodePtr &ptrCandidate)=0

  Function tests whether a candidate node is a descendant of the node.
- virtual FRect getBounds (bool applyTransform=true, bool applyClip=true)

  Find the conservative bounding box of the marking content of the node.
- virtual bool copyNodeData (IDOMNode ∗pSourceNode)=0

  Copy the properties collection, the flags and the DOM ID from the given source node to this one.
- virtual IDOMNodePtr cloneNode (IEDLClassFactory ∗pFactory) const =0

  Simpified node cloning. An exception of type IEDLError will be thrown on failure.
- virtual IDOMNodePtr cloneTree (IEDLClassFactory ∗pFactory) const =0

  Clone the tree of nodes beginning at this node. An exception of type IEDLError will be thrown on failure.
- virtual void cloneTreeAndAppend (IEDLClassFactory ∗pFactory, const IDOMNodePtr &dest) const =0

  Clone the tree of nodes beginning at this node, and append the result to the destination tree.
- virtual void **completeTree** ()=0

  Mark the entire tree from this point as complete. You should not ordinarily need to call this function.
- virtual void **removeCompleteFlagFromTree** ()=0

  Mark the entire tree from this point as complete.
- virtual void findChildrenOfType (eDOMNodeType type, CDOMNodeVect &nodes, bool searchForms=false)=0

  Find all children of this node with the given type, appending to the given vector. Does not descend into brushes.
- virtual void walkTree (WalkTreeFunc func, void ∗priv, bool descendIntoBrushes=false, bool descendInto↩ Forms=false)=0

  Walk through the DOM calling a given function on each node. The function is allowed to:
- virtual void notifyOnDestruct (NodeDeleteFunc func, void ∗priv)=0

  Register interest in being told when this node is about to be destroyed.
- virtual void unregisterNotify (NodeDeleteFunc func, void ∗priv)=0

  Unregister interest in being told when this node is about to be destroyed.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.287.1 Detailed Description

IDOMVisualRoot interface.

7.287.2 Member Function Documentation

classID()

```
static const CClassID & IDOMVisualRoot::classID ( )  [inline], [static]
```

Retrieves the class ID of IDOMVisualRoot.

Returns

CClassID The class ID of IDOMVisualRoot.

The documentation for this class was generated from the following file:

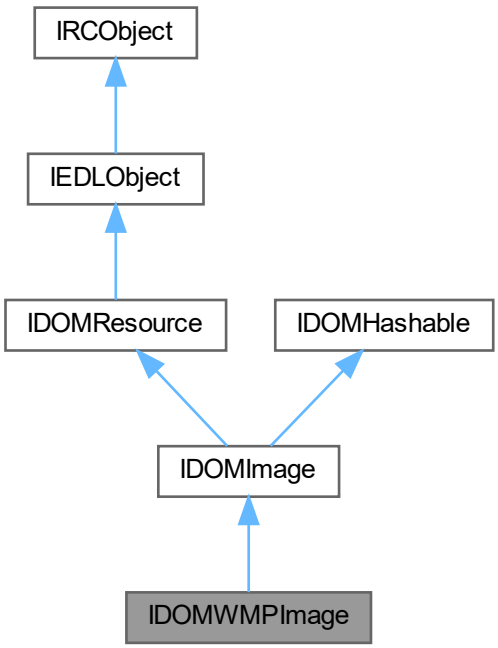
- idombrush.h

7.288 IDOMWMPImage Class Reference

IDOMWMPImage interface.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMWMPImage:



Static Public Member Functions

- static const CClassID & classID ()

  Retrieves class id of IDOMWMPImage.

Additional Inherited Members

Public Member Functions inherited from IDOMImage

- virtual IImageDecoderPtr createImageDecoder (IEDLClassFactory *factory, const IDOMImagePropertiesPtr &imageProperties)=0

  Creates a properly initialized image decoder object that reads from an inputstream that is specific to that image format.
- virtual IImageFramePtr getImageFrame (IEDLClassFactory *factory)

  Fetch the image frame; convenience.
- virtual IImageEncoderPtr createImageEncoder (const ISessionPtr &session, const IOutputStreamPtr &imageDest, const IDOMImagePropertiesPtr &imageProperties)=0

  Creates a properly initialized image encoder object that writes to an outputstream that is specific to that image format.

- virtual IDOMImagePropertiesPtr getImageProperties ()=0

  Returns an object that stores the properties for this image object. The properties can then be inspected (or more added) by clients that need to manipulate the image resource.
- virtual eDOMImageType getImageType ()=0

  Retrieves the image type.
- virtual bool getIsRendered ()=0

  Determine if the image is as a result of rendering. This is indicated if the image type is eDITRendered or if the image explicitly notes this is the case (such as for IDOMPDFImage).
- virtual IDOMImagePtr getImageWithSubstitutedColorSpace (IEDLClassFactory *factory, const IDOMColor↩ SpacePtr &colorSpace)

  Obtain an image that is the same as this image, but with the colorspace substituted for another.

Public Member Functions inherited from IDOMResource

- virtual IInputStreamPtr getStream () const =0

  Retrieves the resource stream.
- virtual void setStream (const IInputStreamPtr &stream)=0

  Sets the resource stream for the node.
- virtual uint64 getStreamLength () const =0

  Retrieves the stream length, if it is available.
- virtual const EDLSysString & getUri () const =0

  Retrieves the resource URI.
- virtual void setUri (const EDLSysString &uri)=0

  Sets the resource URI.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual bool hash (uint64 &hash)=0

  Retrieve a hash for this object.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.288.1 Detailed Description

IDOMWMPImage interface.

7.288.2 Member Function Documentation

classID()

```
static const CClassID & IDOMWMPImage::classID ( )  [inline], [static]
```

Retrieves class id of IDOMWMPImage.

Returns

CClassID Class id of the element

The documentation for this class was generated from the following file:

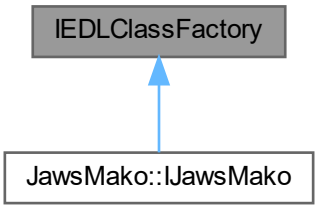
- idomimageresource.h

7.289 IEDLClassFactory Class Reference

EDL Factory Interface allows one part of the EDL infrastructure to register class creation methods identified by either GUIDs and /or names (strings) and then another part of the EDL infrastructure to request the creation of instances of one or more of these classes by quoting the same GUID or name.

```
#include <iedlfactory.h>
```

Inheritance diagram for IEDLClassFactory:



Public Member Functions

- virtual bool registerNamedClass (const EDLSysString &strName, const CClassID &id, bool overwrite=true)=0

  Register a GUID under a string name.
- virtual bool findNamedClass (const EDLSysString &strName, CClassID &id)=0

  Retrieve GUID registered under the string name.
- virtual bool registerClass (const CClassID &id, creatorFunc f)=0

  Register a GUID with creator function.
- virtual IEDLObjectPtr createInstance (const CClassID &id, CClassParams *pParams=NULL)=0

  Creates the registered class by CClassID.
- virtual IEDLObjectPtr createInstanceJawsMako (const CClassID &id, CClassParams *pParams=NULL, IEDLClassFactory *factory=NULL)=0

  Creates the registered class by CClassID.
- virtual IEDLObjectPtr getSingleton (const CClassID &id)=0

  Creates the EDL singleton (for example FontLibrary or ColorManager)

7.289.1 Detailed Description

EDL Factory Interface allows one part of the EDL infrastructure to register class creation methods identified by either GUIDs and /or names (strings) and then another part of the EDL infrastructure to request the creation of instances of one or more of these classes by quoting the same GUID or name.

EDL Class Factory

7.289.2 Member Function Documentation

createInstance()

```
virtual IEDLObjectPtr IEDLClassFactory::createInstance (
            const CClassID & id,
            CClassParams * pParams = NULL )  [pure virtual]
```

Creates the registered class by CClassID.

Parameters

id	Class GUID
pParams	Parameters for class initialization

Returns

IEDLObjectPtr A smart pointer to EDLObject interface to newly created class

createInstanceJawsMako()

```
virtual IEDLObjectPtr IEDLClassFactory::createInstanceJawsMako (
            const CClassID & id,
```

```
        CClassParams * pParams = NULL,
        IEDLClassFactory * factory = NULL )  [pure virtual]
```

Creates the registered class by CClassID.

Parameters

id	Class GUID
pParams	Parameters for class initialization
factory	IEDLClassFactory Pointer to JawsMako

Returns

IEDLObjectPtr A smart pointer to EDLObject interface to newly created class

findNamedClass()

```
virtual bool IEDLClassFactory::findNamedClass (
            const EDLSysString & strName,
            CClassID & id )  [pure virtual]
```

Retrieve GUID registered under the string name.

Parameters

strName	Name to find
id	Placeholder for result

Returns

bool True on success

getSingleton()

```
virtual IEDLObjectPtr IEDLClassFactory::getSingleton (
            const CClassID & id )  [pure virtual]
```

Creates the EDL singleton (for example FontLibrary or ColorManager)

Parameters

id	Class GUID

Returns

IEDLObjectPtr A smart pointer to the IEDLObject

registerClass()

```
virtual bool IEDLClassFactory::registerClass (
            const CClassID & id,
            creatorFunc f )  [pure virtual]
```

Register a GUID with creator function.

Parameters

id	Class GUID
f	Function that creates the class

Returns

bool True on success

registerNamedClass()

```
virtual bool IEDLClassFactory::registerNamedClass (
            const EDLSysString & strName,
            const CClassID & id,
            bool overwrite = true )  [pure virtual]
```

Register a GUID under a string name.

Parameters

strName	Name for registartion
id	Class GUID
overwrite	If overwrite is true then overwites old name registration else returns false

Returns

bool True on success

The documentation for this class was generated from the following file:

- iedlfactory.h

7.290 IEDLError Class Reference

An abstract class for EDL exceptions.

`#include <edlerrors.h>`

Inherits std::exception.

7.290.1 Detailed Description

An abstract class for EDL exceptions.

```
what() may return only a generic message. For more detailed error messages
add edl/edlerrors.cpp to your project, and use IEDLError::getErrorDescription(),
passing the the results of getEDLErrorString(). For example:

catch (const EDL::IEDLError& e)
{
      EDLString formatString = EDL::getEDLErrorString(e.getErrorCode());
      EDLString description = e.getErrorDescription(formatString);
      // deal with the description as seen fit
}
```

The documentation for this class was generated from the following file:

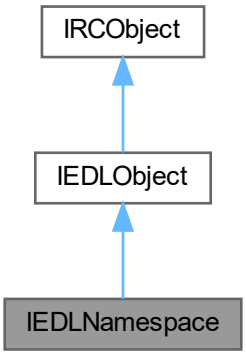
- edlerrors.h

7.291 IEDLNamespace Class Reference

Interface to EDL Namespace class.

`#include <edlqname.h>`

Inheritance diagram for IEDLNamespace:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual bool getPrefix (EDLSysString &sPrefix) const =0

  Retrieves the namespace prefix.
- virtual bool setPrefix (const EDLSysString &sPrefix)=0

  Sets the name space prefix.
- virtual bool getNamespace (EDLSysString &sNamespace) const =0

  Retrieves the namespace.
- virtual bool setNamespace (const EDLSysString &sNamespace)=0

  Sets the namespace.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves class id of IEDLNamespace.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.291.1 Detailed Description

Interface to EDL Namespace class.

7.291.2 Member Function Documentation

classID()

```
static const CClassID & IEDLNamespace::classID ( )  [inline], [static]
```

Retrieves class id of IEDLNamespace.

Returns

CClassID class id of the element

getNamespace()

```
virtual bool IEDLNamespace::getNamespace (
            EDLSysString & sNamespace ) const  [pure virtual]
```

Retrieves the namespace.

Parameters

sNamespace	Namespace

Returns

bool True on success

getPrefix()

```
virtual bool IEDLNamespace::getPrefix (
            EDLSysString & sPrefix ) const  [pure virtual]
```

Retrieves the namespace prefix.

Parameters

sPrefix	Namespace prefix

Returns

bool True on success

setNamespace()

```
virtual bool IEDLNamespace::setNamespace (
            const EDLSysString & sNamespace )  [pure virtual]
```

Sets the namespace.

Parameters

sNamespace	Namespace

Returns

bool Returns true on success

setPrefix()

```
virtual bool IEDLNamespace::setPrefix (
            const EDLSysString & sPrefix )  [pure virtual]
```

Sets the name space prefix.

Parameters

sPrefix	Namespace prefix

Returns

bool True on success

The documentation for this class was generated from the following file:

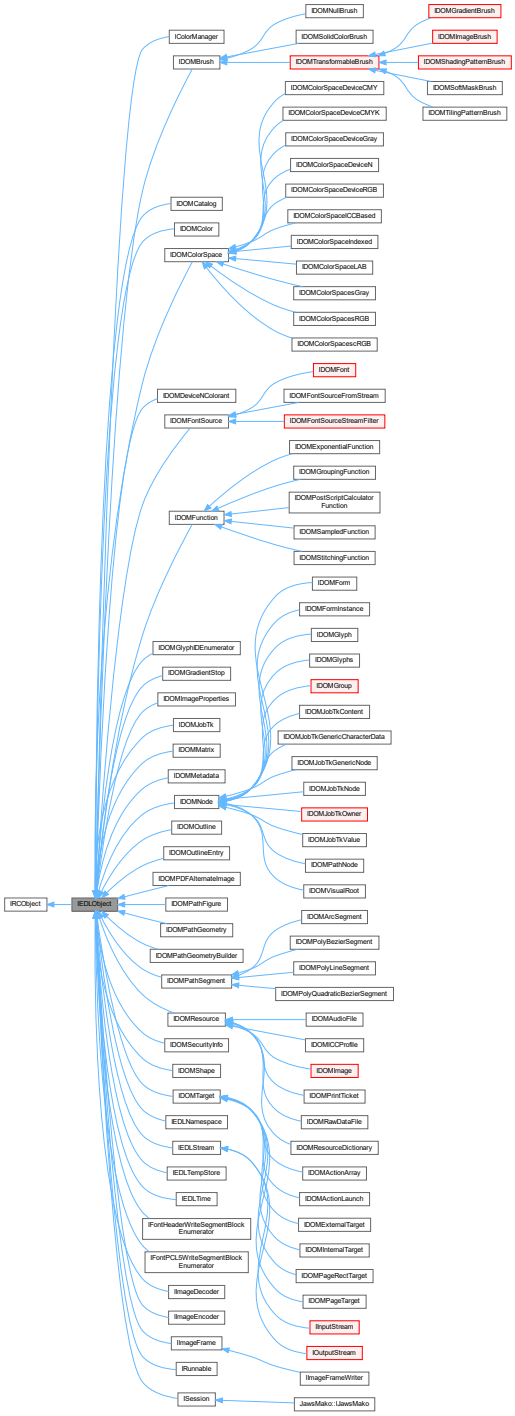
- edlqname.h

7.292 IEDLObject Class Reference

IEDLObject is an abstract base class that is used by all classes that are intended to be created via an EDL class factory.

`#include <iedlobject.h>`

Inheritance diagram for IEDLObject:



Public Member Functions

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.

- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.

- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.292.1 Detailed Description

IEDLObject is an abstract base class that is used by all classes that are intended to be created via an EDL class factory.

IEDLObjects are always internally reference-counted so that any user of an IEDLObject does not need to consider memory-management/object-lifetime/object-deletion issues

7.292.2 Member Function Documentation

clone()

```
virtual bool IEDLObject::clone (
            IEDLObjectPtr & ptrObject,
            IEDLClassFactory * pFactory )  [inline], [virtual]
```

Create a copy of EDLObject.

Parameters

ptrObject	Smart pointer to the source object
pFactory	Pointer to the EDL class factory

Returns

bool Returns true on success

Reimplemented in IEDLTempStore.

getClassID()

```
virtual const CClassID & IEDLObject::getClassID ( ) const  [pure virtual]
```

Returns class ID of IEDLObject.

Returns

CClassID Returns reference to class ID

init()

```
virtual bool IEDLObject::init (
            CClassParams * pData )  [inline], [virtual]
```

The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.

Parameters

pData	It is upto the caller/user of the class factory to construct any sub-class of CClassParams that must be supplied to this init() method

Returns

bool Returns true on success

The documentation for this class was generated from the following file:

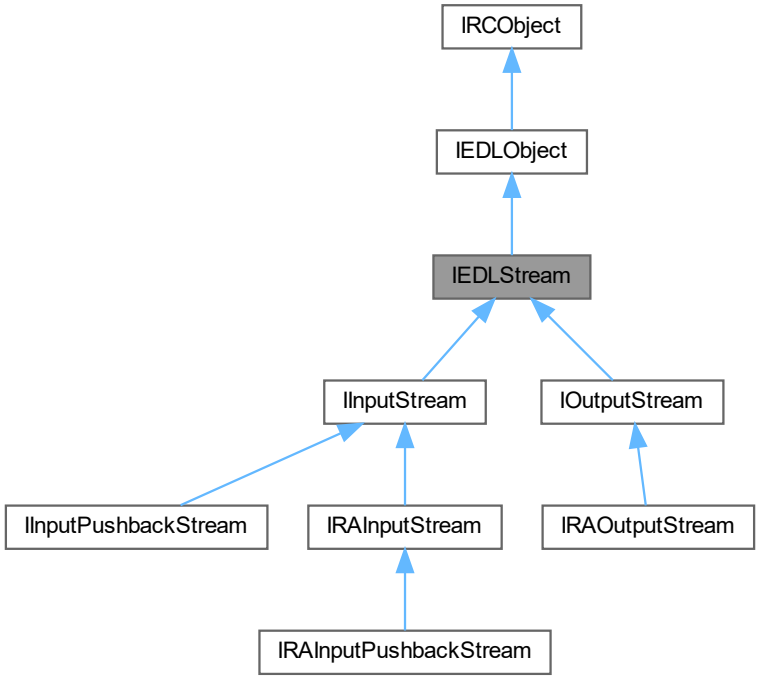
- iedlobject.h

7.293 IEDLStream Class Reference

Generic stream. Abstract base class for EDL stream subsystem.

```
#include <edlstream.h>
```

Inheritance diagram for IEDLStream:



Public Member Functions

- virtual bool isValid () const =0

  Determine stream validity.
- virtual bool open ()

  Opens the stream.
- virtual void **openE** ()=0

  As per open(), but will throw an exception on failure (IEDLError) that for some stream types may contain additional failure information.
- virtual void **close** ()=0

  Closes the stream.
- virtual int64 getPos ()=0

  Get current stream position.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.293.1 Detailed Description

Generic stream. Abstract base class for EDL stream subsystem.

7.293.2 Member Function Documentation

getPos()

```
virtual int64 IEDLStream::getPos ( )  [pure virtual]
```

Get current stream position.

Returns

int64 The current stream position.

isValid()

```
virtual bool IEDLStream::isValid ( ) const  [pure virtual]
```

Determine stream validity.

Returns

bool True if stream is valid for operations (no fatal problems).

open()

```
virtual bool IEDLStream::open ( )  [inline], [virtual]
```

Opens the stream.

Returns

bool True if stream was successfully opened.

The documentation for this class was generated from the following file:

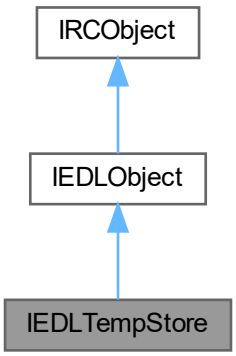
- edlstream.h

7.294 IEDLTempStore Class Reference

A self-cleaning area for storage of temporary data in the form of streams. One per session, obtainable from an ISession. Exceptions of type IEDLError are thrown on failures.

```
#include <iedltempstore.h>
```

Inheritance diagram for IEDLTempStore:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual IEDLTempStoreObjectPtr createTemporaryObject ()=0

  Create a temporary object, which is similar in concept to a file.
- virtual IEDLTempStoreObjectPtr createTemporaryObjectWithContents (const IInputStreamPtr &stream)=0

  Create a temporary object with the contents of an existing input stream.
- virtual IRAInputStreamPtr createTemporaryStreamWithContents (const IInputStreamPtr &stream)=0

  Copy the given stream into a temporary store object and return a buffered reader. Convenience.
- virtual void createTemporaryReaderWriterPair (IRAInputStreamPtr &reader, IRAOutputStreamPtr &writer)=0

  Create a temporary object and provide a reader/writer pair. Convenience method.
- virtual IRAInputOutputStreamPtr createTemporaryReaderWriter ()=0

  Create a temporary object and provide a consolidated reader/writer. Convenience method.
- virtual bool getWasExhausted () const =0

  Determine if the temporary store has been exhausted since the last time this flag was cleared. To clear this, use clearExhaustedFlag() below. Note that this means that this flag can be true even if there is currently available space in the temp store.
- virtual void **clearExhaustedFlag** ()=0

  Clear the exhausted flag. See getWasExhausted() above.
- virtual bool **clone** (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Clone - not available for objects of this type.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves class id of IEDLTempStore.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.294.1 Detailed Description

A self-cleaning area for storage of temporary data in the form of streams. One per session, obtainable from an ISession. Exceptions of type IEDLError are thrown on failures.

7.294.2 Member Function Documentation

classID()

```
static const CClassID & IEDLTempStore::classID ( )  [inline], [static]
```

Retrieves class id of IEDLTempStore.

Returns

CClassID class id of the object

createTemporaryObject()

```
virtual IEDLTempStoreObjectPtr IEDLTempStore::createTemporaryObject ( )  [pure virtual]
```

Create a temporary object, which is similar in concept to a file.

Returns

IEDLTempStoreObjectPtr The new temporary object.

createTemporaryObjectWithContents()

```
virtual IEDLTempStoreObjectPtr IEDLTempStore::createTemporaryObjectWithContents (
            const IInputStreamPtr & stream )  [pure virtual]
```

Create a temporary object with the contents of an existing input stream.

Parameters

stream	Reference to the source stream.

Returns

IEDLTempStoreObjectPtr The new temporary object

createTemporaryReaderWriter()

```
virtual IRAInputOutputStreamPtr IEDLTempStore::createTemporaryReaderWriter ( )  [pure virtual]
```

Create a temporary object and provide a consolidated reader/writer. Convenience method.

Returns

IRAInputOutputStreamPtr The consolidated reader/writer for the temporary object.

createTemporaryReaderWriterPair()

```
virtual void IEDLTempStore::createTemporaryReaderWriterPair (
            IRAInputStreamPtr & reader,
            IRAOutputStreamPtr & writer )  [pure virtual]
```

Create a temporary object and provide a reader/writer pair. Convenience method.

Parameters

reader	reference to receive a reader for the temporary object.
writer	reference to receive a writer for the temporary object.

createTemporaryStreamWithContents()

```
virtual IRAInputStreamPtr IEDLTempStore::createTemporaryStreamWithContents (
            const IInputStreamPtr & stream )  [pure virtual]
```

Copy the given stream into a temporary store object and return a buffered reader. Convenience.

Parameters

stream	Reference to the source stream.

Returns

IRAInputStreamPtr The result stream.

getWasExhausted()

```
virtual bool IEDLTempStore::getWasExhausted ( ) const  [pure virtual]
```

Determine if the temporary store has been exhausted since the last time this flag was cleared. To clear this, use clearExhaustedFlag() below. Note that this means that this flag can be true even if there is currently available space in the temp store.

Note that Mako will not currently clear this flag itself, so it may be used to detect that an exhaustion issue has occurred.

Returns

bool Returns true if the temp store was exhausted since the last time the flag was cleared, false otherwise.

The documentation for this class was generated from the following file:

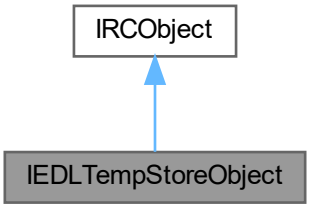
- iedltempstore.h

7.295 IEDLTempStoreObject Class Reference

A mechanism for storing and accessing temporary data for use with Jaws Mako.

`#include <iedltempstore.h>`

Inheritance diagram for IEDLTempStoreObject:



Public Member Functions

- virtual IRAOutputStreamPtr createWriter ()=0

  Create a writer that may be used to write to this temporary object. Multiple writers writing to a single temporary store object is not supported, and the writer may not be cloned.
- virtual IRAInputStreamPtr createReader (bool buffered=true)=0

  Create a reader that may be used to read from this temporary object Many readers can simultaneously perform reads.
- virtual IRAInputOutputStreamPtr **createReaderWriter** ()=0

  Create a stream that may be used to simulatneously read and write from this temporary object. Multiple writers writing to a single temporary object store is not supported. Clones of this object type will be read only. This is convenient for some cases that require simultaneous reading and writing. Reads from this stream will be unbuffered, so for best performance, please consider creating a reader writer pair instead.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.295.1 Detailed Description

A mechanism for storing and accessing temporary data for use with Jaws Mako.

A temporary, file-like object, stored with an IEDLTempStore. Exceptions of type IEDLError are thrown on failures.

7.295.2 Member Function Documentation

createReader()

```
virtual IRAInputStreamPtr IEDLTempStoreObject::createReader (
            bool buffered = true )  [pure virtual]
```

Create a reader that may be used to read from this temporary object Many readers can simultaneously perform reads.

Parameters

buffered	- If set to false, the resulting reader will not buffer read data. Useful if a reader and writer are being used on the same IEDLTempStoreObject, as it allows changes made by a writer to be immediately available to users of the reader.

Returns

IRAInputStreamPtr The reader.

createWriter()

```
virtual IRAOutputStreamPtr IEDLTempStoreObject::createWriter ( )  [pure virtual]
```

Create a writer that may be used to write to this temporary object. Multiple writers writing to a single temporary store object is not supported, and the writer may not be cloned.

Returns

IRAOutputStreamPtr The writer.

The documentation for this class was generated from the following file:

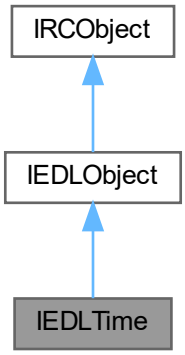
- iedltempstore.h

7.296 IEDLTime Class Reference

Interface to EDL date-time class.

`#include <edltime.h>`

Inheritance diagram for IEDLTime:



Classes

- class Data

  Initialization data.

Public Member Functions

- virtual bool setYear (uint16 year)=0

  Sets the year.
- virtual uint16 getYear () const =0

  Retrieves the year value.
- virtual bool setMonth (uint16 month)=0

  Sets the month.
- virtual uint16 getMonth () const =0

  Retrieves the month value.
- virtual bool setDay (uint16 day)=0

  Sets the day.
- virtual uint16 getDay () const =0

  Retrieves the day value.
- virtual bool setTime (uint16 hour=0, uint16 min=0, uint16 sec=0, uint16 ms=0, int16 tzd_sign=1, uint16 tzd↩_hour=0, uint16 tzd_min=0)=0

  Sets the time.
- virtual void getTime (uint16 &hour, uint16 &min, uint16 &sec, uint16 &ms, int16 &tzd_sign, uint16 &tzd_hour, uint16 &tzd_min) const =0

  Retrieves the time.

- virtual EDLSysString toW3CDTF () const =0

  Convert IEDLTime to string with W3CDTF format.
- virtual bool fromW3CDTF (const EDLSysString &strTime)=0

  Fill IEDLTime value from string with W3CDTF format.
- virtual EDLSysString toPDFDate () const =0

  Convert IEDLTime to string with PDF date format.
- virtual bool fromPDFDate (const EDLSysString &strTime)=0

  Fill IEDLTime value from string with PDF date format.
- virtual bool isEqualTo (const IEDLTimePtr &ptrTime) const =0

  Check this IEDLTime for equality to another IEDLTime.
- virtual int32 compare (const IEDLTimePtr &ptrTime) const =0

  Compare this time against ptrTime.
- virtual void **now** ()=0

  Sets date/time to the current system date and time.
- virtual void **toUTC** ()=0

  Convert UTC date/time.
- virtual void **toLocalTime** ()=0

  Convert to local date/time.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static EDL_API IEDLTimePtr **createNow** (IEDLClassFactory *pFactory)

  Simplified creator to create the time as of now.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.296.1 Detailed Description

Interface to EDL date-time class.

7.296.2 Member Function Documentation

compare()

```
virtual int32 IEDLTime::compare (
            const IEDLTimePtr & ptrTime ) const  [pure virtual]
```

Compare this time against ptrTime.

Parameters

ptrTime	The time to compare.

Returns

-1 if this time is earlier than ptrTime, 0 if this time is equal to ptrTime or 1 if this time is later than ptrTime

fromPDFDate()

```
virtual bool IEDLTime::fromPDFDate (
            const EDLSysString & strTime )  [pure virtual]
```

Fill IEDLTime value from string with PDF date format.

Parameters

strTime	The string

Returns

bool. Returns true on success.

fromW3CDTF()

```
virtual bool IEDLTime::fromW3CDTF (
            const EDLSysString & strTime )  [pure virtual]
```

Fill IEDLTime value from string with W3CDTF format.

Parameters

strTime	The string

getDay()

```
virtual uint16 IEDLTime::getDay ( ) const  [pure virtual]
```

Retrieves the day value.

Returns

bool Returns true on success, false if the call fails.

getMonth()

```
virtual uint16 IEDLTime::getMonth ( ) const  [pure virtual]
```

Retrieves the month value.

Returns

bool Returns true on success, false if the call fails.

getTime()

```
virtual void IEDLTime::getTime (
            uint16 & hour,
            uint16 & min,
            uint16 & sec,
            uint16 & ms,
            int16 & tzd_sign,
            uint16 & tzd_hour,
            uint16 & tzd_min ) const  [pure virtual]
```

Retrieves the time.

Parameters

hour	The hour component
min	The minute component
sec	The second component
ms	The fractional second component
tzd_sign	The timezone sign component
tzd_hour	The timezone hour component
tzd_min	The timezone minute component

getYear()

```
virtual uint16 IEDLTime::getYear ( ) const  [pure virtual]
```

Retrieves the year value.

Returns

bool Returns true on success, false if the call fails.

isEqualTo()

```
virtual bool IEDLTime::isEqualTo (
            const IEDLTimePtr & ptrTime ) const  [pure virtual]
```

Check this IEDLTime for equality to another IEDLTime.

Parameters

ptrTime	The time to compare.

Returns

bool. Returns true if the times are the same.

setDay()

```
virtual bool IEDLTime::setDay (
            uint16 day )  [pure virtual]
```

Sets the day.

Parameters

day	The day value

setMonth()

```
virtual bool IEDLTime::setMonth (
            uint16 month )  [pure virtual]
```

Sets the month.

Parameters

month	The month value

setTime()

```
virtual bool IEDLTime::setTime (
            uint16 hour = 0,
            uint16 min = 0,
```

```
            uint16 sec = 0,
            uint16 ms = 0,
            int16 tzd_sign = 1,
            uint16 tzd_hour = 0,
            uint16 tzd_min = 0 )  [pure virtual]
```

Sets the time.

Parameters

hour	The hour component
min	The minute component
sec	The second component
ms	The fractional second component
tzd_sign	The timezone sign component
tzd_hour	The timezone hour component
tzd_min	The timezone minute component

Returns

bool Returns true on success, false if the call fails.

setYear()

```
virtual bool IEDLTime::setYear (
            uint16 year )  [pure virtual]
```

Sets the year.

Parameters

year	The year value

toPDFDate()

```
virtual EDLSysString IEDLTime::toPDFDate ( ) const  [pure virtual]
```

Convert IEDLTime to string with PDF date format.

Returns

EDLSysString Returns IEDLTime value as string in W3CDTF format

toW3CDTF()

```
virtual EDLSysString IEDLTime::toW3CDTF ( ) const  [pure virtual]
```

Convert IEDLTime to string with W3CDTF format.

Returns

EDLSysString Returns IEDLTime value as string in W3CDTF format

The documentation for this class was generated from the following file:

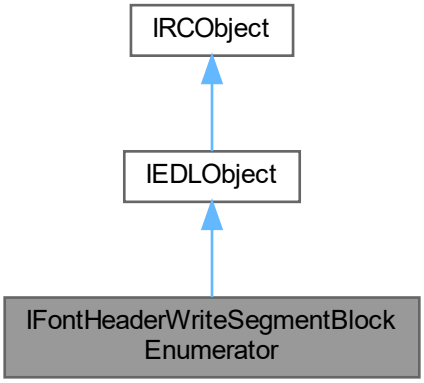
- edltime.h

7.297 IFontHeaderWriteSegmentBlockEnumerator Class Reference

IFontHeaderWriteSegmentBlockEnumerator Enumerates over the PCLXL Font Header block items for the XL ReadFontHeader operator.

`#include <idomfontpcl.h>`

Inheritance diagram for IFontHeaderWriteSegmentBlockEnumerator:



Public Types

- enum eFontHeaderWriteSegmentItem

  The enumeration of the segment types to be enumerated over.

Public Member Functions

- virtual eFontHeaderWriteSegmentItem getSegmentItem ()=0

  Returns the current segment item in the enumeration.
- virtual int32 getSegmentBlockItem ()=0

  Returns the current block item of segment item in the enumeration.
- virtual int32 getSegmentBlockSize ()=0

  Returns the current enumeration item segment block size.
- virtual int32 writeSegmentBlock (const IOutputStreamPtr &outStream, eEDLEndian endian)=0

  Writes the current enumeration item block.
- virtual bool haveMoreEnumerationItems ()=0

  Indicate whether there are more items in the enumeration.
- virtual bool nextEnumerationItem ()=0

  Get the next enumeration item.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves class id of IDOM.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.297.1 Detailed Description

IFontHeaderWriteSegmentBlockEnumerator Enumerates over the PCLXL Font Header block items for the XL ReadFontHeader operator.

7.297.2 Member Function Documentation

classID()

```
static const CClassID & IFontHeaderWriteSegmentBlockEnumerator::classID ( )  [inline], [static]
```

Retrieves class id of IDOM.

Returns

CClassID Class id of the element

getSegmentBlockItem()

```
virtual int32 IFontHeaderWriteSegmentBlockEnumerator::getSegmentBlockItem ( )  [pure virtual]
```

Returns the current block item of segment item in the enumeration.

Returns

int32 The segment block number

getSegmentBlockSize()

```
virtual int32 IFontHeaderWriteSegmentBlockEnumerator::getSegmentBlockSize ( )  [pure virtual]
```

Returns the current enumeration item segment block size.

Returns

int32 The segment block size

getSegmentItem()

```
virtual eFontHeaderWriteSegmentItem IFontHeaderWriteSegmentBlockEnumerator::getSegmentItem ( )
[pure virtual]
```

Returns the current segment item in the enumeration.

Returns

eFontHeaderWriteSegmentItem The segment item

haveMoreEnumerationItems()

```
virtual bool IFontHeaderWriteSegmentBlockEnumerator::haveMoreEnumerationItems ( )  [pure virtual]
```

Indicate whether there are more items in the enumeration.

Returns

bool True when there are more items in the enumeration.

nextEnumerationItem()

```
virtual bool IFontHeaderWriteSegmentBlockEnumerator::nextEnumerationItem ( )  [pure virtual]
```

Get the next enumeration item.

Returns

bool True on success

writeSegmentBlock()

```
virtual int32 IFontHeaderWriteSegmentBlockEnumerator::writeSegmentBlock (
            const IOutputStreamPtr & outStream,
            eEDLEndian endian )  [pure virtual]
```

Writes the current enumeration item block.

Parameters

outStream	The output stream
endian	The endian of the data to write out

Returns

int32 The size of data writen to the output stream

The documentation for this class was generated from the following file:

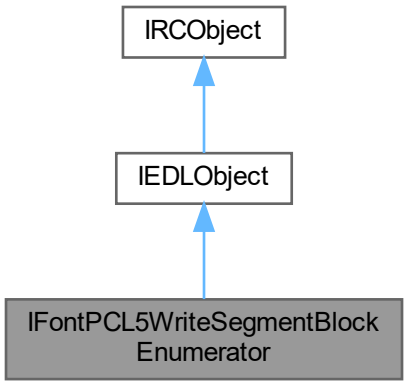
- idomfontpcl.h

7.298 IFontPCL5WriteSegmentBlockEnumerator Class Reference

IFontPCL5WriteSegmentBlockEnumerator Enumerates over the PCL5 font blocks.

`#include <idomfontpcl.h>`

Inheritance diagram for IFontPCL5WriteSegmentBlockEnumerator:



Public Member Functions

- virtual int32 getEnumerationCount ()=0

  Return the total number of enumeration blocks that will be iterated through.
- virtual int32 getEnumerationItemBlockSize ()=0

  Return the current enumeration item block size.
- virtual int32 writeEnumerationItemBlock (const IOutputStreamPtr &outStream, eEDLEndian endian)=0

  Writes the current enumeration item block.
- virtual bool haveMoreEnumerationItems ()=0

  Indicate whether there are more items in the enumeration.
- virtual bool nextEnumerationItem ()=0

  Get the next enumeration item.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves class id of IDOM.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.298.1 Detailed Description

IFontPCL5WriteSegmentBlockEnumerator Enumerates over the PCL5 font blocks.

7.298.2 Member Function Documentation

classID()

```
static const CClassID & IFontPCL5WriteSegmentBlockEnumerator::classID ( )  [inline], [static]
```

Retrieves class id of IDOM.

Returns

CClassID Class id of the element

getEnumerationCount()

```
virtual int32 IFontPCL5WriteSegmentBlockEnumerator::getEnumerationCount ( )  [pure virtual]
```

Return the total number of enumeration blocks that will be iterated through.

Returns

int32 The enumeration count

getEnumerationItemBlockSize()

```
virtual int32 IFontPCL5WriteSegmentBlockEnumerator::getEnumerationItemBlockSize ( )  [pure
virtual]
```

Return the current enumeration item block size.

Returns

int32 The segment block size

haveMoreEnumerationItems()

```
virtual bool IFontPCL5WriteSegmentBlockEnumerator::haveMoreEnumerationItems ( )  [pure virtual]
```

Indicate whether there are more items in the enumeration.

Returns

bool Whether there are more items in the enumeration.

nextEnumerationItem()

```
virtual bool IFontPCL5WriteSegmentBlockEnumerator::nextEnumerationItem ( )  [pure virtual]
```

Get the next enumeration item.

Returns

bool True on success

writeEnumerationItemBlock()

```
virtual int32 IFontPCL5WriteSegmentBlockEnumerator::writeEnumerationItemBlock (
            const IOutputStreamPtr & outStream,
            eEDLEndian endian )  [pure virtual]
```

Writes the current enumeration item block.

Parameters

outStream	The output stream
endian	The endian of the data to write out

Returns

int32 The size of data writen to the output stream

The documentation for this class was generated from the following file:

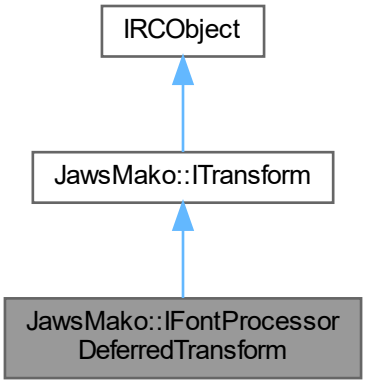
- idomfontpcl.h

7.299 JawsMako::IFontProcessorDeferredTransform Class Reference

A transform for performing font subsetting and merging, but it only replaces modified fonts with placeholders. These fonts will /not/ be usable until finaliseFonts() is called.

`#include <transforms.h>`

Inheritance diagram for JawsMako::IFontProcessorDeferredTransform:



Public Member Functions

- virtual void **finaliseFonts** ()=0

  Finalise any fonts, ensuring that any placeholders have now been populated with real font data.
- virtual void **setShouldSubset** (bool subset)=0

  Set whether or not subsetting should be performed. The default is true.
- virtual void **setShouldMerge** (bool merge)=0

  Set whether or not font merging should be performed. The default is true.
- virtual bool shouldMerge ()=0

  Determine if font merging has been enabled.
- virtual bool shouldSubset ()=0

  Determine if font subsetting has been enabled.

Public Member Functions inherited from JawsMako::ITransform

- virtual IDOMBrushPtr transform (const IDOMBrushPtr &brush, eBrushUsage usage=eBUGeneral, const CTransformState &state=CTransformState())=0

  Apply the transform to the given brush, if applicable. These transforms are thread safe.
- virtual IDOMImagePtr transform (const IDOMImagePtr &image, const CTransformState &state=CTransformState())=0

  Apply the transform to the given image, if applicable. These transforms are thread safe.
- virtual IDOMColorPtr transform (const IDOMColorPtr &color, const CTransformState &state=CTransformState())=0

  Apply the transform to the given color, if applicable. These transforms are thread safe.
- virtual IDOMColorSpacePtr transform (const IDOMColorSpacePtr &colorSpace, const CTransformState &state=CTransformState())=0

  Apply the transform to the given colorspace, if applicable. These transforms are thread safe.
- virtual IDOMNodePtr transform (const IDOMNodePtr &node, bool &changed, bool transformChildren=true, const CTransformState &state=CTransformState())=0

  Apply the transform to the given node, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time.
- virtual void transformPage (const IPagePtr &page, bool transformContent=true, bool transform↩ Annotations=true)=0

  Apply the transform to the given page, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time. The transform will also apply to the annotations appearances.
- virtual void **flushCaches** ()=0

  Flush the caches used by the transform. Most transforms cache recently transformed results to improve the performance of repeated transformations of equivalent results. However, it is possible that some cached results may point to entities that no longer exist, such as content inside an XPS file that no longer exists. If you are deleting or replacing files where transforms have been used, it is advisable to invoke this routine to clear the caches.
- virtual void setProgressMonitor (const IProgressMonitorPtr &progressMonitor)=0

  Set the IProgressMonitor object for this transform to allow for monitoring the progress of the transform.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IFontProcessorDeferredTransformPtr create (const IJawsMakoPtr &jawsMako, const IAbortPtr &abort=IAbortPtr())

  Create the transform.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.299.1 Detailed Description

A transform for performing font subsetting and merging, but it only replaces modified fonts with placeholders. These fonts will /not/ be usable until finaliseFonts() is called.

That is, the results of invoking transform() are not complete and should not be used for other purposes until finaliseFonts() is invoked. As such, it is not safe to use this transform in an ITransformChain with other transforms that may need to use the font information.

This transform exists so that many pages may be transformed before font changes are committed. This means that a single subset of a font may be generated for a batch of pages, or a single merged font for a batch of pages. This generates more efficient results than IFontProcessorTransform can provide. Please see the description of IFontProcessor for further details.

To use this transform effectively, apply it to as large a series of objects as possible, and then invoke finaliseFonts() to complete the operation and fully populate the merged or subsetted fonts.

Internally, the IFontProcessorTransform uses this transform internally, simply invoking finaliseFonts() before returning control to the caller.

7.299.2 Member Function Documentation

create()

```
static JAWSMAKO_API IFontProcessorDeferredTransformPtr JawsMako::IFontProcessorDeferred↩
Transform::create (
            const IJawsMakoPtr & jawsMako,
            const IAbortPtr & abort = IAbortPtr() )  [static]
```

Create the transform.

Parameters

jawsMako	The JawsMako instance.
abort	An abort callback handler.

Returns

The new instance.

shouldMerge()

```
virtual bool JawsMako::IFontProcessorDeferredTransform::shouldMerge ( )  [pure virtual]
```

Determine if font merging has been enabled.

Returns

bool Whether or not font merging has been enabled.

shouldSubset()

```
virtual bool JawsMako::IFontProcessorDeferredTransform::shouldSubset ( )  [pure virtual]
```

Determine if font subsetting has been enabled.

Returns

bool Whether or not font subsetting has been enabled.

The documentation for this class was generated from the following file:

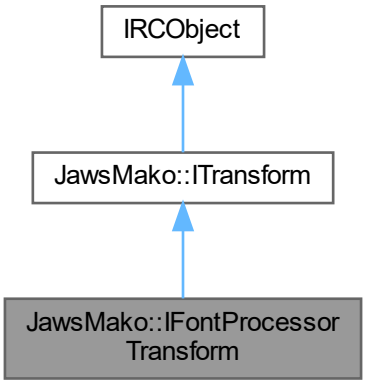
- transforms.h

7.300 JawsMako::IFontProcessorTransform Class Reference

A transform for performing font subsetting and merging.

```
#include <transforms.h>
```

Inheritance diagram for JawsMako::IFontProcessorTransform:



Public Member Functions

- virtual void **setShouldSubset** (bool subset)=0

  Set whether or not subsetting should be performed. The default is true.
- virtual void **setShouldMerge** (bool merge)=0

  Set whether or not font merging should be performed. The default is true.
- virtual bool shouldMerge ()=0

  Determine if font merging has been enabled.
- virtual bool shouldSubset ()=0

  Determine if font subsetting has been enabled.

Public Member Functions inherited from JawsMako::ITransform

- virtual IDOMBrushPtr transform (const IDOMBrushPtr &brush, eBrushUsage usage=eBUGeneral, const CTransformState &state=CTransformState())=0

  Apply the transform to the given brush, if applicable. These transforms are thread safe.
- virtual IDOMImagePtr transform (const IDOMImagePtr &image, const CTransformState &state=CTransformState())=0

  Apply the transform to the given image, if applicable. These transforms are thread safe.
- virtual IDOMColorPtr transform (const IDOMColorPtr &color, const CTransformState &state=CTransformState())=0

  Apply the transform to the given color, if applicable. These transforms are thread safe.
- virtual IDOMColorSpacePtr transform (const IDOMColorSpacePtr &colorSpace, const CTransformState &state=CTransformState())=0

  Apply the transform to the given colorspace, if applicable. These transforms are thread safe.
- virtual IDOMNodePtr transform (const IDOMNodePtr &node, bool &changed, bool transformChildren=true, const CTransformState &state=CTransformState())=0

  Apply the transform to the given node, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time.
- virtual void transformPage (const IPagePtr &page, bool transformContent=true, bool transform↩ Annotations=true)=0

  Apply the transform to the given page, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time. The transform will also apply to the annotations appearances.
- virtual void **flushCaches** ()=0

  Flush the caches used by the transform. Most transforms cache recently transformed results to improve the performance of repeated transformations of equivalent results. However, it is possible that some cached results may point to entities that no longer exist, such as content inside an XPS file that no longer exists. If you are deleting or replacing files where transforms have been used, it is advisable to invoke this routine to clear the caches.
- virtual void setProgressMonitor (const IProgressMonitorPtr &progressMonitor)=0

  Set the IProgressMonitor object for this transform to allow for monitoring the progress of the transform.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IFontProcessorTransformPtr create (const IJawsMakoPtr &jawsMako, const IAbortPtr &abort=IAbortPtr())

  Create the transform.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.300.1 Detailed Description

A transform for performing font subsetting and merging.

```
When merging fonts, this transform will see if fonts inside the object
being transformed can be merged with fonts that the transform has
previously seen. If so, the font will be merged. If a previously seen
font provides all the required glyphs, that previous font will be used.
However, if no previous font provides all the required glyphs, a new
font will be created.

When subsetting fonts, the transform will attempt to use previously
seen subset fonts that support the required glyphs. If not, a new subset
font will be created.


However both these operations should be performed over as large a set
of input as possible as these operations lose their effectiveness when
operating on, for example, single pages.

For example, if in a five page job, five potentially mergable fonts are
used, one per page. If each font adds glyphs that were not present, then
this transform will do a poor job. When run for each page we will be able
to merge each font, but will then need to generate a new font each time
due to the additional glyphs, resulting in essentially the same result
as the input.

For such situations, consider using the IFontProcessorDeferredTransform
transform, which would allow merging to be performed over a set of pages.
For the example above, it would be able to generate a single font as it
is able to defer creation of a merged font until after it has identified
the fonts that can be merged.
```

7.300.2 Member Function Documentation

create()

```
static JAWSMAKO_API IFontProcessorTransformPtr JawsMako::IFontProcessorTransform::create (
            const IJawsMakoPtr & jawsMako,
            const IAbortPtr & abort = IAbortPtr() )  [static]
```

Create the transform.

Parameters

jawsMako	The JawsMako instance.
abort	An abort callback handler.

Returns

The new instance.

shouldMerge()

```
virtual bool JawsMako::IFontProcessorTransform::shouldMerge ( )  [pure virtual]
```

Determine if font merging has been enabled.

Returns

bool Whether or not font merging has been enabled.

shouldSubset()

```
virtual bool JawsMako::IFontProcessorTransform::shouldSubset ( )  [pure virtual]
```

Determine if font subsetting has been enabled.

Returns

bool Whether or not font subsetting has been enabled.

The documentation for this class was generated from the following file:

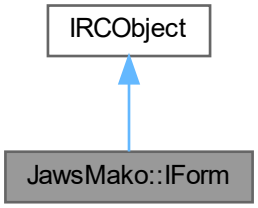
- transforms.h

7.301 JawsMako::IForm Class Reference

An interface class for an interactive form, which tracks a tree of IFormFields and widgets.

```
#include <interactive.h>
```

Inheritance diagram for JawsMako::IForm:



Public Member Functions

- virtual IFormPtr clone () const =0

  Clone the form field, which includes a deep clone of any child IFormFields.
- virtual CFormFieldVect getFields () const =0

  Get the top level fields in a vector.
- virtual CAnnotationReferenceVect getWidgets () const =0

  Get references to the top level widgets in a vector.
- virtual void **clearFields** ()=0

  Clear the top level fields list.

- virtual void addField (const IFormFieldPtr &field)=0

  Add a field at the top level. An exception will be thrown if the field to be inserted has the same partial name or id as an existing field at the top level.
- virtual void removeField (const IFormFieldPtr &field)=0

  Remove a top level field.
- virtual void **clearWidgets** ()=0

  Clear the top level widgets list.
- virtual void addWidget (const IWidgetAnnotationPtr &widget)=0

  Add a top level widget. An exception will be thrown if an annotation with the same reference is already.
- virtual void addWidget (const IAnnotationReferencePtr &widgetReference)=0

  Add a top level widget by reference. An exception will be thrown if an annotation with the same reference is already present at this level.
- virtual void removeWidget (const IWidgetAnnotationPtr &widget)=0

  Remove the given top level widget.
- virtual void removeWidget (const IAnnotationReferencePtr &widgetReference)=0

  Remove the given top level widget, by reference.
- virtual bool fieldInTree (const IFormFieldPtr &field) const =0

  Search the field tree for a field that has the same id as the given field. The search recurses through the children.
- virtual CFormFieldVect getPathToField (const IFormFieldPtr &field) const =0

  Find the path to a field with the same id that is present in the field tree.
- virtual bool widgetInTree (const IWidgetAnnotationPtr &widget) const =0

  Search the field sub for a widget. The search recurses through the child fields.
- virtual bool widgetInTree (const IAnnotationReferencePtr &widgetReference) const =0

  Search the field sub for a widget, by reference The search recurses through the child fields.
- virtual CFormFieldVect getPathToWidget (const IWidgetAnnotationPtr &widget) const =0

  Find the path to a widget with the same reference as the given widget.
- virtual CFormFieldVect getPathToWidget (const IAnnotationReferencePtr &widgetReference) const =0

  Find the path to a widget with the given reference.
- virtual eFieldType getWidgetFieldType (const IWidgetAnnotationPtr &widget) const =0

  Determine the type for the given widget field, resolving inherited values. Convenience. An exception is thrown if the widget is not present in the form.
- virtual uint32 getWidgetFieldFlags (const IWidgetAnnotationPtr &widget) const =0

  Determine the flags for the given widget field, resolving inherited values. Convenience. An exception is thrown if the widget is not present in the form.
- virtual U8String getWidgetDefaultAppearanceString (const IWidgetAnnotationPtr &widget) const =0

  Determine the default appearance string for the given widget field, resolving inherited values. Convenience. An exception is thrown if the widget is not present in the form.
- virtual CU8StringVect getWidgetValue (const IWidgetAnnotationPtr &widget) const =0

  Determine the value for the widget field, resolving inherited values. Convenience. An exception is thrown if the widget is not present in the form.
- virtual CU8StringVect getWidgetDefaultValue (const IWidgetAnnotationPtr &widget) const =0

  Determine the default value for the widget field, resolving inherited values. Convenience. An exception is thrown if the widget is not present in the form.
- virtual CU8StringVect getFieldValue (const IFormFieldPtr &field) const =0

  Determine the value for the field, resolving inherited values. Convenience. An exception is thrown if the widget is not present in the form.
- virtual CU8StringVect getFieldDefaultValue (const IFormFieldPtr &field) const =0

  Determine the default value for field, resolving inherited values. Convenience. An exception is thrown if the widget is not present in the form.
- virtual CFieldOptionVect getWidgetOptions (const IWidgetAnnotationPtr &widget) const =0

  Determine the options value for the widget field, resolving inherited values. Convenience. An exception is thrown if the widget is not present in the form.
- virtual eQuadding getWidgetQuadding (const IWidgetAnnotationPtr &widget) const =0

Determine the quadding for the given widget field, resolving inherited values. Convenience. An exception is thrown if the widget is not present in the form.

- virtual bool getNeedAppearances () const =0

  Get whether a viewer should construct its own appearances for the widgets in this form.
- virtual void setNeedAppearances (bool needAppearances)=0

  Set whether a viewer should construct its own appearances for the widgets in this field.
- virtual CXFAPacketVect getXfaPacketData () const =0

  Get the XFA packet data, if present, for this form. This information is not currently used by Mako, but is preserved.
- virtual void setXfaPacketData (const CXFAPacketVect &xfaPacketData)=0

  Set the XFA packet data for this form.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IFormPtr create (const IJawsMakoPtr &jawsMako)

  Create a new form.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.301.1 Detailed Description

An interface class for an interactive form, which tracks a tree of IFormFields and widgets.

7.301.2 Member Function Documentation

addField()

```
virtual void JawsMako::IForm::addField (
            const IFormFieldPtr & field )  [pure virtual]
```

Add a field at the top level. An exception will be thrown if the field to be inserted has the same partial name or id as an existing field at the top level.

Parameters

field	Field too be added

addWidget() [1/2]

```
virtual void JawsMako::IForm::addWidget (
            const IAnnotationReferencePtr & widgetReference )  [pure virtual]
```

Add a top level widget by reference. An exception will be thrown if an annotation with the same reference is already present at this level.

Parameters

widgetReference	The widget, by reference, to be added

addWidget() [2/2]

```
virtual void JawsMako::IForm::addWidget (
            const IWidgetAnnotationPtr & widget )  [pure virtual]
```

Add a top level widget. An exception will be thrown if an annotation with the same reference is already.

Parameters

widget	The widget to be added

clone()

```
virtual IFormPtr JawsMako::IForm::clone ( ) const  [pure virtual]
```

Clone the form field, which includes a deep clone of any child IFormFields.

Returns

IFormPtr The cloned form field

create()

```
static JAWSMAKO_API IFormPtr JawsMako::IForm::create (
            const IJawsMakoPtr & jawsMako )  [static]
```

Create a new form.

Parameters

jawsMako	The Mako instance

Returns

IFormPtr The new form

fieldInTree()

```
virtual bool JawsMako::IForm::fieldInTree (
            const IFormFieldPtr & field ) const  [pure virtual]
```

Search the field tree for a field that has the same id as the given field. The search recurses through the children.

Parameters

field	The field to be matched

Returns

bool True if field was found, otherwise false

getFieldDefaultValue()

```
virtual CU8StringVect JawsMako::IForm::getFieldDefaultValue (
            const IFormFieldPtr & field ) const  [pure virtual]
```

Determine the default value for field, resolving inherited values. Convenience. An exception is thrown if the widget is not present in the form.

Parameters

field	The field to inspect

Returns

CU8StringVect The value(s)

getFields()

```
virtual CFormFieldVect JawsMako::IForm::getFields ( ) const  [pure virtual]
```

Get the top level fields in a vector.

Returns

CFormFieldVect The vector of fields

getFieldValue()

```
virtual CU8StringVect JawsMako::IForm::getFieldValue (
            const IFormFieldPtr & field ) const  [pure virtual]
```

Determine the value for the field, resolving inherited values. Convenience. An exception is thrown if the widget is not present in the form.

Parameters

field	The field to inspect

Returns

CU8StringVect The value(s)

getNeedAppearances()

```
virtual bool JawsMako::IForm::getNeedAppearances ( ) const  [pure virtual]
```

Get whether a viewer should construct its own appearances for the widgets in this form.

Returns

bool True if viewer should construct its own appearances

getPathToField()

```
virtual CFormFieldVect JawsMako::IForm::getPathToField (
            const IFormFieldPtr & field ) const  [pure virtual]
```

Find the path to a field with the same id that is present in the field tree.

This returns a vector of IFormFields representing the branches of the field tree that lead to the given field. That is, the last entry in the returned vector is the immediate parent of the desired field, the second last entry is the grandparent of the desired field, and so forth leading back to the top level.

This is useful in particular for determining the fully qualified name of a field.

An empty vector is returned if the requested field is at the top level of the form and therefore has no parents.

An exception is thrown if the field is not present in the tree.

Parameters

field	The field to be matched

Returns

CFormFieldVect A vector of IFormFields

getPathToWidget() [1/2]

```
virtual CFormFieldVect JawsMako::IForm::getPathToWidget (
            const IAnnotationReferencePtr & widgetReference ) const  [pure virtual]
```

Find the path to a widget with the given reference.

This returns a vector of IFormFields representing the branches of the field tree that lead to the given widget. That is, the last entry in the returned vector is the immediate parent of the desired widget, the second last entry is the grandparent of the desired widget, and so forth leading back to the top level.

This is useful in particular for determining the fully qualified name of a field.

An empty vector is returned if the requested widget is at the top level of the form and therefore has no parents.

An exception is thrown if the widget is not present in the tree.

Parameters

widgetReference	The widget, by reference, to be matched

Returns

CFormFieldVect A vector of IFormFields

getPathToWidget() [2/2]

```
virtual CFormFieldVect JawsMako::IForm::getPathToWidget (
            const IWidgetAnnotationPtr & widget ) const  [pure virtual]
```

Find the path to a widget with the same reference as the given widget.

This returns a vector of IFormFields representing the branches of the field tree that lead to the given widget. That is, the last entry in the returned vector is the immediate parent of the desired widget, the second last entry is the grandparent of the desired widget, and so forth leading back to the top level.

This is useful in particular for determining the fully qualified name of a field.

An empty vector is returned if the requested widget is at the top level of the form and therefore has no parents.

An exception is thrown if the widget is not present in the tree.

Parameters

widget	The widget to be matched

Returns

CFormFieldVect A vector of IFormFields

getWidgetDefaultAppearanceString()

```
virtual U8String JawsMako::IForm::getWidgetDefaultAppearanceString (
            const IWidgetAnnotationPtr & widget ) const  [pure virtual]
```

Determine the default appearance string for the given widget field, resolving inherited values. Convenience. An exception is thrown if the widget is not present in the form.

Parameters

widget	The widget to inspect

Returns

U8String The default apperance string

getWidgetDefaultValue()

```
virtual CU8StringVect JawsMako::IForm::getWidgetDefaultValue (
            const IWidgetAnnotationPtr & widget ) const  [pure virtual]
```

Determine the default value for the widget field, resolving inherited values. Convenience. An exception is thrown if the widget is not present in the form.

Parameters

widget	The widget to inspect

Returns

CU8StringVect The value(s)

getWidgetFieldFlags()

```
virtual uint32 JawsMako::IForm::getWidgetFieldFlags (
            const IWidgetAnnotationPtr & widget ) const  [pure virtual]
```

Determine the flags for the given widget field, resolving inherited values. Convenience. An exception is thrown if the widget is not present in the form.

Parameters

widget	The widget to inspect

Returns

uint32 The field flags

getWidgetFieldType()

```
virtual eFieldType JawsMako::IForm::getWidgetFieldType (
            const IWidgetAnnotationPtr & widget ) const  [pure virtual]
```

Determine the type for the given widget field, resolving inherited values. Convenience. An exception is thrown if the widget is not present in the form.

Parameters

widget	The widget to inspect

Returns

fieldType The field type

getWidgetOptions()

```
virtual CFieldOptionVect JawsMako::IForm::getWidgetOptions (
            const IWidgetAnnotationPtr & widget ) const  [pure virtual]
```

Determine the options value for the widget field, resolving inherited values. Convenience. An exception is thrown if the widget is not present in the form.

Parameters

widget	The widget to inspect

Returns

CFormFieldVect The options

getWidgetQuadding()

```
virtual eQuadding JawsMako::IForm::getWidgetQuadding (
            const IWidgetAnnotationPtr & widget ) const  [pure virtual]
```

Determine the quadding for the given widget field, resolving inherited values. Convenience. An exception is thrown if the widget is not present in the form.

Parameters

widget	The widget to inspect

Returns

eQuadding The quadding

getWidgets()

```
virtual CAnnotationReferenceVect JawsMako::IForm::getWidgets ( ) const  [pure virtual]
```

Get references to the top level widgets in a vector.

Returns

CAnnotationReferenceVect The vector of annotation references

getWidgetValue()

```
virtual CU8StringVect JawsMako::IForm::getWidgetValue (
            const IWidgetAnnotationPtr & widget ) const  [pure virtual]
```

Determine the value for the widget field, resolving inherited values. Convenience. An exception is thrown if the widget is not present in the form.

Parameters

widget	The widget to inspect

Returns

CU8StringVect The value(s)

getXfaPacketData()

```
virtual CXFAPacketVect JawsMako::IForm::getXfaPacketData ( ) const  [pure virtual]
```

Get the XFA packet data, if present, for this form. This information is not currently used by Mako, but is preserved.

Returns

CXFAPacketVect The XFA packet data, or an empty vector if not present.

removeField()

```
virtual void JawsMako::IForm::removeField (
            const IFormFieldPtr & field )  [pure virtual]
```

Remove a top level field.

Parameters

field	Field to be removed

removeWidget() [1/2]

```
virtual void JawsMako::IForm::removeWidget (
            const IAnnotationReferencePtr & widgetReference )  [pure virtual]
```

Remove the given top level widget, by reference.

Parameters

widgetReference	The widget, by reference, to be removed

removeWidget() [2/2]

```
virtual void JawsMako::IForm::removeWidget (
            const IWidgetAnnotationPtr & widget )  [pure virtual]
```

Remove the given top level widget.

Parameters

widget	The widget to be removed

setNeedAppearances()

```
virtual void JawsMako::IForm::setNeedAppearances (
            bool needAppearances )  [pure virtual]
```

Set whether a viewer should construct its own appearances for the widgets in this field.

Parameters

needAppearances	True or false

setXfaPacketData()

```
virtual void JawsMako::IForm::setXfaPacketData (
            const CXFAPacketVect & xfaPacketData )  [pure virtual]
```

Set the XFA packet data for this form.

Parameters

xfaPacketData	The packet data to use, or an empty vector to clear.

widgetInTree() [1/2]

```
virtual bool JawsMako::IForm::widgetInTree (
            const IAnnotationReferencePtr & widgetReference ) const  [pure virtual]
```

Search the field sub for a widget, by reference The search recurses through the child fields.

Parameters

widgetReference	The widget, by reference, to be matched

Returns

bool True if widget was found, otherwise false

widgetInTree() [2/2]

```
virtual bool JawsMako::IForm::widgetInTree (
            const IWidgetAnnotationPtr & widget ) const  [pure virtual]
```

Search the field sub for a widget. The search recurses through the child fields.

Parameters

widget	The widget to be matched

Returns

bool True if widget was found, otherwise false

The documentation for this class was generated from the following file:

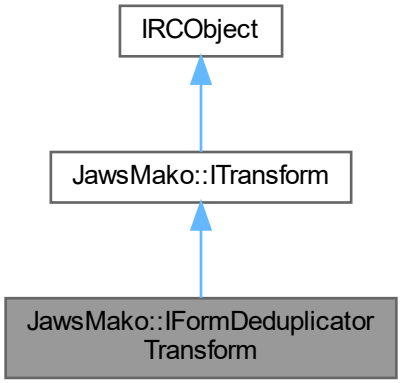
- interactive.h

7.302 JawsMako::IFormDeduplicatorTransform Class Reference

A transform that attempts to deduplicate graphically identical IDOMForm objects. Note that non-graphical properties, such as additional dictionary entries, structure information, etc, are not consulted when looking for identical forms.

```
#include <transforms.h>
```

Inheritance diagram for JawsMako::IFormDeduplicatorTransform:



Additional Inherited Members

Public Member Functions inherited from JawsMako::ITransform

- virtual IDOMBrushPtr transform (const IDOMBrushPtr &brush, eBrushUsage usage=eBUGeneral, const CTransformState &state=CTransformState())=0

  Apply the transform to the given brush, if applicable. These transforms are thread safe.
- virtual IDOMImagePtr transform (const IDOMImagePtr &image, const CTransformState &state=CTransformState())=0

  Apply the transform to the given image, if applicable. These transforms are thread safe.
- virtual IDOMColorPtr transform (const IDOMColorPtr &color, const CTransformState &state=CTransformState())=0

  Apply the transform to the given color, if applicable. These transforms are thread safe.
- virtual IDOMColorSpacePtr transform (const IDOMColorSpacePtr &colorSpace, const CTransformState &state=CTransformState())=0

  Apply the transform to the given colorspace, if applicable. These transforms are thread safe.
- virtual IDOMNodePtr transform (const IDOMNodePtr &node, bool &changed, bool transformChildren=true, const CTransformState &state=CTransformState())=0

  Apply the transform to the given node, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time.
- virtual void transformPage (const IPagePtr &page, bool transformContent=true, bool transform↩ Annotations=true)=0

  Apply the transform to the given page, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time. The transform will also apply to the annotations appearances.
- virtual void **flushCaches** ()=0

  Flush the caches used by the transform. Most transforms cache recently transformed results to improve the performance of repeated transformations of equivalent results. However, it is possible that some cached results may point to entities that no longer exist, such as content inside an XPS file that no longer exists. If you are deleting or replacing files where transforms have been used, it is advisable to invoke this routine to clear the caches.
- virtual void setProgressMonitor (const IProgressMonitorPtr &progressMonitor)=0

  Set the IProgressMonitor object for this transform to allow for monitoring the progress of the transform.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.302.1 Detailed Description

A transform that attempts to deduplicate graphically identical IDOMForm objects. Note that non-graphical properties, such as additional dictionary entries, structure information, etc, are not consulted when looking for identical forms.

The documentation for this class was generated from the following file:

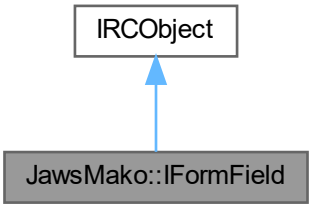
- transforms.h

7.303 JawsMako::IFormField Class Reference

An interface class for a form field. A form field may have multiple child fields and widget annotations, arranged in a tree.

`#include <interactive.h>`

Inheritance diagram for JawsMako::IFormField:



Public Member Functions

- virtual IFormFieldPtr clone () const =0

  Clone the form field, which includes a deep clone of any child IFormFields.
- virtual eFieldType getFieldType () const =0

  Get the type of the field. If eFTInherited is returned, the parent field should be consulted to determine the type.
- virtual bool getFieldFlags (uint32 &flags) const =0

  Get the field flags, if present.
- virtual void setFieldFlags (uint32 flags)=0

  Set the field flags - #see eFieldFlags.
- virtual CFormFieldVect getChildFields () const =0

  Get the child fields in a vector.
- virtual CAnnotationReferenceVect getChildWidgets () const =0

  Get references to the child widgets in a vector.
- virtual DOMid getFieldId () const =0

  Get the unique id for this field. This id is preserved across cloning, and therefore all clones of this IFormField will have the same id. This id is particularly useful when merging fields from one form to another.
- virtual void **clearChildFields** ()=0

  Clear the child fields list.
- virtual void addChildField (const IFormFieldPtr &field)=0

  Add a child field. An exception will be thrown if the field to be inserted has the same partial name or id as an existing field.
- virtual void removeChildField (const IFormFieldPtr &field)=0

  Remove a child field.
- virtual bool fieldInSubtree (const IFormFieldPtr &field) const =0

  Search the IFormField subtree for a field that has the same id as the given field. The search recurses through the children.
- virtual void **clearChildWidgets** ()=0

  Clear the child widgets list.
- virtual void addChildWidget (const IWidgetAnnotationPtr &widget)=0

  Add a child widget. An exception will be thrown if an annotation with the same reference is already present at this level.
- virtual void addChildWidget (const IAnnotationReferencePtr &widgetReference)=0

  Add a child widget by reference. An exception will be thrown if an annotation with the same reference is already present at this level.
- virtual void removeChildWidget (const IWidgetAnnotationPtr &widget)=0

  Remove the given child widget from the field.
- virtual void removeChildWidget (const IAnnotationReferencePtr &widgetReference)=0

  Remove the given child widget from the field, by reference.
- virtual bool widgetInSubtree (const IWidgetAnnotationPtr &widget) const =0

  Search the IFormField subtree for a widget The search recurses through the children.
- virtual bool widgetInSubtree (const IAnnotationReferencePtr &widgetReference) const =0

  Search the IFormField subtree for a widget, by reference The search recurses through the children.
- virtual U8String getPartialName () const =0

  Get the partial name of the field, if defined. If not present, an empty string will be returned.
- virtual void setPartialName (const U8String &name)=0

  Set the partial name of the field.
- virtual bool getValue (CU8StringVect &value) const =0

  Get the value of the field, as an array of strings.
- virtual void setValue (const CU8StringVect &value)=0

  Set the value of the field, as an array of strings. For fields that can take multiple values, eg radio buttons, there will be multiple entries. For fields requiring only a single value, a single entry will be returned. If the field has no value an empty array will be returned.

- virtual void setValue (const U8String &value)

  Convenience form for fields that only need a single value.
- virtual bool getDefaultValue (CU8StringVect &value) const =0

  Get the default value of the field, as an array of strings.
- virtual void setDefaultValue (const CU8StringVect &value)=0

  Set the value of the field, as an array of strings.
- virtual void setDefaultValue (const U8String &value)

  Convenience form for fields that only need a single value.
- virtual bool getDefaultAppearanceString (U8String &defaultAppearanceString) const =0

  Get the default appearance string for variable text, if set.
- virtual void setDefaultAppearanceString (const U8String &defaultAppearanceString)=0

  Set the default appearance string for variable text.
- virtual eQuadding getQuadding () const =0

  Get the Quadding (Justification) for variable text.
- virtual void setQuadding (eQuadding quadding)=0

  Set the Quadding (Justification) for variable text.
- virtual bool getOptions (CFieldOptionVect &options) const =0

  Get the options for this field.
- virtual void setOptions (const CFieldOptionVect &options)=0

  Set the options for this field.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IFormFieldPtr create (const IJawsMakoPtr &jawsMako, eFieldType type=eFTInherited, const U8String &partialName=U8String(), uint32 flags=0, const CFormFieldVect &childFields=CFormFieldVect(), const CAnnotationReferenceVect &childWidgets=CAnnotationReferenceVect())

  Create a new form field.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.303.1 Detailed Description

An interface class for a form field. A form field may have multiple child fields and widget annotations, arranged in a tree.

7.303.2 Member Function Documentation

addChildField()

```
virtual void JawsMako::IFormField::addChildField (
            const IFormFieldPtr & field )  [pure virtual]
```

Add a child field. An exception will be thrown if the field to be inserted has the same partial name or id as an existing field.

Parameters

field	Field to be added

addChildWidget() [1/2]

```
virtual void JawsMako::IFormField::addChildWidget (
            const IAnnotationReferencePtr & widgetReference )  [pure virtual]
```

Add a child widget by reference. An exception will be thrown if an annotation with the same reference is already present at this level.

Parameters

widgetReference	The widget, by reference, to be added

addChildWidget() [2/2]

```
virtual void JawsMako::IFormField::addChildWidget (
            const IWidgetAnnotationPtr & widget )  [pure virtual]
```

Add a child widget. An exception will be thrown if an annotation with the same reference is already present at this level.

Parameters

widget	The widget to be added

clone()

```
virtual IFormFieldPtr JawsMako::IFormField::clone ( ) const  [pure virtual]
```

Clone the form field, which includes a deep clone of any child IFormFields.

Returns

IFormFieldPtr The cloned field

create()

```
static JAWSMAKO_API IFormFieldPtr JawsMako::IFormField::create (
            const IJawsMakoPtr & jawsMako,
            eFieldType type = eFTInherited,
            const U8String & partialName = U8String(),
            uint32 flags = 0,
            const CFormFieldVect & childFields = CFormFieldVect(),
            const CAnnotationReferenceVect & childWidgets = CAnnotationReferenceVect() )
[static]
```

Create a new form field.

Parameters

jawsMako	The Mako instance
type	The type of field to create
partialName	The desired partial name of the field
flags	The field flags to set (see eFieldFlags)
childFields	A vector of child fields for this field
childWidgets	A vector of child widgets for this field

Returns

IFormFieldPtr The new form field

fieldInSubtree()

```
virtual bool JawsMako::IFormField::fieldInSubtree (
            const IFormFieldPtr & field ) const  [pure virtual]
```

Search the IFormField subtree for a field that has the same id as the given field. The search recurses through the children.

Parameters

field	Field to search for

Returns

bool True if found, otherwise false

getChildFields()

```
virtual CFormFieldVect JawsMako::IFormField::getChildFields ( ) const  [pure virtual]
```

Get the child fields in a vector.

Returns

CFormFieldVect The vector of child fields

getChildWidgets()

```
virtual CAnnotationReferenceVect JawsMako::IFormField::getChildWidgets ( ) const  [pure virtual]
```

Get references to the child widgets in a vector.

Returns

CAnnotationReferenceVect The vector of child widgets' references

getDefaultAppearanceString()

```
virtual bool JawsMako::IFormField::getDefaultAppearanceString (
            U8String & defaultAppearanceString ) const  [pure virtual]
```

Get the default appearance string for variable text, if set.

Parameters

defaultAppearanceString	A reference to receive the default appearance string

Returns

bool true if the field defines a default appearance. If false, the value may be inherited from a parent field

getDefaultValue()

```
virtual bool JawsMako::IFormField::getDefaultValue (
            CU8StringVect & value ) const  [pure virtual]
```

Get the default value of the field, as an array of strings.

Parameters

value	A vector to receive the value. For fields that can take multiple values, such as radio buttons or choices, there may be multiple entries. For fields requiring only a single entry.

Returns

bool True if the field has a value. If false, the value may be stored in a parent field

getFieldFlags()

```
virtual bool JawsMako::IFormField::getFieldFlags (
            uint32 & flags ) const  [pure virtual]
```

Get the field flags, if present.

Parameters

flags	Reference to receive the flags - #see eFieldFlags

Returns

bool False if the flags have not been set for this field, otherwise true. If false, the parent field should be consulted.

getFieldId()

```
virtual DOMid JawsMako::IFormField::getFieldId ( ) const  [pure virtual]
```

Get the unique id for this field. This id is preserved across cloning, and therefore all clones of this IFormField will have the same id. This id is particularly useful when merging fields from one form to another.

Returns

DOMid The unique id for this field

getFieldType()

```
virtual eFieldType JawsMako::IFormField::getFieldType ( ) const  [pure virtual]
```

Get the type of the field. If eFTInherited is returned, the parent field should be consulted to determine the type.

Returns

eFieldType The field type

getOptions()

```
virtual bool JawsMako::IFormField::getOptions (
            CFieldOptionVect & options ) const  [pure virtual]
```

Get the options for this field.

Parameters

options	Reference to receive the options

Returns

bool true if the option is present. If false, the options may be inherited and the parent field should be consulted.

getPartialName()

```
virtual U8String JawsMako::IFormField::getPartialName ( ) const  [pure virtual]
```

Get the partial name of the field, if defined. If not present, an empty string will be returned.

Returns

U8String The partial name of the field

getQuadding()

```
virtual eQuadding JawsMako::IFormField::getQuadding ( ) const  [pure virtual]
```

Get the Quadding (Justification) for variable text.

Returns

eQuadding The quadding to use, or eQInherited if it is inherited from a parent field

getValue()

```
virtual bool JawsMako::IFormField::getValue (
            CU8StringVect & value ) const  [pure virtual]
```

Get the value of the field, as an array of strings.

Parameters

value	A vector to receive the value. For fields that can take multiple values, such as radio buttons or choices, there may be multiple entries. For fields requiring only a single entry.

Returns

bool True if the field has a value. If false, the value may be stored in a parent field

removeChildField()

```
virtual void JawsMako::IFormField::removeChildField (
            const IFormFieldPtr & field )  [pure virtual]
```

Remove a child field.

Parameters

field	Field to be removed

removeChildWidget() [1/2]

```
virtual void JawsMako::IFormField::removeChildWidget (
            const IAnnotationReferencePtr & widgetReference )  [pure virtual]
```

Remove the given child widget from the field, by reference.

Parameters

widgetReference	The widget, by reference, to be removed

removeChildWidget() [2/2]

```
virtual void JawsMako::IFormField::removeChildWidget (
            const IWidgetAnnotationPtr & widget )  [pure virtual]
```

Remove the given child widget from the field.

Parameters

widget	The widget to be removed

setDefaultAppearanceString()

```
virtual void JawsMako::IFormField::setDefaultAppearanceString (
            const U8String & defaultAppearanceString )  [pure virtual]
```

Set the default appearance string for variable text.

Parameters

defaultAppearanceString	The desired default appearance string

setDefaultValue() [1/2]

```
virtual void JawsMako::IFormField::setDefaultValue (
            const CU8StringVect & value )  [pure virtual]
```

Set the value of the field, as an array of strings.

For fields that can take multiple values, eg radio buttons, there will be multiple entries. For fields requiring only a single value, a single entry must be specified. Pass an empty vector to clear the value.

Parameters

value	The value.

setDefaultValue() [2/2]

```
virtual void JawsMako::IFormField::setDefaultValue (
            const U8String & value )  [inline], [virtual]
```

Convenience form for fields that only need a single value.

Parameters

value	The string value for the field.

setFieldFlags()

```
virtual void JawsMako::IFormField::setFieldFlags (
            uint32 flags )  [pure virtual]
```

Set the field flags - #see eFieldFlags.

Parameters

flags	The desired flags.

setOptions()

```
virtual void JawsMako::IFormField::setOptions (
            const CFieldOptionVect & options )  [pure virtual]
```

Set the options for this field.

Parameters

options The desired options

setPartialName()

```
virtual void JawsMako::IFormField::setPartialName (
            const U8String & name )  [pure virtual]
```

Set the partial name of the field.

Parameters

name	The partial name

setQuadding()

```
virtual void JawsMako::IFormField::setQuadding (
            eQuadding quadding )  [pure virtual]
```

Set the Quadding (Justification) for variable text.

Parameters

quadding	The quadding to use

setValue() [1/2]

```
virtual void JawsMako::IFormField::setValue (
            const CU8StringVect & value )  [pure virtual]
```

Set the value of the field, as an array of strings. For fields that can take multiple values, eg radio buttons, there will be multiple entries. For fields requiring only a single value, a single entry will be returned. If the field has no value an empty array will be returned.

Parameters

value	The value.

setValue() [2/2]

```
virtual void JawsMako::IFormField::setValue (
            const U8String & value )  [inline], [virtual]
```

Convenience form for fields that only need a single value.

Parameters

value	The string value for the field.

widgetInSubtree() [1/2]

```
virtual bool JawsMako::IFormField::widgetInSubtree (
            const IAnnotationReferencePtr & widgetReference ) const  [pure virtual]
```

Search the IFormField subtree for a widget, by reference The search recurses through the children.

Parameters

widgetReference	The widget, by reference, to search for

Returns

bool True if found, otherwise false

widgetInSubtree() [2/2]

```
virtual bool JawsMako::IFormField::widgetInSubtree (
            const IWidgetAnnotationPtr & widget ) const  [pure virtual]
```

Search the IFormField subtree for a widget The search recurses through the children.

Parameters

widget	The widget to search for

Returns

bool True if found, otherwise false

The documentation for this class was generated from the following file:

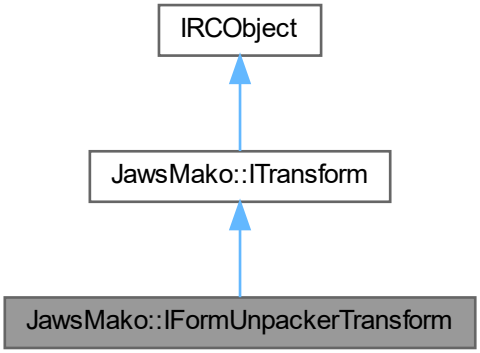
- interactive.h

7.304 JawsMako::IFormUnpackerTransform Class Reference

A transform for unpacking an IDOMFormInstance directly into the DOM tree. That is, in the DOM tree the IDOMFormInstance is replaced with the unpacked contents of the referenced IDOMForm.

```
#include <transforms.h>
```

Inheritance diagram for JawsMako::IFormUnpackerTransform:



Public Member Functions

- virtual void **setUnpackFlaggedFormInstancesOnly** (bool unpack)=0

  Sets whether or not only IDOMFormInstances with the eNodeUnpackFlag flag set will be unpacked. The default is false.

Public Member Functions inherited from JawsMako::ITransform

- virtual IDOMBrushPtr transform (const IDOMBrushPtr &brush, eBrushUsage usage=eBUGeneral, const CTransformState &state=CTransformState())=0

  Apply the transform to the given brush, if applicable. These transforms are thread safe.
- virtual IDOMImagePtr transform (const IDOMImagePtr &image, const CTransformState &state=CTransformState())=0

  Apply the transform to the given image, if applicable. These transforms are thread safe.
- virtual IDOMColorPtr transform (const IDOMColorPtr &color, const CTransformState &state=CTransformState())=0

  Apply the transform to the given color, if applicable. These transforms are thread safe.
- virtual IDOMColorSpacePtr transform (const IDOMColorSpacePtr &colorSpace, const CTransformState &state=CTransformState())=0

  Apply the transform to the given colorspace, if applicable. These transforms are thread safe.
- virtual IDOMNodePtr transform (const IDOMNodePtr &node, bool &changed, bool transformChildren=true, const CTransformState &state=CTransformState())=0

  Apply the transform to the given node, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time.
- virtual void transformPage (const IPagePtr &page, bool transformContent=true, bool transform↩ Annotations=true)=0

  Apply the transform to the given page, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time. The transform will also apply to the annotations appearances.
- virtual void **flushCaches** ()=0

  Flush the caches used by the transform. Most transforms cache recently transformed results to improve the performance of repeated transformations of equivalent results. However, it is possible that some cached results may point to entities that no longer exist, such as content inside an XPS file that no longer exists. If you are deleting or replacing files where transforms have been used, it is advisable to invoke this routine to clear the caches.
- virtual void setProgressMonitor (const IProgressMonitorPtr &progressMonitor)=0

  Set the IProgressMonitor object for this transform to allow for monitoring the progress of the transform.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IFormUnpackerTransformPtr create (const IJawsMakoPtr &jawsMako, const IAbort↩ Ptr &abort=IAbortPtr())

  Create the transform.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.304.1 Detailed Description

A transform for unpacking an IDOMFormInstance directly into the DOM tree. That is, in the DOM tree the IDOMFormInstance is replaced with the unpacked contents of the referenced IDOMForm.

The results will be unpacked into the simplest grouping node possible; either an IDOMGroup, IDOMCanvas or IDOMTransparencyGroup as required.

Useful for consumers that cannot directly handle PDF and PS Form objects.

Note that for this transform, transformPage() does not operate on annotations as the annotation appearances are themselves, forms.

7.304.2 Member Function Documentation

create()

```
static JAWSMAKO_API IFormUnpackerTransformPtr JawsMako::IFormUnpackerTransform::create (
            const IJawsMakoPtr & jawsMako,
            const IAbortPtr & abort = IAbortPtr() )  [static]
```

Create the transform.

Parameters

jawsMako	The JawsMako instance.
abort	An abort callback handler.

Returns

The new instance.

The documentation for this class was generated from the following file:

- transforms.h

7.305 IFrame Class Reference

A frame, into which text can be flowed by the layout engine.

```
#include <layout.h>
```

7.305.1 Detailed Description

A frame, into which text can be flowed by the layout engine.

The documentation for this class was generated from the following file:

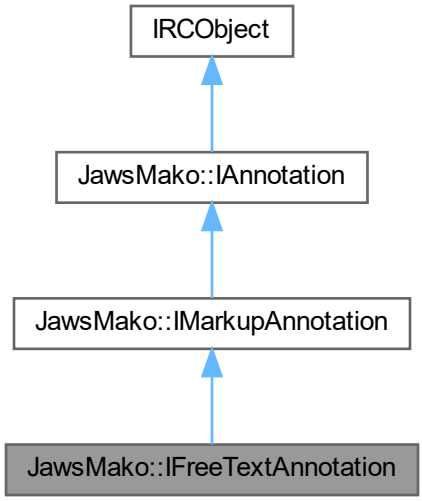
- layout.h

7.306 JawsMako::IFreeTextAnnotation Class Reference

A generic interface class for a free text annotation It is intended that future releases of JawsMako will extend this interface.

`#include <interactive.h>`

Inheritance diagram for JawsMako::IFreeTextAnnotation:



Public Member Functions

- virtual CRectInset getRectInset () const =0

  Get the rect inset describing, relative to the annotation rect, the free text area within the annotation. The points are relative to the annotation rect.
- virtual CFPointVect getCalloutLine () const =0

  Get the points that comprise the callout line of the free text annotation.

Public Member Functions inherited from JawsMako::IMarkupAnnotation

- virtual U8String getAuthor () const =0

  Get the Author of the markup annotation. This is the "T" entry in the annotation, which by convention is the author.
- virtual void setAuthor (const U8String &author)=0

  Set the Author of the markup annotation. This is the "T" entry in the annotation, which by convention is the author.
- virtual IEDLTimePtr getCreationTime () const =0

  Get the creation date and time of the annotation, if present.
- virtual void setCreationTime (const IEDLTimePtr &creationTime)=0

  Set the creation date and time of the annotation.
- virtual float getOpacity () const =0

  Get the opacity of the markup annotation.
- virtual void setOpacity (float opacity)=0

  Set the opacity of the markup annotation.
- virtual IAnnotationReferencePtr getPopupReference () const =0

  Get a reference to the popup, if present.
- virtual void setPopup (const IAnnotationReferencePtr &popupReference)=0

  Set or clear the popup, by reference.
- virtual void setPopup (const IPopupAnnotationPtr &popup)=0

  Set a reference to a popup, if present.
- virtual IAnnotationAppearancePtr createNormalAppearance () const =0

  Create a basic appearance given the current annotation's parameters. This can then be installed by using IAnnotation::addAppearance().

Public Member Functions inherited from JawsMako::IAnnotation

- virtual eAnnotationType getType () const =0

  Get the type of the annotation.
- virtual const FRect & getRect () const =0

  Get the rect in which the appearances should be displayed.
- virtual void setRect (const FRect &rect)=0

  Set the rect in which the appearances should be displayed.
- virtual U8String getContents () const =0

  Get the Contents entry in UTF-8.
- virtual void setContents (const U8String &contents)=0

  Set the Contents entry in UTF-8.
- virtual IDOMColorPtr getColor () const =0

  Get the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.
- virtual void setColor (const IDOMColorPtr &color)=0

  Set the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.
- virtual IEDLTimePtr getModificationTime () const =0

  Get the Modification date and time of the annotation, if present.
- virtual void setModificationTime (const IEDLTimePtr &modificationTime)=0

  Set the Modification date and time of the annotation.
- virtual CAnnotationBorder getBorder () const =0

  Get the annotation's border. See CAnnotationBorder for details.
- virtual void setBorder (const CAnnotationBorder &border)=0

  Set the annotation's border.
- virtual uint32 getFlags () const =0

  Get the annotation flags. To interpret these flags please see section 8.4.2 "Annotation Flags" in the PDF 1.7 specification.

- virtual void setFlags (uint32 flags)=0

  Set the annotation flags. Please see section 8.4.2 "Annotation Flags" for details about these flags.
- virtual void rotate (uint16 rotate, double pageWidth, double pageHeight)=0

  Rotate the annotation clockwise as if the page was rotated by the same amount.
- virtual CAnnotationAppearanceVect getAppearances () const =0

  Return all the annotation appearances in a vector.
- virtual void **removeAppearances** ()=0

  Remove all annotation appearances.
- virtual U8String getState () const =0

  Get the current annotation state.
- virtual void setState (const U8String &state)=0

  Set the current annotation state.
- virtual IAnnotationAppearancePtr getAppearance (eAppearanceUsage usage, const U8String &state=U8String()) const =0

  Fetch the annotation appearance that should be used for the given annotation usage and state according to the following rules:
- virtual void addAppearance (const IAnnotationAppearancePtr &appearance)=0

  Add or replace an appearance.
- virtual bool hasNormalAppearance () const =0

  Does the annotation have a normal appearance? Convenience utility function.
- virtual IAnnotationPtr clone () const =0

  Clone the annotation. This is a deep clone. The annotation reference will remain the same.
- virtual bool matchesReference (const IAnnotationReferencePtr &reference) const =0

  Does this annotation match the given IAnnotationReference?
- virtual IAnnotationReferencePtr getReference () const =0

  Get a reference that can be used to refer to this annotation.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Public Types inherited from JawsMako::IAnnotation

- enum eAnnotationType {
  eAT3D , eATCaret , eATCircle , eATFileAttachment ,
  eATFreeText , eATHighlight , eATInk , eATLine ,
  eATLink , eATMovie , eATPolygon , eATPolyLine ,
  eATPopup , eATPrinterMark , eATProjection , eATRedact ,
  eATRichMedia , eATScreen , eATSound , eATSquare ,
  eATSquiggly , eATStamp , eATStrikeOut , eATText ,
  eATTrapNet , eATUnderline , eATWatermark , eATWidget ,
  eATOther }

  Types of annotations, listed with the earliest version of PDF that supported them.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.306.1 Detailed Description

A generic interface class for a free text annotation It is intended that future releases of JawsMako will extend this interface.

7.306.2 Member Function Documentation

getCalloutLine()

```
virtual CFPointVect JawsMako::IFreeTextAnnotation::getCalloutLine ( ) const  [pure virtual]
```

Get the points that comprise the callout line of the free text annotation.

Returns

CFPointVect A vector of the points that comprise the callout line. An empty vector will be returned if there is no callout line

getRectInset()

```
virtual CRectInset JawsMako::IFreeTextAnnotation::getRectInset ( ) const  [pure virtual]
```

Get the rect inset describing, relative to the annotation rect, the free text area within the annotation. The points are relative to the annotation rect.

Returns

CRectInset The rect inset

The documentation for this class was generated from the following file:

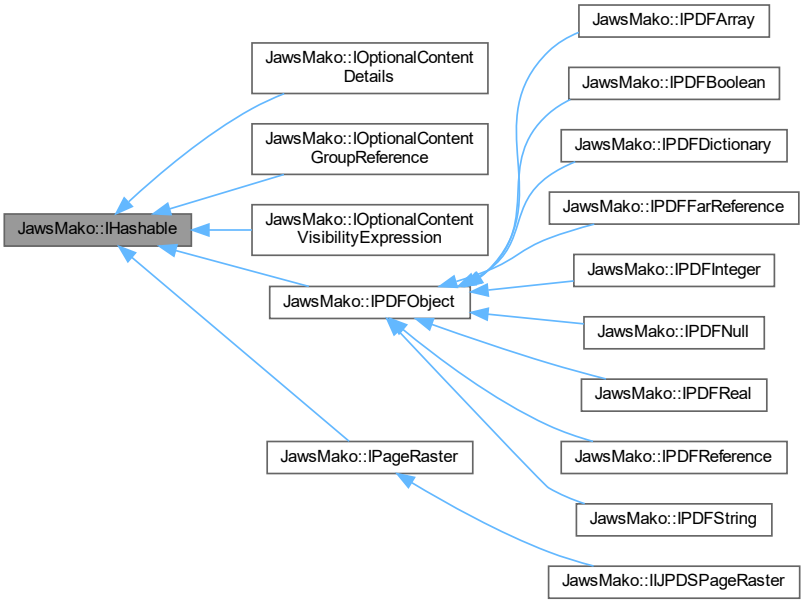
- interactive.h

7.307 JawsMako::IHashable Class Reference

Simple interface to provide a consistent hashing method for Mako objects.

`#include <types.h>`

Inheritance diagram for JawsMako::IHashable:



Public Member Functions

- virtual uint64 hash () const

  Obtain a 64-bit hash of the receiving object.
- virtual void updateHash (uint64 &hash) const =0

  Update the given hash to include the receiver.

7.307.1 Detailed Description

Simple interface to provide a consistent hashing method for Mako objects.

7.307.2 Member Function Documentation

hash()

`virtual uint64 JawsMako::IHashable::hash ( ) const  [inline], [virtual]`

Obtain a 64-bit hash of the receiving object.

Returns

uint64 The hash.

updateHash()

```
virtual void JawsMako::IHashable::updateHash (
            uint64 & hash ) const  [pure virtual]
```

Update the given hash to include the receiver.

Parameters

hash	The hash to update.

The documentation for this class was generated from the following file:

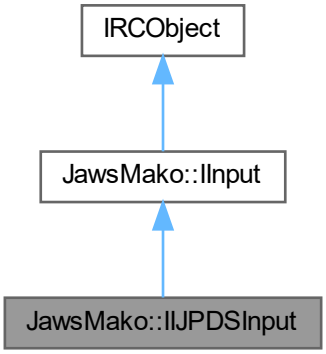
- types.h

7.308 JawsMako::IIJPDSInput Class Reference

An instance of the JawsMako IJPDS input class.

`#include <ijpdsinput.h>`

Inheritance diagram for JawsMako::IIJPDSInput:



Static Public Member Functions

- static JAWSMAKO_API IIJPDSInputPtr create (const IJawsMakoPtr &jawsMako, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create an input for reading source documents in IJPDS format.

Static Public Member Functions inherited from JawsMako::IInput

- static JAWSMAKO_API IInputPtr create (const IJawsMakoPtr &jawsMako, eFileFormat format, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create an input for reading source documents in the given format. The following formats are currently supported:

Additional Inherited Members

Public Member Functions inherited from JawsMako::IInput

- virtual IDocumentAssemblyPtr open (const U8String &pathToFile)=0

  Open a file on disk, returning the IDocumentAssembly representing the contents.
- virtual IDocumentAssemblyPtr open (const String &pathToFile)=0

  Open a file on disk, returning the IDocumentAssembly representing the contents. Takes a wide character string.
- virtual IDocumentAssemblyPtr open (const IInputStreamPtr &inputStream)=0

  Open a stream, returning the IDocumentAssembly representing the contents.
- virtual void setSequentialMode (bool sequential)=0

  Set/unset sequential mode on this input.
- virtual void setParameter (const U8String &param, const U8String &value)=0

  Apply a key value pair output parameter with a string value. The parameter name is case insensitive. Please refer to the supplied documentation for the details of the available parameters and their ranges.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.308.1 Detailed Description

An instance of the JawsMako IJPDS input class.

7.308.2 Member Function Documentation

create()

```
static JAWSMAKO_API IIJPDSInputPtr JawsMako::IIJPDSInput::create (
            const IJawsMakoPtr & jawsMako,
            const IProgressMonitorPtr & progressMonitor = IProgressMonitorPtr() )  [static]
```

Create an input for reading source documents in IJPDS format.

Parameters

jawsMako	The IJawsMako object
progressMonitor	A smart pointer to an IProgressMonitor object which can be NULL if no such object was passed in.

Returns

IIJPDSInputPtr the IJPDS input

The documentation for this class was generated from the following file:

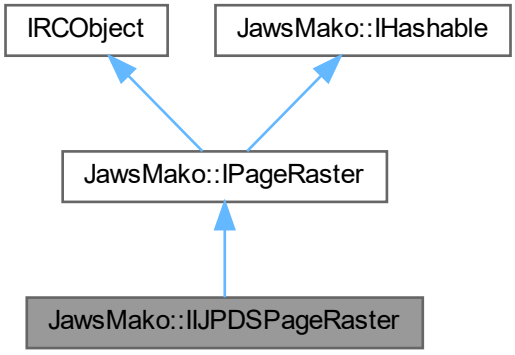
- ijpdsinput.h

7.309 JawsMako::IIJPDSPageRaster Class Reference

An instance of the JawsMako IJPDS page raster that contains the source input page number and rip number.

`#include <ijpdsinput.h>`

Inheritance diagram for JawsMako::IIJPDSPageRaster:



Public Member Functions

- virtual int32 **getInputPageNo** () const =0

  Get the page number from the source document where this page raster is derived from.
- virtual int32 **getRipNo** () const =0

  Get the rip number from the source document where this page raster is derived from.
- virtual uint16 **getUserOutputSignals** () const =0

  Get the UserOutputSignal flags for this page.

Public Member Functions inherited from JawsMako::IPageRaster

- virtual uint32 **getWidth** () const =0

  Get the width of the raster image.
- virtual uint32 **getHeight** () const =0

  Get the height of the raster image.
- virtual uint32 **getResX** () const =0

  Get the resolution of the raster image along the X-axis.
- virtual uint32 **getResY** () const =0

  Get the resolution of the raster image along the Y-axis.
- virtual uint32 **getBPC** () const =0

  Get the number of bits per component.
- virtual uint32 **getNumComponents** () const =0

  Get the number of component per pixel.
- virtual uint32 **getRawBytesPerRow** () const =0

  Get the number of bytes per row.
- virtual bool **isBlank** () const =0

  Return true if the page is unmarked.
- virtual uint8 * **getFrameBuffer** () const =0

  Get the address the raw frame buffer.
- virtual IDOMImagePtr getAsDOMImage () const =0

  Return an IDOMImage equivalent of the raster image.
- virtual void **releaseFrameBuffer** ()=0

  Release the frame buffer for this raster.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from JawsMako::IHashable

- virtual uint64 hash () const

  Obtain a 64-bit hash of the receiving object.
- virtual void updateHash (uint64 &hash) const =0

  Update the given hash to include the receiver.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.309.1 Detailed Description

An instance of the JawsMako IJPDS page raster that contains the source input page number and rip number.

The documentation for this class was generated from the following file:

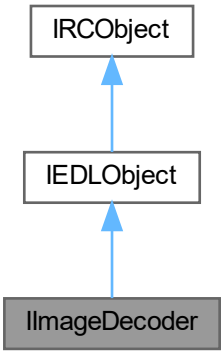
- ijpdsinput.h

7.310 IImageDecoder Class Reference

IImageDecoder returns IImageFrame objects as requested by the client. This object knows about the imageformat internals and knows how to unpack the image.

`#include <iimagecodec.h>`

Inheritance diagram for IImageDecoder:



Public Member Functions

- **IImageDecoder** ()

  Constructor.
- virtual ~**IImageDecoder** ()

  Destructor.
- virtual uint32 **getNumFrames** ()

  Returns the number of frames in this image file. Mako does not currently support reading extra frames in image files; this API is present for future expansion.
- virtual IImageFramePtr getFrame (uint32 frameNo=0)=0

  Get the frame from the decoder for the image. Mako does not currently support reading extra frames in image files; this API is present for future expansion.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.310.1 Detailed Description

IImageDecoder returns IImageFrame objects as requested by the client. This object knows about the imageformat internals and knows how to unpack the image.

7.310.2 Member Function Documentation

getFrame()

```
virtual IImageFramePtr IImageDecoder::getFrame (
            uint32 frameNo = 0 )  [pure virtual]
```

Get the frame from the decoder for the image. Mako does not currently support reading extra frames in image files; this API is present for future expansion.

Returns

IImageFramePtr The image frame

The documentation for this class was generated from the following file:

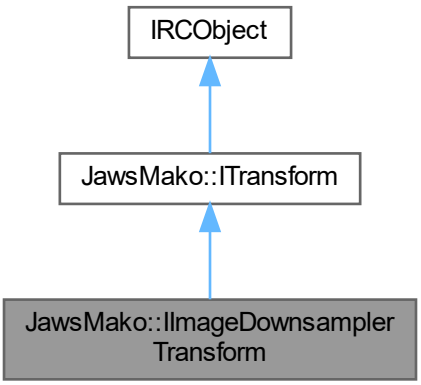
- iimagecodec.h

7.311 JawsMako::IImageDownsamplerTransform Class Reference

A transform for downsampling images above a given effective resolution to a desired target effective resolution.

`#include <transforms.h>`

Inheritance diagram for JawsMako::IImageDownsamplerTransform:



Public Member Functions

- virtual void **setColorDownsamplingMethod** (IDOMImageDownsamplerFilter::eDownsamplingMethod method)=0

  Set the desired downsampling method for color images. The default is bicubic.
- virtual IDOMImageDownsamplerFilter::eDownsamplingMethod **getColorDownsamplingMethod** () const =0

  Get the downsampling method to be used for color images.
- virtual void **setGrayDownsamplingMethod** (IDOMImageDownsamplerFilter::eDownsamplingMethod method)=0

  Set the desired downsampling method for gray images The default is bicubic.
- virtual IDOMImageDownsamplerFilter::eDownsamplingMethod **getGrayDownsamplingMethod** () const =0

  Get the downsampling method to be used for gray images.
- virtual void **setMonoDownsamplingMethod** (IDOMImageDownsamplerFilter::eDownsamplingMethod method)=0

  Set the desired downsampling method for monochrome images. The default is subsample. NB: using any other method other than subsampling for monochrome images will result in grayscale output.
- virtual IDOMImageDownsamplerFilter::eDownsamplingMethod **getMonoDownsamplingMethod** () const =0

  Get the downsampling method to be used for mono images.
- virtual void **setTargetResolution** (float resolution)=0

  Set a blanket target resolution (dpi) for downsampling all images. This will also set the threshold resolution to the same value.
- virtual void **setThresholdResolution** (float resolution)=0

  Set a blanket threshold resolution (dpi) for downsampling all images. Only images with an effective resolution of at least this value will be downsampled.
- virtual void **setColorTargetResolution** (float resolution)=0

Set a target resolution (dpi) for downsampling color images. This will also set the threshold resolution to the same value. Setting this to 0.0 will result in no downsampling for this image type. The default is 0.0.

- virtual float **getColorTargetResolution** ()=0

  Get the target resolution (dpi) for downsampling color images.

- virtual void **setColorThresholdResolution** (float resolution)=0

  Set a threshold resolution (dpi) for downsampling color images. Only images with an effective resolution of at least this value will be downsampled.

- virtual float **getColorThresholdResolution** ()=0

  Get the theshold resolution (dpi) for downsampling color images.

- virtual void **setGrayTargetResolution** (float resolution)=0

  Set a target resolution (dpi) for downsampling gray images. This will also set the threshold resolution to the same value. The default is 300dpi. Setting this to 0.0 will result in no downsampling for this image type. The default is 0.0.

- virtual float **getGrayTargetResolution** ()=0

  Get the target resolution (dpi) for downsampling gray images.

- virtual void **setGrayThresholdResolution** (float resolution)=0

  Set a threshold resolution (dpi) for downsampling gray images. Only images with an effective resolution of at least this value will be downsampled.

- virtual float **getGrayThresholdResolution** ()=0

  Get the theshold resolution (dpi) for downsampling gray images.

- virtual void **setMonoTargetResolution** (float resolution)=0

  Set a target resolution (dpi) for downsampling monochrome images. This will also set the threshold resolution to the same value. The default is 1200dpi. Setting this to 0.0 will result in no downsampling for this image type. The default is 0.0.

- virtual float **getMonoTargetResolution** ()=0

  Get the target resolution (dpi) for downsampling mono images.

- virtual void **setMonoThresholdResolution** (float resolution)=0

  Set a threshold resolution (dpi) for downsampling monochrome images. Only images with an effective resolution of at least this value will be downsampled.

- virtual float **getMonoThresholdResolution** ()=0

  Get the theshold resolution (dpi) for downsampling mono images.

- virtual void setDownsampleMaskedImages (bool downsampleMaskedImages)=0

  Set whether or not to downsample masked images.

- virtual void setUseMaskResolutionForMaskedImages (bool useMaskResolutionSettingForMaskedImages)=0

  Set whether or not the mask resolution setting should be applied to the image portion of a masked image.

Public Member Functions inherited from JawsMako::ITransform

- virtual IDOMBrushPtr transform (const IDOMBrushPtr &brush, eBrushUsage usage=eBUGeneral, const CTransformState &state=CTransformState())=0

  Apply the transform to the given brush, if applicable. These transforms are thread safe.

- virtual IDOMImagePtr transform (const IDOMImagePtr &image, const CTransformState &state=CTransformState())=0

  Apply the transform to the given image, if applicable. These transforms are thread safe.

- virtual IDOMColorPtr transform (const IDOMColorPtr &color, const CTransformState &state=CTransformState())=0

  Apply the transform to the given color, if applicable. These transforms are thread safe.

- virtual IDOMColorSpacePtr transform (const IDOMColorSpacePtr &colorSpace, const CTransformState &state=CTransformState())=0

  Apply the transform to the given colorspace, if applicable. These transforms are thread safe.

- virtual IDOMNodePtr transform (const IDOMNodePtr &node, bool &changed, bool transformChildren=true, const CTransformState &state=CTransformState())=0

  Apply the transform to the given node, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time.

- virtual void transformPage (const IPagePtr &page, bool transformContent=true, bool transform↩ Annotations=true)=0

  Apply the transform to the given page, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time. The transform will also apply to the annotations appearances.
- virtual void **flushCaches** ()=0

  Flush the caches used by the transform. Most transforms cache recently transformed results to improve the performance of repeated transformations of equivalent results. However, it is possible that some cached results may point to entities that no longer exist, such as content inside an XPS file that no longer exists. If you are deleting or replacing files where transforms have been used, it is advisable to invoke this routine to clear the caches.
- virtual void setProgressMonitor (const IProgressMonitorPtr &progressMonitor)=0

  Set the IProgressMonitor object for this transform to allow for monitoring the progress of the transform.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IImageDownsamplerTransformPtr create (const IJawsMakoPtr &jawsMako, const IAbortPtr &abort=IAbortPtr())

  Create the transform.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.311.1 Detailed Description

A transform for downsampling images above a given effective resolution to a desired target effective resolution.

7.311.2 Member Function Documentation

create()

```
static JAWSMAKO_API IImageDownsamplerTransformPtr JawsMako::IImageDownsamplerTransform::create
(
            const IJawsMakoPtr & jawsMako,
            const IAbortPtr & abort = IAbortPtr() )  [static]
```

Create the transform.

Parameters

jawsMako	The JawsMako instance.

Returns

The new instance.

setDownsampleMaskedImages()

```
virtual void JawsMako::IImageDownsamplerTransform::setDownsampleMaskedImages (
            bool downsampleMaskedImages )  [pure virtual]
```

Set whether or not to downsample masked images.

```
This applies to cases where an image is masked by a separate
masked image, such as types of PDF masked or soft-masked images.
These are represented in the DOM using IDOMMaskedBrush, where the
sub-brush is an image.

The default is true.
```

setUseMaskResolutionForMaskedImages()

```
virtual void JawsMako::IImageDownsamplerTransform::setUseMaskResolutionForMaskedImages (
            bool useMaskResolutionSettingForMaskedImages )  [pure virtual]
```

Set whether or not the mask resolution setting should be applied to the image portion of a masked image.

This applies to cases where an image is masked by a separate masked image, such as types of PDF masked or soft-masked images. These are represented in the DOM using IDOMMaskedBrush, where the sub-brush is an image.

If false, then the mask and the image are evaluated separately and a downsampling resolution and threshold are chosen. For the mask, this is normally either grayscale or monochrome. The image data can be anything. In this mode it is possible for the downsampled image and mask to be downsampled to different resolutions.

If true, then whatever target resolution and threshold is applied to the mask will also be applied to the image samples. If these images have the same effective resolution before downsampling, then they will also share the same effective resolution after downsampling.

The default is false.

The documentation for this class was generated from the following file:

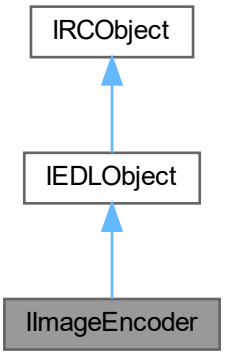
- transforms.h

7.312 IImageEncoder Class Reference

IImageEncoder accepts IImageFrame objects and streams it out to an image format.

`#include <iimagecodec.h>`

Inheritance diagram for IImageEncoder:



Public Member Functions

- **IImageEncoder** ()
  Constructor.
- virtual ~**IImageEncoder** ()
  Destructor.
- virtual IImageFrameWriterPtr createFrame ()=0
  Create a frame to write images from this encoder.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0
  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)
  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)
  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0
  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0
  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0
  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()
  Virtual destructor.

7.312.1 Detailed Description

IImageEncoder accepts IImageFrame objects and streams it out to an image format.

7.312.2 Member Function Documentation

createFrame()

```
virtual IImageFrameWriterPtr IImageEncoder::createFrame ( )  [pure virtual]
```

Create a frame to write images from this encoder.

Returns

IImageFrameWriterPtr The image frame

The documentation for this class was generated from the following file:

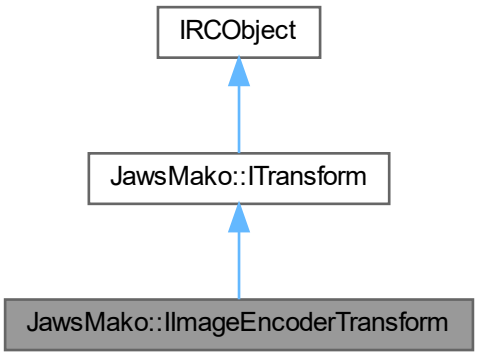
- iimagecodec.h

7.313 JawsMako::IImageEncoderTransform Class Reference

A simple transform for image encoding. Most useful for encoding abstract images such as IDOMRecombineImage, IDOMRawImage and IDOMFilteredImage as PNG, Tiff or Jpeg. Images may be color converted if they are not compatible with the desired image type.

```
#include <transforms.h>
```

Inheritance diagram for JawsMako::IImageEncoderTransform:



Public Types

- enum eEncodeFormat { }

  Available target image formats.

Public Member Functions

- virtual void **setReencodeAllImages** (bool value)=0

  Sets whether or not all images should be reencoded. The default is false; that is, only image types other than JPEG, TIFF, HDPHOTO and PNG will be re-encoded.
- virtual void **setEnableTIFFEncoding** (bool value)=0

  Set whether or not TIFF should ever be allowed. The default value is true. This is needed as some output formats do not support TIFF. If a TIFF is seen it will be re-encoded as something else.
- virtual void **setReencodeMultiImageTIFF** (bool value)=0

  Set whether or not TIFF images that use any image other than the first image in the TIFF file should be re-encoded. The default is true. Ignored if setEnableTIFFEncoding() is true, in which case all TIFF images will be re-encoded.
- virtual void setPreferredFormat (eEncodeFormat format)=0

  Set the preferred format for all images. This is only a preference and may not be honoured in all cases. The default is eEFAuto.
- virtual void setPreferredMonoFormat (eEncodeFormat format)=0

  Set the preferred format for monochrome images. This is only a preference and may not be honoured in all cases. The default is eEFAuto.
- virtual eEncodeFormat **getPreferredMonoFormat** () const =0

  Get the preferred format for monochrome images.
- virtual void setPreferredGrayFormat (eEncodeFormat format)=0

  Set the preferred format for gray images. This is only a preference and may not be honoured in all cases. The default is eEFAuto.
- virtual eEncodeFormat **getPreferredGrayFormat** () const =0

  Get the preferred format for gray images.
- virtual void setPreferredColorFormat (eEncodeFormat format)=0

  Set the preferred format for color images. The default is eEFAuto.
- virtual eEncodeFormat **getPreferredColorFormat** () const =0

  Get the preferred format for color images.
- virtual void setJPEGQuality (uint8 quality)=0

  Set the JPEG encoding quality when JPEG encoding is used.
- virtual uint8 **getJPEGQuality** () const =0

  Get the currently set JPEG encoding quality, where 1 being lowest quality and 5 being highest quality.
- virtual void setTIFFCompression (IDOMTIFFImage::eTIFFCompression scheme)=0

  Set the preferred TIFF compression when TIFF encoding is used.
- virtual void setMonoTIFFCompression (IDOMTIFFImage::eTIFFCompression scheme)=0

  Set the preferred TIFF compression when TIFF encoding is used for monochome images. Any compresion scheme may be used.
- virtual void setGrayTIFFCompression (IDOMTIFFImage::eTIFFCompression scheme)=0

  Set the preferred TIFF compression when TIFF encoding is used for gray images. Only PackBits, LZW or None may be used.
- virtual void setColorTIFFCompression (IDOMTIFFImage::eTIFFCompression scheme)=0

  Set the preferred TIFF compression when TIFF encoding is used for color images. Only PackBits, LZW or None may be used.
- virtual void **setEncodeSolidColorMaskedBrushesAsSingleImage** (bool encode)=0

  Set whether or not IDOMMaskedBrushes where the sub-brush is a solid color should be encoded as an image with an alpha channel, rather than encoding the image separately. The default is true.

Public Member Functions inherited from JawsMako::ITransform

- virtual IDOMBrushPtr transform (const IDOMBrushPtr &brush, eBrushUsage usage=eBUGeneral, const CTransformState &state=CTransformState())=0

  Apply the transform to the given brush, if applicable. These transforms are thread safe.
- virtual IDOMImagePtr transform (const IDOMImagePtr &image, const CTransformState &state=CTransformState())=0

  Apply the transform to the given image, if applicable. These transforms are thread safe.
- virtual IDOMColorPtr transform (const IDOMColorPtr &color, const CTransformState &state=CTransformState())=0

  Apply the transform to the given color, if applicable. These transforms are thread safe.
- virtual IDOMColorSpacePtr transform (const IDOMColorSpacePtr &colorSpace, const CTransformState &state=CTransformState())=0

  Apply the transform to the given colorspace, if applicable. These transforms are thread safe.
- virtual IDOMNodePtr transform (const IDOMNodePtr &node, bool &changed, bool transformChildren=true, const CTransformState &state=CTransformState())=0

  Apply the transform to the given node, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time.
- virtual void transformPage (const IPagePtr &page, bool transformContent=true, bool transform↩ Annotations=true)=0

  Apply the transform to the given page, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time. The transform will also apply to the annotations appearances.
- virtual void **flushCaches** ()=0

  Flush the caches used by the transform. Most transforms cache recently transformed results to improve the performance of repeated transformations of equivalent results. However, it is possible that some cached results may point to entities that no longer exist, such as content inside an XPS file that no longer exists. If you are deleting or replacing files where transforms have been used, it is advisable to invoke this routine to clear the caches.
- virtual void setProgressMonitor (const IProgressMonitorPtr &progressMonitor)=0

  Set the IProgressMonitor object for this transform to allow for monitoring the progress of the transform.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IImageEncoderTransformPtr create (const IJawsMakoPtr &jawsMako, const IAbortPtr &abort=IAbortPtr())

  Create the transform.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.313.1 Detailed Description

A simple transform for image encoding. Most useful for encoding abstract images such as IDOMRecombineImage, IDOMRawImage and IDOMFilteredImage as PNG, Tiff or Jpeg. Images may be color converted if they are not compatible with the desired image type.

7.313.2 Member Enumeration Documentation

eEncodeFormat

```
enum JawsMako::IImageEncoderTransform::eEncodeFormat
```

Available target image formats.

Enumerator

eEFJPEG	Attempt to choose an appropriate format for each image.

7.313.3 Member Function Documentation

create()

```
static JAWSMAKO_API IImageEncoderTransformPtr JawsMako::IImageEncoderTransform::create (
            const IJawsMakoPtr & jawsMako,
            const IAbortPtr & abort = IAbortPtr() )  [static]
```

Create the transform.

Parameters

jawsMako	The JawsMako instance.
abort	An abort callback handler.

Returns

The new instance.

setColorTIFFCompression()

```
virtual void JawsMako::IImageEncoderTransform::setColorTIFFCompression (
            IDOMTIFFImage::eTIFFCompression scheme )  [pure virtual]
```

Set the preferred TIFF compression when TIFF encoding is used for color images. Only PackBits, LZW or None may be used.

Parameters

scheme	The preferred TIFF compression scheme. May not always be used as not all compression types can be used for all types.

setGrayTIFFCompression()

```
virtual void JawsMako::IImageEncoderTransform::setGrayTIFFCompression (
            IDOMTIFFImage::eTIFFCompression scheme )  [pure virtual]
```

Set the preferred TIFF compression when TIFF encoding is used for gray images. Only PackBits, LZW or None may be used.

Parameters

scheme	The preferred TIFF compression scheme. May not always be used as not all compression types can be used for all types.

setJPEGQuality()

```
virtual void JawsMako::IImageEncoderTransform::setJPEGQuality (
            uint8 quality )  [pure virtual]
```

Set the JPEG encoding quality when JPEG encoding is used.

Parameters

quality	The desired quality level, with 1 being lowest quality and 5 being highest quality.

setMonoTIFFCompression()

```
virtual void JawsMako::IImageEncoderTransform::setMonoTIFFCompression (
            IDOMTIFFImage::eTIFFCompression scheme )  [pure virtual]
```

Set the preferred TIFF compression when TIFF encoding is used for monochome images. Any compresion scheme may be used.

Parameters

scheme	The preferred TIFF compression scheme. May not always be used as not all compression types can be used for all types.

setPreferredColorFormat()

```
virtual void JawsMako::IImageEncoderTransform::setPreferredColorFormat (
            eEncodeFormat format )  [pure virtual]
```

Set the preferred format for color images. The default is eEFAuto.

Parameters

format	The preferred format.

setPreferredFormat()

```
virtual void JawsMako::IImageEncoderTransform::setPreferredFormat (
            eEncodeFormat format ) [pure virtual]
```

Set the preferred format for all images. This is only a preference and may not be honoured in all cases. The default is eEFAuto.

Parameters

format	The preferred format.

setPreferredGrayFormat()

```
virtual void JawsMako::IImageEncoderTransform::setPreferredGrayFormat (
            eEncodeFormat format ) [pure virtual]
```

Set the preferred format for gray images. This is only a preference and may not be honoured in all cases. The default is eEFAuto.

Parameters

format	The preferred format.

setPreferredMonoFormat()

```
virtual void JawsMako::IImageEncoderTransform::setPreferredMonoFormat (
            eEncodeFormat format ) [pure virtual]
```

Set the preferred format for monochrome images. This is only a preference and may not be honoured in all cases. The default is eEFAuto.

Parameters

format	The preferred format.

setTIFFCompression()

```
virtual void JawsMako::IImageEncoderTransform::setTIFFCompression (
            IDOMTIFFImage::eTIFFCompression scheme ) [pure virtual]
```

Set the preferred TIFF compression when TIFF encoding is used.

Parameters

scheme	The preferred TIFF compression scheme. May not always be used as not all compression types can be used for all types.

The documentation for this class was generated from the following file:

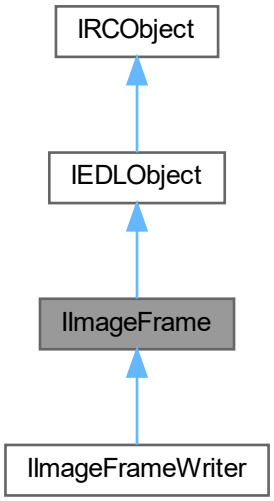
- transforms.h

7.314 IImageFrame Class Reference

IImageFrame encapsulates an EDL image with its details.

`#include <iimagecodec.h>`

Inheritance diagram for IImageFrame:



Public Member Functions

- virtual uint32 getWidth () const =0

  Gets the width of the image.
- virtual uint32 getHeight () const =0

  Gets the height of the image.
- virtual uint8 getBPS () const =0

  Gets the bits per sample of the image.
- virtual IDOMColorSpacePtr getColorSpace () const =0

Gets the color space.

- virtual eImageExtraChannelType getExtraChannelType () const =0

  Gets the type of information contained in the extra channel.
- virtual uint8 getNumExtraChannels () const

  Gets the number of image extra channels (for example an alpha or mask)
- virtual uint8 getNumChannels () const

  Gets the total number of image channels (including extra channels)
- virtual bool getHasAlphaChannel () const

  Indicates if the image has an alpha mask.
- virtual bool getHasMask (bool &isNoisy) const

  Indicates if the image has a mask (i.e. a binary alpha channel)
- virtual double getXResolution () const =0

  Gets the resolution of the image in the X direction.
- virtual double getYResolution () const =0

  Gets the resolution of the image in the Y direction.
- virtual uint32 getRawBytesPerRow () const =0

  Gets the image row size in bytes, which may include padding space.
- virtual void readScanLine (void *pRow, size_t bufferSize)=0

  Gets an image row scanline (of size given by getRawBytesPerRow())
- virtual void skipScanLines (uint32 skipCount)

  Skips the given number of scanline rows, which may be fast for some image types.
- virtual bool getEfficientlySkippable () const =0

  Determine if the image may be efficiently skipped via skipScanLines (rather than having to decode every scanline in the skipped range).

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.314.1 Detailed Description

IImageFrame encapsulates an EDL image with its details.

7.314.2 Member Function Documentation

getBPS()

```
virtual uint8 IImageFrame::getBPS ( ) const  [pure virtual]
```

Gets the bits per sample of the image.

Returns

uint8 Returns the image bits from sample.

getColorSpace()

```
virtual IDOMColorSpacePtr IImageFrame::getColorSpace ( ) const  [pure virtual]
```

Gets the color space.

Returns

IDOMColorSpacePtr The colorspace of the image.

getEfficientlySkippable()

```
virtual bool IImageFrame::getEfficientlySkippable ( ) const  [pure virtual]
```

Determine if the image may be efficiently skipped via skipScanLines (rather than having to decode every scanline in the skipped range).

Returns

bool true if the the image has an efficient skipScanLines implementation.

Implemented in IImageFrameWriter.

getExtraChannelType()

```
virtual eImageExtraChannelType IImageFrame::getExtraChannelType ( ) const  [pure virtual]
```

Gets the type of information contained in the extra channel.

Returns

eImageExtraChannelType Returns the type of information in the extra channel.

getHasAlphaChannel()

```
virtual bool IImageFrame::getHasAlphaChannel ( ) const  [inline], [virtual]
```

Indicates if the image has an alpha mask.

Returns

bool. Returns an indication of alpha mask present in the image.

getHasMask()

```
virtual bool IImageFrame::getHasMask (
            bool & isNoisy ) const  [inline], [virtual]
```

Indicates if the image has a mask (i.e. a binary alpha channel)

Parameters

isNoisy	An indicator that the alpha channel has binary values and so is acting as a mask.

Returns

bool. Returns an indication of mask present in the image

getHeight()

```
virtual uint32 IImageFrame::getHeight ( ) const  [pure virtual]
```

Gets the height of the image.

Returns

uint32 Returns the image height.

getNumChannels()

```
virtual uint8 IImageFrame::getNumChannels ( ) const  [inline], [virtual]
```

Gets the total number of image channels (including extra channels)

Returns

uint8 Returns the total number of image channels.

getNumExtraChannels()

```
virtual uint8 IImageFrame::getNumExtraChannels ( ) const  [inline], [virtual]
```

Gets the number of image extra channels (for example an alpha or mask)

Returns

uint8 Returns the number of image extra channels.

getRawBytesPerRow()

```
virtual uint32 IImageFrame::getRawBytesPerRow ( ) const  [pure virtual]
```

Gets the image row size in bytes, which may include padding space.

Returns

uint32 The image row size in bytes

getWidth()

```
virtual uint32 IImageFrame::getWidth ( ) const  [pure virtual]
```

Gets the width of the image.

Returns

uint32 Returns the image width.

getXResolution()

```
virtual double IImageFrame::getXResolution ( ) const  [pure virtual]
```

Gets the resolution of the image in the X direction.

Returns

double The resolution in the X direction.

getYResolution()

```
virtual double IImageFrame::getYResolution ( ) const  [pure virtual]
```

Gets the resolution of the image in the Y direction.

Returns

double The resolution in the Y direction.

readScanLine()

```
virtual void IImageFrame::readScanLine (
            void * pRow,
            size_t bufferSize )  [pure virtual]
```

Gets an image row scanline (of size given by getRawBytesPerRow())

Parameters

pRow	The buffer in which the row scanline will be copied into.
bufferSize	The size of the buffer. A check will be made that the buffer is large enough to receive the scanline before it is read.

Implemented in IImageFrameWriter.

skipScanLines()

```
virtual void IImageFrame::skipScanLines (
            uint32 skipCount )  [inline], [virtual]
```

Skips the given number of scanline rows, which may be fast for some image types.

Parameters

skipCount	The number of scanlines to skip..

The documentation for this class was generated from the following file:

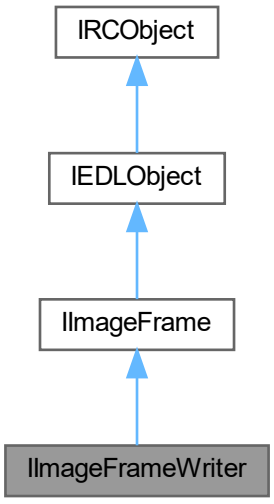
- iimagecodec.h

7.315 IImageFrameWriter Class Reference

IImageFrameWriter writes an image from an imageframe.

`#include <iimagecodec.h>`

Inheritance diagram for IImageFrameWriter:



Public Member Functions

- **IImageFrameWriter** ()

  Constructor.
- virtual ~**IImageFrameWriter** ()

  Destructor.
- virtual void writeScanLine (const void *pScanLine)=0

  Writes an image row into the image frame.
- virtual void setWidth (uint32 width)=0

  Set the width of the frame in pixels.
- virtual void setHeight (uint32 height)=0

  Set the height of the frame in pixels.
- virtual void setBPS (uint8 bps)=0

  Set the bits per sample/components of the frame.
- virtual void setImageExtraChannelType (eImageExtraChannelType extraChannelType)=0

  Set the type of the extra image channel, if present.
- virtual void setXResolution (double xRes)=0

  Set the resolution of the frame in the X direction.
- virtual void setYResolution (double yRes)=0

  Set the resolution of the frame in the Y direction.
- virtual void setColorSpace (const IDOMColorSpacePtr &colorSpace)=0

  Set the color space of the frame's pixels.
- virtual void **readScanLine** (void *pRow, size_t bufferSize)

  It is not possible to read a scanline from an image frame writer.
- virtual void **flushData** ()=0

  Finish the image and flush. Must be called after the last scanline has been written.
- virtual bool **getEfficientlySkippable** () const

  Skipping is not possible when writing images.

Public Member Functions inherited from IImageFrame

- virtual uint32 getWidth () const =0

  Gets the width of the image.
- virtual uint32 getHeight () const =0

  Gets the height of the image.
- virtual uint8 getBPS () const =0

  Gets the bits per sample of the image.
- virtual IDOMColorSpacePtr getColorSpace () const =0

  Gets the color space.
- virtual eImageExtraChannelType getExtraChannelType () const =0

  Gets the type of information contained in the extra channel.
- virtual uint8 getNumExtraChannels () const

  Gets the number of image extra channels (for example an alpha or mask)
- virtual uint8 getNumChannels () const

  Gets the total number of image channels (including extra channels)
- virtual bool getHasAlphaChannel () const

  Indicates if the image has an alpha mask.
- virtual bool getHasMask (bool &isNoisy) const

  Indicates if the image has a mask (i.e. a binary alpha channel)
- virtual double getXResolution () const =0

Gets the resolution of the image in the X direction.

- virtual double getYResolution () const =0

  Gets the resolution of the image in the Y direction.

- virtual uint32 getRawBytesPerRow () const =0

  Gets the image row size in bytes, which may include padding space.

- virtual void skipScanLines (uint32 skipCount)

  Skips the given number of scanline rows, which may be fast for some image types.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.

- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.

- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.

- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.

- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.315.1 Detailed Description

IImageFrameWriter writes an image from an imageframe.

7.315.2 Member Function Documentation

setBPS()

```
virtual void IImageFrameWriter::setBPS (
            uint8 bps )  [pure virtual]
```

Set the bits per sample/components of the frame.

Parameters

bps	The desired bit depth per sample/component

setColorSpace()

```
virtual void IImageFrameWriter::setColorSpace (
            const IDOMColorSpacePtr & colorSpace )  [pure virtual]
```

Set the color space of the frame's pixels.

Parameters

colorSpace	The desired colorspace

setHeight()

```
virtual void IImageFrameWriter::setHeight (
            uint32 height )  [pure virtual]
```

Set the height of the frame in pixels.

Parameters

height	The height

setImageExtraChannelType()

```
virtual void IImageFrameWriter::setImageExtraChannelType (
            eImageExtraChannelType extraChannelType )  [pure virtual]
```

Set the type of the extra image channel, if present.

Parameters

extraChannelType	The extra channel type

setWidth()

```
virtual void IImageFrameWriter::setWidth (
            uint32 width )  [pure virtual]
```

Set the width of the frame in pixels.

Parameters

width	The width

setXResolution()

```
virtual void IImageFrameWriter::setXResolution (
            double xRes )  [pure virtual]
```

Set the resolution of the frame in the X direction.

Parameters

xRes	The desired resolution in the X direction

setYResolution()

```
virtual void IImageFrameWriter::setYResolution (
            double yRes )  [pure virtual]
```

Set the resolution of the frame in the Y direction.

Parameters

yRes	The desired resolution in the Y direction

writeScanLine()

```
virtual void IImageFrameWriter::writeScanLine (
            const void * pScanLine )  [pure virtual]
```

Writes an image row into the image frame.

Parameters

pScanLine	A buffer that contains the scanline.

The documentation for this class was generated from the following file:

- iimagecodec.h

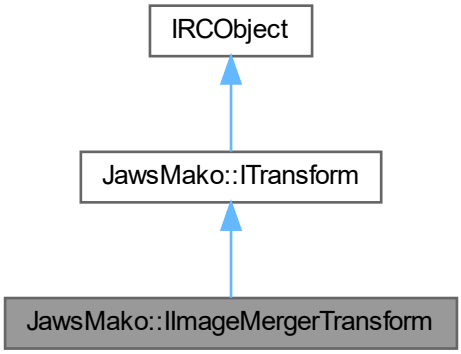
7.316 JawsMako::IImageMergerTransform Class Reference

A simple transform that looks for nearby images and attempts to glom them together in a single image. Some producers can break images up into images consisting of a single scanline; this transform attempts to put them

back together again. This transform can handle images with a mask channel, but does not attempt to merge images with an alpha channel.

`#include <transforms.h>`

Inheritance diagram for JawsMako::IImageMergerTransform:



Public Member Functions

- virtual void setAllowMaskedResults (bool allowMasked)=0

  Sets whether or a mask channel is allowed in the merged result.

Public Member Functions inherited from JawsMako::ITransform

- virtual IDOMBrushPtr transform (const IDOMBrushPtr &brush, eBrushUsage usage=eBUGeneral, const CTransformState &state=CTransformState())=0

  Apply the transform to the given brush, if applicable. These transforms are thread safe.
- virtual IDOMImagePtr transform (const IDOMImagePtr &image, const CTransformState &state=CTransformState())=0

  Apply the transform to the given image, if applicable. These transforms are thread safe.
- virtual IDOMColorPtr transform (const IDOMColorPtr &color, const CTransformState &state=CTransformState())=0

  Apply the transform to the given color, if applicable. These transforms are thread safe.
- virtual IDOMColorSpacePtr transform (const IDOMColorSpacePtr &colorSpace, const CTransformState &state=CTransformState())=0

  Apply the transform to the given colorspace, if applicable. These transforms are thread safe.
- virtual IDOMNodePtr transform (const IDOMNodePtr &node, bool &changed, bool transformChildren=true, const CTransformState &state=CTransformState())=0

  Apply the transform to the given node, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time.
- virtual void transformPage (const IPagePtr &page, bool transformContent=true, bool transform↩Annotations=true)=0

  Apply the transform to the given page, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time. The transform will also apply to the annotations appearances.
- virtual void **flushCaches** ()=0

  Flush the caches used by the transform. Most transforms cache recently transformed results to improve the performance of repeated transformations of equivalent results. However, it is possible that some cached results may point to entities that no longer exist, such as content inside an XPS file that no longer exists. If you are deleting or replacing files where transforms have been used, it is advisable to invoke this routine to clear the caches.
- virtual void setProgressMonitor (const IProgressMonitorPtr &progressMonitor)=0

  Set the IProgressMonitor object for this transform to allow for monitoring the progress of the transform.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IImageMergerTransformPtr create (const IJawsMakoPtr &jawsMako, const IAbortPtr &abort=IAbortPtr())

  Create the transform.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.316.1 Detailed Description

A simple transform that looks for nearby images and attempts to glom them together in a single image. Some producers can break images up into images consisting of a single scanline; this transform attempts to put them back together again. This transform can handle images with a mask channel, but does not attempt to merge images with an alpha channel.

7.316.2 Member Function Documentation

create()

```
static JAWSMAKO_API IImageMergerTransformPtr JawsMako::IImageMergerTransform::create (
            const IJawsMakoPtr & jawsMako,
            const IAbortPtr & abort = IAbortPtr() )  [static]
```

Create the transform.

Parameters

jawsMako	The JawsMako instance.
abort	An abort callback handler.

Returns

The new instance.

setAllowMaskedResults()

```
virtual void JawsMako::IImageMergerTransform::setAllowMaskedResults (
            bool allowMasked )  [pure virtual]
```

Sets whether or a mask channel is allowed in the merged result.

```
The merger deals with images that do not exactly abut alone a single
edge by producing a mask channel that describes the area of the
resulting image that contains merged image data. If set to false this
transform will not produce masked images, and as a result will not
be able to merge some images. The default is true.
```

The documentation for this class was generated from the following file:

- transforms.h

7.317 JawsMako::ICustomTransform::IImplementation Class Reference

Callback interface that provides methods for actually doing the work. Override the cases for the objects you wish to edit or are otherwise interested in.

```
#include <customtransform.h>
```

Public Member Functions

- virtual void transformAnnotation (IImplementation *genericImplementation, const IAnnotationPtr &annotation)

  Callback to process an annotation.
- virtual IAnnotationAppearancePtr transformAnnotationAppearance (IImplementation *genericImplementation, const IAnnotationAppearancePtr &appearance, const FRect &annotationRect)

  Callback to process an annotation appearance.
- virtual IDOMColorPtr transformColor (IImplementation *genericImplementation, const IDOMColorPtr &color, const CTransformState &state)

  Callback to process a color. This will be invoked for any color anywhere in the DOM, with the exception of IDOMShadingPatternBrushes which require special handling.
- virtual IDOMColorSpacePtr transformColorSpace (IImplementation *genericImplementation, const IDOMColorSpacePtr &colorSpace, const CTransformState &state)

  Callback to process a color space. This will be invoked for any color anywhere in the DOM, with the exception of IDOMShadingPatternBrushes which require special handling.
- virtual IDOMImagePtr transformImage (IImplementation *genericImplementation, const IDOMImagePtr &image, const CTransformState &state)

  Callback to process an image.
- virtual IDOMNodePtr transformNode (IImplementation *genericImplementation, const IDOMNodePtr &node, bool &changed, bool transformChildren, const CTransformState &state)

  Callback to process an IDOMNode (of any type).
- virtual IDOMNodePtr transformFixedPage (IImplementation *genericImplementation, const IDOMFixed↩PagePtr &page, bool &changed, bool transformChildren, const CTransformState &state)

  Callback to process an IDOMFixedPage.
- virtual IDOMNodePtr transformGroup (IImplementation *genericImplementation, const IDOMGroupPtr &group, bool &changed, bool transformChildren, const CTransformState &state)

  Callback to process an IDOMGroup.

- virtual IDOMNodePtr transformCharPathGroup (IImplementation *genericImplementation, const IDOMCharPathGroupPtr &group, bool &changed, bool transformChildren, const CTransformState &state)

  Callback to process an IDOMCharPathGroup.
- virtual IDOMNodePtr transformTransparencyGroup (IImplementation *genericImplementation, const IDOMTransparencyGroupPtr &group, bool &changed, bool transformChildren, const CTransformState &state)

  Callback to process an IDOMTransparencyGroup.
- virtual IDOMNodePtr transformCanvas (IImplementation *genericImplementation, const IDOMCanvasPtr &canvas, bool &changed, bool transformChildren, const CTransformState &state)

  Callback to process an IDOMCanvas.
- virtual IDOMNodePtr transformGlyphs (IImplementation *genericImplementation, const IDOMGlyphsPtr &glyphs, bool &changed, const CTransformState &state)

  Callback to process an IDOMGlyphs node.
- virtual IDOMFontPtr transformFont (IImplementation *genericImplementation, const IDOMFontPtr &font, uint32 &index, const CTransformState &state)

  Callback to process an IDOMFont.
- virtual IDOMNodePtr transformPath (IImplementation *genericImplementation, const IDOMPathNodePtr &path, bool &changed, const CTransformState &state)

  Callback to process an IDOMPathNode.
- virtual IDOMNodePtr transformVisualRoot (IImplementation *genericImplementation, const IDOMVisualRootPtr &root, bool &changed, bool transformChildren, const CTransformState &state)

  Callback to process an IDOMVisualRoot node.
- virtual IDOMNodePtr transformForm (IImplementation *genericImplementation, const IDOMFormPtr &form, bool &changed, bool transformChildren, const CTransformState &state)

  Callback to process an IDOMForm node.
- virtual IDOMNodePtr transformFormInstance (IImplementation *genericImplementation, const IDOMFormInstancePtr &instance, bool &changed, bool transformChildren, const CTransformState &state)

  Callback to process an IDOMFormInstance node.
- virtual IDOMBrushPtr transformBrush (IImplementation *genericImplementation, const IDOMBrushPtr &brush, eBrushUsage usage, const CTransformState &state)

  Callback to process any kind of brush.
- virtual IDOMBrushPtr transformSolidColorBrush (IImplementation *genericImplementation, const IDOMSolidColorBrushPtr &brush, const CTransformState &state)

  Callback to process an IDOMSolidColorBrush.
- virtual IDOMBrushPtr transformGradientBrush (IImplementation *genericImplementation, const IDOMGradientBrushPtr &gradient, const CTransformState &state)

  Callback to process an IDOMGradientBrush.
- virtual IDOMBrushPtr transformLinearGradientBrush (IImplementation *genericImplementation, const IDOMLinearGradientBrushPtr &gradient, const CTransformState &state)

  Callback to process an IDOMLinearGradientBrush.
- virtual IDOMBrushPtr transformRadialGradientBrush (IImplementation *genericImplementation, const IDOMRadialGradientBrushPtr &gradient, const CTransformState &state)

  Callback to process an IDOMRadialGradientBrush.
- virtual IDOMBrushPtr transformVisualBrush (IImplementation *genericImplementation, const IDOMVisualBrushPtr &brush, const CTransformState &state)

  Callback to process an IDOMVisualBrush.
- virtual IDOMBrushPtr transformImageBrush (IImplementation *genericImplementation, const IDOMImageBrushPtr &brush, const CTransformState &state)

  Callback to process an IDOMImageBrush.
- virtual IDOMBrushPtr transformTilingPatternBrush (IImplementation *genericImplementation, const IDOMTilingPatternBrushPtr &brush, const CTransformState &state)

  Callback to process an IDOMTilingPatternBrush.
- virtual IDOMBrushPtr transformShadingPatternBrush (IImplementation *genericImplementation, const IDOMShadingPatternBrushPtr &brush, const CTransformState &state)

Callback to process an IDOMShadingPatternBrush.

- virtual IDOMBrushPtr transformSoftMaskBrush (IImplementation *genericImplementation, const IDOMSoftMaskBrushPtr &brush, const CTransformState &state)

  Callback to process an IDOMSoftMaskBrush.

- virtual IDOMBrushPtr transformMaskedBrush (IImplementation *genericImplementation, const IDOMMaskedBrushPtr &brush, const CTransformState &state)

  Callback to process an IDOMMaskedBrush.

- virtual IDOMBrushPtr transformNullBrush (IImplementation *genericImplementation, const IDOMNullBrushPtr &brush, const CTransformState &state)

  Callback to process an IDOMNullBrush.

7.317.1 Detailed Description

Callback interface that provides methods for actually doing the work. Override the cases for the objects you wish to edit or are otherwise interested in.

Each method is passed in a generic implementation. Here the generic implementation continues the process of the transformation to the lower levels of the tree. For example, the default generic implementation of transformPath() proceeds to recurse, processing the brushes, etc. For these, you can choose to make changes before or after the generic implementation, or simply not use the generic implementation if there is no need to further recurse into the tree.

Implementing transformCharPathGroup() to deal with the IDOMPathNodes associated with IDOMCharPathGroup was required if implemented transformPath(). But it is not required anymore.

The CTransformState transformPriv is available for your use to store additional contextual information in an arbitrary fashion.

7.317.2 Member Function Documentation

transformAnnotation()

```
virtual void JawsMako::ICustomTransform::IImplementation::transformAnnotation (
            IImplementation * genericImplementation,
            const IAnnotationPtr & annotation )  [inline], [virtual]
```

Callback to process an annotation.

Parameters

genericImplementation	The generic implementation that continues processing further down the tree.
annotation	The annotation, which you may modify.

transformAnnotationAppearance()

```
virtual IAnnotationAppearancePtr JawsMako::ICustomTransform::IImplementation::transform↩
AnnotationAppearance (
            IImplementation * genericImplementation,
            const IAnnotationAppearancePtr & appearance,
            const FRect & annotationRect )  [inline], [virtual]
```

Callback to process an annotation appearance.

Parameters

genericImplementation	The generic implementation that continues processing further down the tree.
appearance	The annotation appearance. Do not modify this; instead, if you must modify it, return a cloned version containing your modifications. Do not return NULL.
annotationRect	The owning annotation's rectangle.

Returns

IAnnotationAppearancePtr The result appearance, or the original appearance if no changes were required.

transformBrush()

```
virtual IDOMBrushPtr JawsMako::ICustomTransform::IImplementation::transformBrush (
            IImplementation * genericImplementation,
            const IDOMBrushPtr & brush,
            eBrushUsage usage,
            const CTransformState & state )  [inline], [virtual]
```

Callback to process any kind of brush.

Parameters

genericImplementation	The generic implementation that continues processing further down the tree.
brush	The brush. Do not edit this brush in place, instead create a copy or a new brush instead.
state	The current state *outside* the brush.

Returns

IDOMBrushPtr The resulting brush, or NULL if the brush should be dropped.

transformCanvas()

```
virtual IDOMNodePtr JawsMako::ICustomTransform::IImplementation::transformCanvas (
            IImplementation * genericImplementation,
            const IDOMCanvasPtr & canvas,
            bool & changed,
            bool transformChildren,
            const CTransformState & state )  [inline], [virtual]
```

Callback to process an IDOMCanvas.

Parameters

genericImplementation	The generic implementation that continues processing further down the tree.
canvas	The canvas. You are free to modify this node or its children in place, but do not modify any potentially shared objects, such as brushes. Instead, those objects should be cloned first. You may return NULL, or a node of a completely different type.

Parameters

changed	You must set this to true if any changes have been made to the node, and set it to false if you have not.
transformChildren	True if the children of the node should also be transformed
state	The current state *inside* the node, with the node's attributes taken into account (such as its renderTransform)

Returns

IDOMNodePtr The resulting node, or NULL if the node should be dropped.

transformCharPathGroup()

```
virtual IDOMNodePtr JawsMako::ICustomTransform::IImplementation::transformCharPathGroup (
            IImplementation * genericImplementation,
            const IDOMCharPathGroupPtr & group,
            bool & changed,
            bool transformChildren,
            const CTransformState & state )  [inline], [virtual]
```

Callback to process an IDOMCharPathGroup.

Parameters

genericImplementation	The generic implementation that continues processing further down the tree.
group	The group. You are free to modify this node or its children in place, but do not modify any potentially shared objects, such as brushes. Instead, those objects should be cloned first. You may return NULL, or a node of a completely different type.
changed	You must set this to true if any changes have been made to the node, and set it to false if you have not.
transformChildren	True if the children of the node should also be transformed
state	The current state *inside* the node, with the node's attributes taken into account (such as its renderTransform)

Returns

IDOMNodePtr The resulting node, or NULL if the node should be dropped.

transformColor()

```
virtual IDOMColorPtr JawsMako::ICustomTransform::IImplementation::transformColor (
            IImplementation * genericImplementation,
            const IDOMColorPtr & color,
            const CTransformState & state )  [inline], [virtual]
```

Callback to process a color. This will be invoked for any color anywhere in the DOM, with the exception of IDOMShadingPatternBrushes which require special handling.

Parameters

genericImplementation	The generic implementation that continues processing further down the tree.
color	The color. Do not modify this color; instead create a clone and apply your modifications to that.
state	The current state when the color was encountered.

Returns

IDOMColorPtr The result color, or the original color if no changes were required.

transformColorSpace()

```
virtual IDOMColorSpacePtr JawsMako::ICustomTransform::IImplementation::transformColorSpace (
            IImplementation * genericImplementation,
            const IDOMColorSpacePtr & colorSpace,
            const CTransformState & state )  [inline], [virtual]
```

Callback to process a color space. This will be invoked for any color anywhere in the DOM, with the exception of IDOMShadingPatternBrushes which require special handling.

Parameters

genericImplementation	The generic implementation that continues processing further down the tree.
colorSpace	The color space. Do not modify this color space; instead create a clone and apply your modifications to that. Any color space you return must have the same number of components as colorSpace
state	The current state when the color space was encountered.

Returns

IDOMColorPtr The result color space, or the original color space if no changes were required.

transformFixedPage()

```
virtual IDOMNodePtr JawsMako::ICustomTransform::IImplementation::transformFixedPage (
            IImplementation * genericImplementation,
            const IDOMFixedPagePtr & page,
            bool & changed,
            bool transformChildren,
            const CTransformState & state )  [inline], [virtual]
```

Callback to process an IDOMFixedPage.

Parameters

genericImplementation	The generic implementation that continues processing further down the tree.
page	The fixed page. You are free to modify this node or its children in place, but do not modify any potentially shared objects, such as brushes. Instead, those objects should be cloned first.

Parameters

changed	You must set this to true if any changes have been made to the node, and set it to false if you have not.
transformChildren	True if the children of the node should also be transformed
state	The current state when the node was encountered.

Returns

IDOMNodePtr The resulting node.

transformFont()

```
virtual IDOMFontPtr JawsMako::ICustomTransform::IImplementation::transformFont (
            IImplementation * genericImplementation,
            const IDOMFontPtr & font,
            uint32 & index,
            const CTransformState & state )  [inline], [virtual]
```

Callback to process an IDOMFont.

Parameters

genericImplementation	The generic implementation that continues processing further down the tree.
font	The font. You must not modify this font. Instead, create a new font or work on a clone.
index	The font index of the font being edited. On return, this must be the index of the resulting font.
state	The current state *inside* the node, with the node's attributes taken into account (such as its renderTransform)

Returns

IDOMFontPtr The resulting font, or the original font if no changes were required.

transformForm()

```
virtual IDOMNodePtr JawsMako::ICustomTransform::IImplementation::transformForm (
            IImplementation * genericImplementation,
            const IDOMFormPtr & form,
            bool & changed,
            bool transformChildren,
            const CTransformState & state )  [inline], [virtual]
```

Callback to process an IDOMForm node.

Parameters

genericImplementation	The generic implementation that continues processing further down the tree.
form	The form. You must not edit this form directly. Instead, make a deep clone first.

Parameters

changed	You must set this to true if any changes have been made to the node, and set it to false if you have not.
transformChildren	True if the children of the node should also be transformed
state	The current state *inside* the node, with the node's attributes taken into account (such as its renderTransform)

Returns

IDOMNodePtr The resulting node, or NULL if the node should be dropped.

transformFormInstance()

```
virtual IDOMNodePtr JawsMako::ICustomTransform::IImplementation::transformFormInstance (
            IImplementation * genericImplementation,
            const IDOMFormInstancePtr & instance,
            bool & changed,
            bool transformChildren,
            const CTransformState & state )  [inline], [virtual]
```

Callback to process an IDOMFormInstance node.

Parameters

genericImplementation	The generic implementation that continues processing further down the tree.
instance	The instance. You are free to modify this node in place, but do not modify the contents of the form unless you first create a deep clone of it.
changed	You must set this to true if any changes have been made to the node, and set it to false if you have not.
transformChildren	True if the children of the node should also be transformed
state	The current state *inside* the node, with the node's attributes taken into account (such as its renderTransform)

Returns

IDOMNodePtr The resulting node, or NULL if the node should be dropped.

transformGlyphs()

```
virtual IDOMNodePtr JawsMako::ICustomTransform::IImplementation::transformGlyphs (
            IImplementation * genericImplementation,
            const IDOMGlyphsPtr & glyphs,
            bool & changed,
            const CTransformState & state )  [inline], [virtual]
```

Callback to process an IDOMGlyphs node.

Parameters

genericImplementation	The generic implementation that continues processing further down the tree.
glyphs	The glyphs. You are free to modify this node or its children in place, but do not modify any potentially shared objects, such as brushes. Instead, those objects should be cloned first. You may return NULL, or a node of a completely different type.
changed	You must set this to true if any changes have been made to the node, and set it to false if you have not.
state	The current state *inside* the node, with the node's attributes taken into account (such as its renderTransform)

Returns

IDOMNodePtr The resulting node, or NULL if the node should be dropped.

transformGradientBrush()

```
virtual IDOMBrushPtr JawsMako::ICustomTransform::IImplementation::transformGradientBrush (
            IImplementation * genericImplementation,
            const IDOMGradientBrushPtr & gradient,
            const CTransformState & state )  [inline], [virtual]
```

Callback to process an IDOMGradientBrush.

Parameters

genericImplementation	The generic implementation that continues processing further down the tree.
gradient	The brush. Do not edit this brush in place, instead create a copy or a new brush instead.
state	The current state *inside* the brush, including any renderTransform applied by the brush.

Returns

IDOMBrushPtr The resulting brush (which need not be the same type), or NULL if the brush should be dropped.

transformGroup()

```
virtual IDOMNodePtr JawsMako::ICustomTransform::IImplementation::transformGroup (
            IImplementation * genericImplementation,
            const IDOMGroupPtr & group,
            bool & changed,
            bool transformChildren,
            const CTransformState & state )  [inline], [virtual]
```

Callback to process an IDOMGroup.

Parameters

genericImplementation	The generic implementation that continues processing further down the tree.

Parameters

group	The group. You are free to modify this node or its children in place, but do not modify any potentially shared objects, such as brushes. Instead, those objects should be cloned first. You may return NULL, or a node of a completely different type.
changed	You must set this to true if any changes have been made to the node, and set it to false if you have not.
transformChildren	True if the children of the node should also be transformed
state	The current state *inside* the node, with the node's attributes taken into account (such as its renderTransform)

Returns

IDOMNodePtr The resulting node, or NULL if the node should be dropped.

transformImage()

```
virtual IDOMImagePtr JawsMako::ICustomTransform::IImplementation::transformImage (
            IImplementation * genericImplementation,
            const IDOMImagePtr & image,
            const CTransformState & state )  [inline], [virtual]
```

Callback to process an image.

Parameters

genericImplementation	The generic implementation that continues processing further down the tree.
image	The image. Do not modify this image; instead create a clone and apply your modifications to that. You may return a NULL image if you wish to drop the image entirely
state	The current state when the image was encountered.

Returns

IDOMImagePtr The result image, the original image if no changes were required, or NULL if the image should be dropped entirely.

transformImageBrush()

```
virtual IDOMBrushPtr JawsMako::ICustomTransform::IImplementation::transformImageBrush (
            IImplementation * genericImplementation,
            const IDOMImageBrushPtr & brush,
            const CTransformState & state )  [inline], [virtual]
```

Callback to process an IDOMImageBrush.

Parameters

genericImplementation	The generic implementation that continues processing further down the tree.
brush	The brush. Do not edit this brush in place, instead create a copy or a new brush instead.
state	The current state *inside* the brush, including any renderTransform applied by the brush.

Returns

IDOMBrushPtr The resulting brush (which need not be the same type), or NULL if the brush should be dropped.

transformLinearGradientBrush()

```
virtual IDOMBrushPtr JawsMako::ICustomTransform::IImplementation::transformLinearGradientBrush
(
            IImplementation * genericImplementation,
            const IDOMLinearGradientBrushPtr & gradient,
            const CTransformState & state )  [inline], [virtual]
```

Callback to process an IDOMLinearGradientBrush.

Parameters

genericImplementation	The generic implementation that continues processing further down the tree.
gradient	The brush. Do not edit this brush in place, instead create a copy or a new brush instead.
state	The current state *inside* the brush, including any renderTransform applied by the brush.

Returns

IDOMBrushPtr The resulting brush (which need not be the same type), or NULL if the brush should be dropped.

transformMaskedBrush()

```
virtual IDOMBrushPtr JawsMako::ICustomTransform::IImplementation::transformMaskedBrush (
            IImplementation * genericImplementation,
            const IDOMMaskedBrushPtr & brush,
            const CTransformState & state )  [inline], [virtual]
```

Callback to process an IDOMMaskedBrush.

Parameters

genericImplementation	The generic implementation that continues processing further down the tree.
brush	The brush. Do not edit this brush in place, instead create a copy or a new brush instead.
state	The current state *outside* the brush. Note that this differs from other brush types. The reason for this is that the sub brush of an IDOMMaskedBrush is not affected by the renderTransform of the IDOMMaskedBrush itself. As such any render transforms must be handled by the implementation of this function.

Returns

IDOMBrushPtr The resulting brush (which need not be the same type), or NULL if the brush should be dropped.

transformNode()

```
virtual IDOMNodePtr JawsMako::ICustomTransform::IImplementation::transformNode (
            IImplementation * genericImplementation,
            const IDOMNodePtr & node,
            bool & changed,
            bool transformChildren,
            const CTransformState & state )  [inline], [virtual]
```

Callback to process an IDOMNode (of any type).

Parameters

genericImplementation	The generic implementation that continues processing further down the tree.
node	The node. You are free to modify this node or its children in place, but do not modify any potentially shared objects, such as brushes. Instead, those objects should be cloned first.
changed	You must set this to true if any changes have been made to the node, and set it to false if you have not.
transformChildren	True if the children of the node should also be transformed
state	The current state when the node was encountered.

Returns

IDOMNodePtr The result. NULL may be returned if the node should be dropped entirely. A completely different node type may also be returned if required.

transformNullBrush()

```
virtual IDOMBrushPtr JawsMako::ICustomTransform::IImplementation::transformNullBrush (
            IImplementation * genericImplementation,
            const IDOMNullBrushPtr & brush,
            const CTransformState & state )  [inline], [virtual]
```

Callback to process an IDOMNullBrush.

Parameters

genericImplementation	The generic implementation that continues processing further down the tree.
brush	The brush. Do not edit this brush in place, instead create a copy or a new brush instead.
state	The current state *inside* the brush, including any renderTransform applied by the brush.

Returns

IDOMBrushPtr The resulting brush (which need not be the same type), or NULL if the brush should be dropped.

transformPath()

```
virtual IDOMNodePtr JawsMako::ICustomTransform::IImplementation::transformPath (
            IImplementation * genericImplementation,
```

```
            const IDOMPathNodePtr & path,
            bool & changed,
            const CTransformState & state )  [inline], [virtual]
```

Callback to process an IDOMPathNode.

Parameters

genericImplementation	The generic implementation that continues processing further down the tree.
path	The path. You are free to modify this node or its children in place, but do not modify any potentially shared objects, such as brushes. Instead, those objects should be cloned first. You may return NULL, or a node of a completely different type.
changed	You must set this to true if any changes have been made to the node, and set it to false if you have not.
state	The current state *inside* the node, with the node's attributes taken into account (such as its renderTransform)

Returns

IDOMNodePtr The resulting node, or NULL if the node should be dropped.

transformRadialGradientBrush()

```
virtual IDOMBrushPtr JawsMako::ICustomTransform::IImplementation::transformRadialGradientBrush
(
            IImplementation * genericImplementation,
            const IDOMRadialGradientBrushPtr & gradient,
            const CTransformState & state )  [inline], [virtual]
```

Callback to process an IDOMRadialGradientBrush.

Parameters

genericImplementation	The generic implementation that continues processing further down the tree.
gradient	The brush. Do not edit this brush in place, instead create a copy or a new brush instead.
state	The current state *inside* the brush, including any renderTransform applied by the brush.

Returns

IDOMBrushPtr The resulting brush (which need not be the same type), or NULL if the brush should be dropped.

transformShadingPatternBrush()

```
virtual IDOMBrushPtr JawsMako::ICustomTransform::IImplementation::transformShadingPatternBrush
(
            IImplementation * genericImplementation,
            const IDOMShadingPatternBrushPtr & brush,
            const CTransformState & state )  [inline], [virtual]
```

Callback to process an IDOMShadingPatternBrush.

Parameters

genericImplementation	The generic implementation that continues processing further down the tree.
brush	The brush. Do not edit this brush in place, instead create a copy or a new brush instead.
state	The current state *inside* the brush, including any renderTransform applied by the brush.

Returns

IDOMBrushPtr The resulting brush (which need not be the same type), or NULL if the brush should be dropped.

transformSoftMaskBrush()

```
virtual IDOMBrushPtr JawsMako::ICustomTransform::IImplementation::transformSoftMaskBrush (
            IImplementation * genericImplementation,
            const IDOMSoftMaskBrushPtr & brush,
            const CTransformState & state )  [inline], [virtual]
```

Callback to process an IDOMSoftMaskBrush.

Parameters

genericImplementation	The generic implementation that continues processing further down the tree.
brush	The brush. Do not edit this brush in place, instead create a copy or a new brush instead.
state	The current state *inside* the brush, including any renderTransform applied by the brush.

Returns

IDOMBrushPtr The resulting brush (which need not be the same type), or NULL if the brush should be dropped.

transformSolidColorBrush()

```
virtual IDOMBrushPtr JawsMako::ICustomTransform::IImplementation::transformSolidColorBrush (
            IImplementation * genericImplementation,
            const IDOMSolidColorBrushPtr & brush,
            const CTransformState & state )  [inline], [virtual]
```

Callback to process an IDOMSolidColorBrush.

Parameters

genericImplementation	The generic implementation that continues processing further down the tree.
brush	The brush. Do not edit this brush in place, instead create a copy or a new brush instead.
state	The current state *inside* the brush.

Returns

IDOMBrushPtr The resulting brush (which need not be the same type), or NULL if the brush should be dropped.

transformTilingPatternBrush()

```
virtual IDOMBrushPtr JawsMako::ICustomTransform::IImplementation::transformTilingPatternBrush
(
            IImplementation * genericImplementation,
            const IDOMTilingPatternBrushPtr & brush,
            const CTransformState & state )  [inline], [virtual]
```

Callback to process an IDOMTilingPatternBrush.

Parameters

genericImplementation	The generic implementation that continues processing further down the tree.
brush	The brush. Do not edit this brush in place, instead create a copy or a new brush instead.
state	The current state *inside* the brush, including any renderTransform applied by the brush.

Returns

IDOMBrushPtr The resulting brush (which need not be the same type), or NULL if the brush should be dropped.

transformTransparencyGroup()

```
virtual IDOMNodePtr JawsMako::ICustomTransform::IImplementation::transformTransparencyGroup (
            IImplementation * genericImplementation,
            const IDOMTransparencyGroupPtr & group,
            bool & changed,
            bool transformChildren,
            const CTransformState & state )  [inline], [virtual]
```

Callback to process an IDOMTransparencyGroup.

Parameters

genericImplementation	The generic implementation that continues processing further down the tree.
group	The group. You are free to modify this node or its children in place, but do not modify any potentially shared objects, such as brushes. Instead, those objects should be cloned first. You may return NULL, or a node of a completely different type.
changed	You must set this to true if any changes have been made to the node, and set it to false if you have not.
transformChildren	True if the children of the node should also be transformed
state	The current state *inside* the node, with the node's attributes taken into account (such as its renderTransform)

Returns

IDOMNodePtr The resulting node, or NULL if the node should be dropped.

transformVisualBrush()

```
virtual IDOMBrushPtr JawsMako::ICustomTransform::IImplementation::transformVisualBrush (
            IImplementation * genericImplementation,
            const IDOMVisualBrushPtr & brush,
            const CTransformState & state )  [inline], [virtual]
```

Callback to process an IDOMVisualBrush.

Parameters

genericImplementation	The generic implementation that continues processing further down the tree.
brush	The brush. Do not edit this brush in place, instead create a copy or a new brush instead.
state	The current state *inside* the brush, including any renderTransform applied by the brush.

Returns

IDOMBrushPtr The resulting brush (which need not be the same type), or NULL if the brush should be dropped.

transformVisualRoot()

```
virtual IDOMNodePtr JawsMako::ICustomTransform::IImplementation::transformVisualRoot (
            IImplementation * genericImplementation,
            const IDOMVisualRootPtr & root,
            bool & changed,
            bool transformChildren,
            const CTransformState & state )  [inline], [virtual]
```

Callback to process an IDOMVisualRoot node.

Parameters

genericImplementation	The generic implementation that continues processing further down the tree.
root	The visual root. You are free to modify this node or its children in place, but do not modify any potentially shared objects, such as brushes. Instead, those objects should be cloned first. You may return NULL, or a node of a completely different type.
changed	You must set this to true if any changes have been made to the node, and set it to false if you have not.
transformChildren	True if the children of the node should also be transformed
state	The current state *inside* the node, with the node's attributes taken into account (such as its renderTransform)

Returns

IDOMNodePtr The resulting node, or NULL if the node should be dropped.

The documentation for this class was generated from the following file:

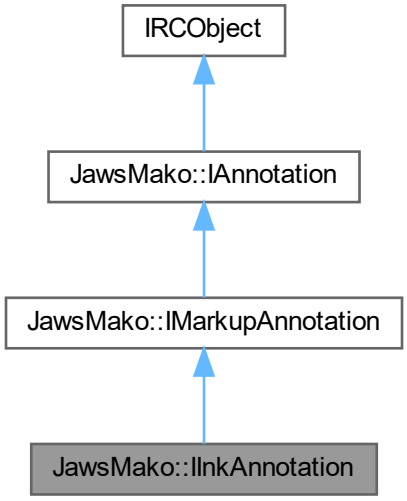
- customtransform.h

7.318 JawsMako::IInkAnnotation Class Reference

A generic interface class for a ink annotation It is intended that future releases of JawsMako will extend this interface.

`#include <interactive.h>`

Inheritance diagram for JawsMako::IInkAnnotation:



Public Member Functions

- virtual CFPointVectVect getInkList () const =0

  Get the annotation's ink list. All coordinates are relative to the annotation rectangle.
- virtual void setInkList (const CFPointVectVect &inkList)=0

  Get the annotation's ink list. All coordinates are relative to the annotation rectangle.

Public Member Functions inherited from JawsMako::IMarkupAnnotation

- virtual U8String getAuthor () const =0

  Get the Author of the markup annotation. This is the "T" entry in the annotation, which by convention is the author.
- virtual void setAuthor (const U8String &author)=0

  Set the Author of the markup annotation. This is the "T" entry in the annotation, which by convention is the author.
- virtual IEDLTimePtr getCreationTime () const =0

  Get the creation date and time of the annotation, if present.
- virtual void setCreationTime (const IEDLTimePtr &creationTime)=0

Set the creation date and time of the annotation.

- virtual float getOpacity () const =0

  Get the opacity of the markup annotation.
- virtual void setOpacity (float opacity)=0

  Set the opacity of the markup annotation.
- virtual IAnnotationReferencePtr getPopupReference () const =0

  Get a reference to the popup, if present.
- virtual void setPopup (const IAnnotationReferencePtr &popupReference)=0

  Set or clear the popup, by reference.
- virtual void setPopup (const IPopupAnnotationPtr &popup)=0

  Set a reference to a popup, if present.
- virtual IAnnotationAppearancePtr createNormalAppearance () const =0

  Create a basic appearance given the current annotation's parameters. This can then be installed by using IAnnotation::addAppearance().

Public Member Functions inherited from JawsMako::IAnnotation

- virtual eAnnotationType getType () const =0

  Get the type of the annotation.
- virtual const FRect & getRect () const =0

  Get the rect in which the appearances should be displayed.
- virtual void setRect (const FRect &rect)=0

  Set the rect in which the appearances should be displayed.
- virtual U8String getContents () const =0

  Get the Contents entry in UTF-8.
- virtual void setContents (const U8String &contents)=0

  Set the Contents entry in UTF-8.
- virtual IDOMColorPtr getColor () const =0

  Get the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.
- virtual void setColor (const IDOMColorPtr &color)=0

  Set the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.
- virtual IEDLTimePtr getModificationTime () const =0

  Get the Modification date and time of the annotation, if present.
- virtual void setModificationTime (const IEDLTimePtr &modificationTime)=0

  Set the Modification date and time of the annotation.
- virtual CAnnotationBorder getBorder () const =0

  Get the annotation's border. See CAnnotationBorder for details.
- virtual void setBorder (const CAnnotationBorder &border)=0

  Set the annotation's border.
- virtual uint32 getFlags () const =0

  Get the annotation flags. To interpret these flags please see section 8.4.2 "Annotation Flags" in the PDF 1.7 specification.
- virtual void setFlags (uint32 flags)=0

  Set the annotation flags. Please see section 8.4.2 "Annotation Flags" for details about these flags.
- virtual void rotate (uint16 rotate, double pageWidth, double pageHeight)=0

  Rotate the annotation clockwise as if the page was rotated by the same amount.
- virtual CAnnotationAppearanceVect getAppearances () const =0

  Return all the annotation appearances in a vector.
- virtual void **removeAppearances** ()=0

  Remove all annotation appearances.
- virtual U8String getState () const =0

Get the current annotation state.

- virtual void setState (const U8String &state)=0

  Set the current annotation state.

- virtual IAnnotationAppearancePtr getAppearance (eAppearanceUsage usage, const U8String &state=U8String()) const =0

  Fetch the annotation appearance that should be used for the given annotation usage and state according to the following rules:

- virtual void addAppearance (const IAnnotationAppearancePtr &appearance)=0

  Add or replace an appearance.

- virtual bool hasNormalAppearance () const =0

  Does the annotation have a normal appearance? Convenience utility function.

- virtual IAnnotationPtr clone () const =0

  Clone the annotation. This is a deep clone. The annotation reference will remain the same.

- virtual bool matchesReference (const IAnnotationReferencePtr &reference) const =0

  Does this annotation match the given IAnnotationReference?

- virtual IAnnotationReferencePtr getReference () const =0

  Get a reference that can be used to refer to this annotation.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.

- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.

- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IInkAnnotationPtr create (const IJawsMakoPtr &jawsMako, const FRect &rect, const IDOMColorPtr &color=IDOMColorPtr(), const CFPointVectVect &inkList=CFPointVectVect())

  Create an ink annotation.

Additional Inherited Members

Public Types inherited from JawsMako::IAnnotation

- enum eAnnotationType {
  eAT3D , eATCaret , eATCircle , eATFileAttachment ,
  eATFreeText , eATHighlight , eATInk , eATLine ,
  eATLink , eATMovie , eATPolygon , eATPolyLine ,
  eATPopup , eATPrinterMark , eATProjection , eATRedact ,
  eATRichMedia , eATScreen , eATSound , eATSquare ,
  eATSquiggly , eATStamp , eATStrikeOut , eATText ,
  eATTrapNet , eATUnderline , eATWatermark , eATWidget ,
  eATOther }

  Types of annotations, listed with the earliest version of PDF that supported them.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()
  Virtual destructor.

7.318.1 Detailed Description

A generic interface class for a ink annotation It is intended that future releases of JawsMako will extend this interface.

7.318.2 Member Function Documentation

create()

```
static JAWSMAKO_API IInkAnnotationPtr JawsMako::IInkAnnotation::create (
            const IJawsMakoPtr & jawsMako,
            const FRect & rect,
            const IDOMColorPtr & color = IDOMColorPtr(),
            const CFPointVectVect & inkList = CFPointVectVect() )  [static]
```

Create an ink annotation.

Parameters

jawsMako	The JawsMako instance
rect	The annotations bounds. Must not be empty.
color	Optional; The desired annotation color. If NULL, RGB yellow will be chosen. If provided, the color space must be DeviceRGB, DeviceCMYK or DeviceGray
inkList	Optional; the ink list to use

Returns

IInkAnnotationPtr A smart pointer to the new ink annotation

getInkList()

```
virtual CFPointVectVect JawsMako::IInkAnnotation::getInkList ( ) const  [pure virtual]
```

Get the annotation's ink list. All coordinates are relative to the annotation rectangle.

Returns

CFPointVectVect The ink list (a collection of quadpoint data)

setInkList()

```
virtual void JawsMako::IInkAnnotation::setInkList (
            const CFPointVectVect & inkList )  [pure virtual]
```

Get the annotation's ink list. All coordinates are relative to the annotation rectangle.

Parameters

inkList	The ink list (a collection of quadpoint data)

The documentation for this class was generated from the following file:

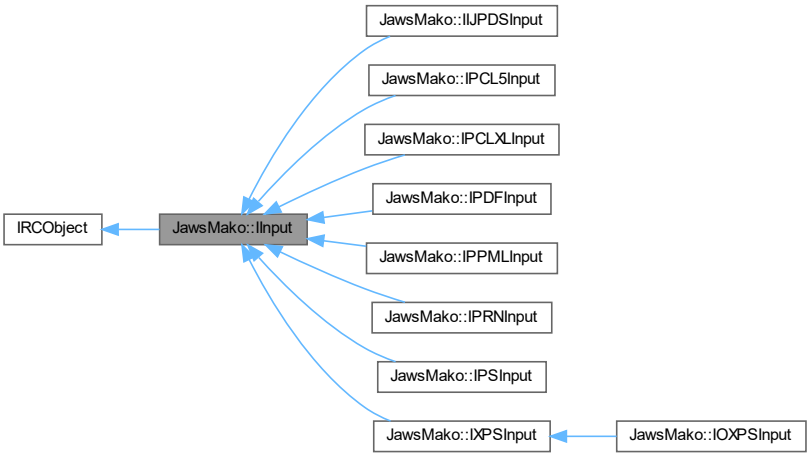
- interactive.h

7.319 JawsMako::IInput Class Reference

Abstract input source that can open files from disk or a stream and create an IDocumentAssembly for the contents.

`#include <jawsmako.h>`

Inheritance diagram for JawsMako::IInput:



Public Member Functions

- virtual IDocumentAssemblyPtr open (const U8String &pathToFile)=0

  Open a file on disk, returning the IDocumentAssembly representing the contents.
- virtual IDocumentAssemblyPtr open (const String &pathToFile)=0

  Open a file on disk, returning the IDocumentAssembly representing the contents. Takes a wide character string.
- virtual IDocumentAssemblyPtr open (const IInputStreamPtr &inputStream)=0

  Open a stream, returning the IDocumentAssembly representing the contents.
- virtual void setSequentialMode (bool sequential)=0

  Set/unset sequential mode on this input.
- virtual void setParameter (const U8String &param, const U8String &value)=0

  Apply a key value pair output parameter with a string value. The parameter name is case insensitive. Please refer to the supplied documentation for the details of the available parameters and their ranges.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IInputPtr create (const IJawsMakoPtr &jawsMako, eFileFormat format, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create an input for reading source documents in the given format. The following formats are currently supported:

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.319.1 Detailed Description

Abstract input source that can open files from disk or a stream and create an IDocumentAssembly for the contents.

7.319.2 Member Function Documentation

create()

```
static JAWSMAKO_API IInputPtr JawsMako::IInput::create (
            const IJawsMakoPtr & jawsMako,
            eFileFormat format,
            const IProgressMonitorPtr & progressMonitor = IProgressMonitorPtr() )  [static]
```

Create an input for reading source documents in the given format. The following formats are currently supported:

- PDF
- XPS
- OpenXPS
- PCL5
- PCL/XL
- PostScript
- Encapsulated PostScript
- IJPDS
- PPML
- PRN

Parameters

jawsMako	A smart pointer to an IJawsMako object.
progressMonitor	A smart pointer to an IProgressMonitor object which can be NULL of no object was passed.

Returns

IInputPtr the input

open() [1/3]

```
virtual IDocumentAssemblyPtr JawsMako::IInput::open (
            const IInputStreamPtr & inputStream )  [pure virtual]
```

Open a stream, returning the IDocumentAssembly representing the contents.

Returns

IDocumentAssemblyPtr the document assembly.

open() [2/3]

```
virtual IDocumentAssemblyPtr JawsMako::IInput::open (
            const String & pathToFile )  [pure virtual]
```

Open a file on disk, returning the IDocumentAssembly representing the contents. Takes a wide character string.

Returns

IDocumentAssemblyPtr the document assembly.

open() [3/3]

```
virtual IDocumentAssemblyPtr JawsMako::IInput::open (
            const U8String & pathToFile )  [pure virtual]
```

Open a file on disk, returning the IDocumentAssembly representing the contents.

Returns

IDocumentAssemblyPtr the document assembly.

setParameter()

```
virtual void JawsMako::IInput::setParameter (
            const U8String & param,
            const U8String & value )  [pure virtual]
```

Apply a key value pair output parameter with a string value. The parameter name is case insensitive. Please refer to the supplied documentation for the details of the available parameters and their ranges.

Parameters

param	Key
value	Value

setSequentialMode()

```
virtual void JawsMako::IInput::setSequentialMode (
            bool sequential )  [pure virtual]
```

Set/unset sequential mode on this input.

The Mako APIs allow random access to pages for all input types. However some input formats, particularly PCL/5 do not lend themselves to efficient random access. An input may need to buffer up page information as it passes it in order to allow pages to be re-fetched or accessed in random order.

This API allows the user to hint to the input that pages will be accessed in sequential order and will not be re-requested. This may allow the input to use more memory efficient methods.

Should this mode be enabled and a page requested out of order, or re-requested, an exception may result. Also with this mode, do not used IDocument::getNumPages() or IDocumentAssembly::getNumDocuments(). Instead use an idiom, similar to the following, to advance through the documents and pages:

```
for (uint32 docNum = 0; ; docNum++)
{
    IDocumentPtr doc;

    try
    {
        doc = assembly->getDocument(docNum);
    }
    catch (IError &e)
    {
        if (e.getErrorCode() == JM_ERR_DOCUMENT_NOT_FOUND)
        {
            // No more documents
            break;
        }
        // Some other error
        throw;
    }

    for (uint32 pageNum = 0; ; pageNum++)
    {
        IPagePtr page;
        try
        {
            page = doc->getPage(pageNum);
        }
        catch (IError &e)
        {
            if (e.getErrorCode() == JM_ERR_PAGE_NOT_FOUND)
            {
                // No more pages
                break;
            }
            // Some other failure. Propagate.
            throw;
        }
    }
}
```

Note

Currently this mode is only implemented for PCL/5.

The default is false.

Parameters

sequential	Whether or not to use sequential mode.

The documentation for this class was generated from the following file:

- jawsmako.h

7.320 IInputEnum< typename T > Class Reference

Iterator template class to allow iteration over a collection of instances of type <T>

`#include <iedlenum.h>`

7.320.1 Detailed Description

Iterator template class to allow iteration over a collection of instances of type <T>

The documentation for this class was generated from the following file:

- iedlenum.h

7.321 IInputEnumRC< typename T > Class Reference

Reference-counted iterator template class to allow iteration over a collection of instances of type <T>

`#include <iedlenum.h>`

7.321.1 Detailed Description

Reference-counted iterator template class to allow iteration over a collection of instances of type <T>

The documentation for this class was generated from the following file:

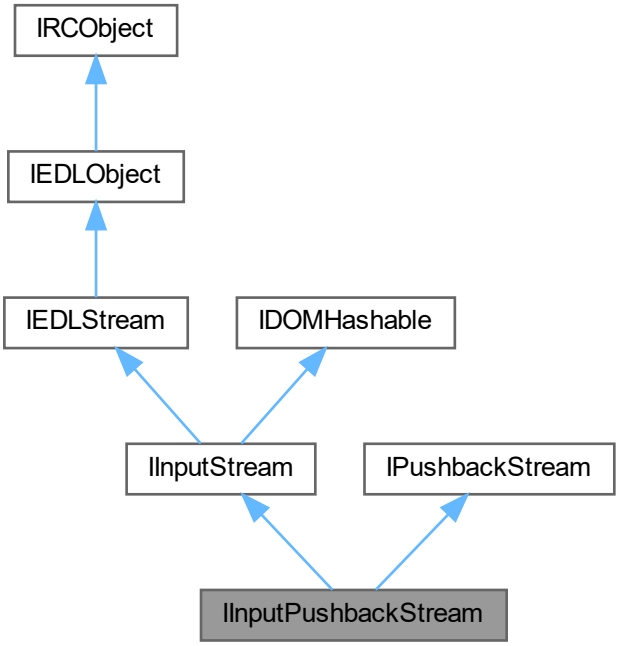
- iedlenum.h

7.322 IInputPushbackStream Class Reference

Input Stream with pushback support.

`#include <edlstream.h>`

Inheritance diagram for IInputPushbackStream:



Additional Inherited Members

Public Member Functions inherited from IInputStream

- virtual int32 read (void *buffer, int32 count)=0

  Read specified number of bytes from a stream into buffer.
- virtual int8 read ()=0

  Read single byte from a stream.
- virtual bool eof () const =0

  Determine if the stream has exhausted.
- virtual int64 skip (int64 count)

  Skip a specified number of bytes.
- virtual bool getSourceFilePath (EDLSysString &sourcePath)=0

  If available, find the file path of the file that this stream references, or for streams that take their input from other streams, the ultimate source stream. If at all possible, the source path will be canonicalised, but this is not guaranteed. If a canonical name is absolutely required, check getCanonicalSourceFilePath().
- virtual bool getCanonicalSourceFilePath (EDLSysString &sourceCanonicalPath)=0

If available, find the canonical file path of the file that this stream references, or for streams that take their input from other streams, the ultimate source stream. For canonical names to be discovered on all platforms, the underlying file must exist. If the canonical file path cannot be determined, false will be returned. In which case, you may wish to try getSourceFilePath() above.

- virtual bool completeRead (void *buffer, int32 count)

  Perform a complete read.
- virtual void completeReadE (void *buffer, int32 count)

  As completeRead(), but throws an exception if the operation fails.
- virtual bool hash (uint64 &hash)

  Obtain a 64-bit hash of the stream. Please note that this requires reading the stream and is therefore not thread safe. If thread safety is desired, make a clone of the stream first.

Public Member Functions inherited from IEDLStream

- virtual bool isValid () const =0

  Determine stream validity.
- virtual bool open ()

  Opens the stream.
- virtual void **openE** ()=0

  As per open(), but will throw an exception on failure (IEDLError) that for some stream types may contain additional failure information.
- virtual void **close** ()=0

  Closes the stream.
- virtual int64 getPos ()=0

  Get current stream position.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Public Member Functions inherited from IPushbackStream

- virtual ~**IPushbackStream** ()

  Virtual destructor.
- virtual bool pushBack (uint8 byte)=0

  Push back a byte.
- virtual bool pushBack (const void *buffer, int32 count)=0

  Push back from a buffer.

Static Public Member Functions inherited from IInputStream

- static EDL_API IRAInputStreamPtr createFromFile (IEDLClassFactory *pFactory, const EDLSysString &path)

  Creation function for an IInputStream for a file on disk. Throws an IEDLError exception on failure.
- static EDL_API IRAInputStreamPtr createFromFile (IEDLClassFactory *pFactory, const EDLString &path)

  Creation function for an IInputStream for a file on disk. Throws an IEDLError exception on failure.
- static EDL_API IRAInputStreamPtr createFromFileShared (IEDLClassFactory *pFactory, const EDLSys↩String &path)

  Creation function for an IInputStream for a file on disk. Similar to createFromFile, but if this file is cloned, only one file handle will be open at a time. This is slower, but is useful if many users are likely to want to have this file open at the same file. On some systems, having too many files open may result in errors. Performance should be acceptable providing there is not a great deal of contention and simultaneous access.
- static EDL_API IRAInputStreamPtr createFromFileShared (IEDLClassFactory *pFactory, const EDLString &path)

  Creation function for an IInputStream for a file on disk. Similar to createFromFile, but if this stream is cloned, only one file handle will be open at a time. This is slower, but is useful if many users are likely to want to have this file open at the same file. On some systems, having too many files open may result in errors. Performance should be acceptable providing there is not a great deal of contention and simultaneous access.
- static EDL_API IRAInputStreamPtr createSharedFromStream (IEDLClassFactory *pFactory, const IRAInputStreamPtr &stream)

  Creation function for a shared stream overlaying an existing stream. If this stream is cloned, the underlying file will not be cloned. This is useful if many users are likely to want to have this file open at the same time. On some systems, having too many files open may result in errors. Performance should be acceptable providing there is not a great deal of contention and simultaneous access. Only possible for random access streams. If the stream is already shared, the stream will be returned as is.
- static EDL_API IRAInputStreamPtr createFromMemory (IEDLClassFactory *pFactory, const void *mem, uint32 length, bool copy=false, bool free=true)

  Creation function for an IInputStream for data in memory. Throws an IEDLError exception on failure.
- static EDL_API IInputStreamPtr createFromUserReadFunc (IEDLClassFactory *pFactory, UserStream↩ReadFunc readFunc, void *priv)

  Creation function for an IInputStream from a user function that provides data. Throws an IEDLError exception on failure.
- static EDL_API IRAInputStreamPtr createFromRAUserFunc (IEDLClassFactory *pFactory, int64 length, UserRAReadFunc readFunc, void *priv)

  Creation function for an IRAInputStream from a user function that provides random access. Throws an IEDLError exception on failure.
- static EDL_API IRAInputStreamPtr createRandomAccessFromNonRandomAccess (IEDLClassFactory *p↩Factory, const IInputStreamPtr &stream)

  Creation function to make a random access stream from a stream that is not random access. Currently, the temporary store will be used to store a copy of the source data, but a concrete implementation method should not be assumed. If the source stream is random access, it will be returned as is. Throws an IEDLError exception on failure.
- static EDL_API IRAInputStreamPtr createFromNewFileWithContents (IEDLClassFactory *pFactory, const EDLSysString &path, const IInputStreamPtr &stream)

  Creation function for an IInputStream for a file on disk created with the contents of an existing stream. Throws an IEDLError exception on failure.

- static EDL_API IRAInputStreamPtr createFromNewFileWithContents (IEDLClassFactory *pFactory, const EDLString &path, const IInputStreamPtr &stream)

  Creation function for an IInputStream for a file on disk created with the contents of an existing stream. Throws an IEDLError exception on failure.
- static EDL_API IInputStreamPtr createSubFile (IEDLClassFactory *pFactory, const IInputStreamPtr &stream, int64 offset, int64 length)

  Creation routine for a stream representing a portion of a file on disk. If the source file is random access, then the created file shall be also. Throws an IEDLError exception on failure.
- static EDL_API IInputStreamPtr createFromFlateCompressed (IEDLClassFactory *pFactory, const IInput↩ StreamPtr &stream, bool raw=true, bool ignoreChecksums=false)

  Creation routine for a input stream for decompressing a flate stream. Throws an IEDLError exception on failure.
- static EDL_API IInputStreamPtr createFromLz4Compressed (IEDLClassFactory *pFactory, const IInput↩ StreamPtr &stream)

  Creation routine for a input stream for decompressing an lz4 block compressed stream. Throws an IEDLError exception on failure. Note: This is not intended for interoperability with other LZ4 formats, but is useful for things like temporary storage.
- static EDL_API IInputStreamPtr createFromPredictorCompressed (IEDLClassFactory *pFactory, const IInputStreamPtr &stream, uint8 predictor, uint8 colors, uint8 bitsPerComponent, uint32 columns)

  Creation routine for a input stream for applying a predictor algorithm to a compressed stream. Throws an IEDLError exception on failure.
- static EDL_API IInputStreamPtr createCompositeStream (IEDLClassFactory *pFactory, const CIInput↩ StreamVect &streams)

  Creation routine for creating a composite input stream representing the concatenation of a series of streams.
- static EDL_API IInputPushbackStreamPtr createPushbackStream (IEDLClassFactory *pFactory, const IInputStreamPtr &sourceStream, bool clone=true)

  Creation routine for creating a push-back stream using a non-pushback stream as a data source. If the source stream is random access, the resulting stream will also be random access.
- static EDL_API IInputStreamPtr createUelStream (IEDLClassFactory *pFactory, const IInputStreamPtr &sourceStream)

  Creation routine for a stream that emulates an end-of-file condition should a Universal End of Language (UEL) sequence be encountered in an underlying data source. The resulting stream is not random access. The underlying stream is neither opened nor closed. This stream type is useful for restricting parsing of substreams inside a print stream. Streams of this type are not clonable. Please also note, that this stream must read ahead of read requests in order to search for UEL sequences. However it will never read beyond the UEL sequence.
- static EDL_API IInputStreamPtr createTbcpStream (IEDLClassFactory *pFactory, const IInputStreamPtr &sourceStream, bool openSource=false)

  Creation routine for a stream that reads data from an underlying stream encoded using Adobe's Tagged Binary Communication Protocol (TBCP). The resulting stream is not random access. Streams of this type are not clonable.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.322.1 Detailed Description

Input Stream with pushback support.

The documentation for this class was generated from the following file:

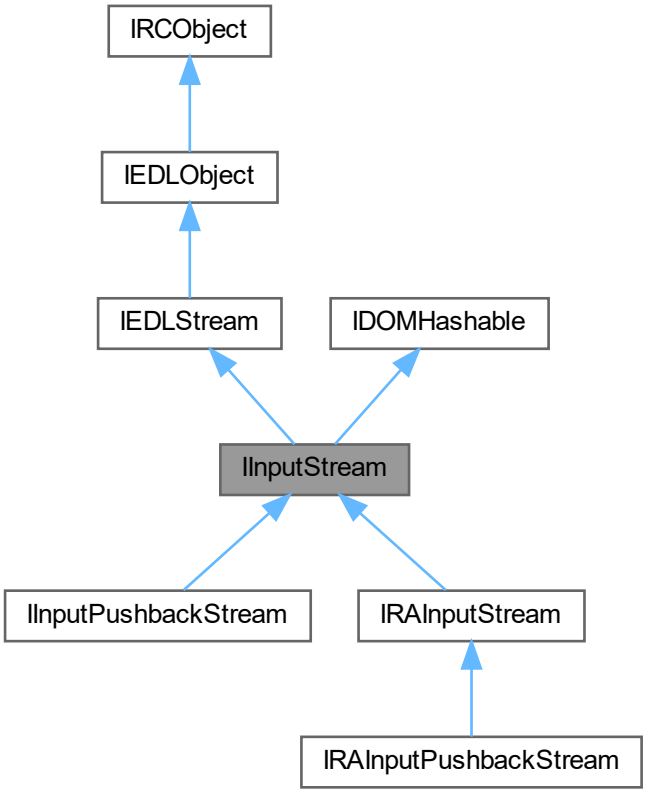
- edlstream.h

7.323 IInputStream Class Reference

Generic input stream. Abstract base class for all input streams.

`#include <edlstream.h>`

Inheritance diagram for IInputStream:



Public Member Functions

- virtual int32 read (void *buffer, int32 count)=0

  Read specified number of bytes from a stream into buffer.
- virtual int8 read ()=0

  Read single byte from a stream.
- virtual bool eof () const =0

  Determine if the stream has exhausted.
- virtual int64 skip (int64 count)

  Skip a specified number of bytes.
- virtual bool getSourceFilePath (EDLSysString &sourcePath)=0

  If available, find the file path of the file that this stream references, or for streams that take their input from other streams, the ultimate source stream. If at all possible, the source path will be canonicalised, but this is not guaranteed. If a canonical name is absolutely required, check getCanonicalSourceFilePath().

- virtual bool getCanonicalSourceFilePath (EDLSysString &sourceCanonicalPath)=0

  If available, find the canonical file path of the file that this stream references, or for streams that take their input from other streams, the ultimate source stream. For canonical names to be discovered on all platforms, the underlying file must exist. If the canonical file path cannot be determined, false will be returned. In which case, you may wish to try getSourceFilePath() above.
- virtual bool completeRead (void *buffer, int32 count)

  Perform a complete read.
- virtual void completeReadE (void *buffer, int32 count)

  As completeRead(), but throws an exception if the operation fails.
- virtual bool hash (uint64 &hash)

  Obtain a 64-bit hash of the stream. Please note that this requires reading the stream and is therefore not thread safe. If thread safety is desired, make a clone of the stream first.

Public Member Functions inherited from IEDLStream

- virtual bool isValid () const =0

  Determine stream validity.
- virtual bool open ()

  Opens the stream.
- virtual void **openE** ()=0

  As per open(), but will throw an exception on failure (IEDLError) that for some stream types may contain additional failure information.
- virtual void **close** ()=0

  Closes the stream.
- virtual int64 getPos ()=0

  Get current stream position.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Static Public Member Functions

- static EDL_API IRAInputStreamPtr createFromFile (IEDLClassFactory *pFactory, const EDLSysString &path)

  Creation function for an IInputStream for a file on disk. Throws an IEDLError exception on failure.
- static EDL_API IRAInputStreamPtr createFromFile (IEDLClassFactory *pFactory, const EDLString &path)

  Creation function for an IInputStream for a file on disk. Throws an IEDLError exception on failure.
- static EDL_API IRAInputStreamPtr createFromFileShared (IEDLClassFactory *pFactory, const EDLSys↩ String &path)

  Creation function for an IInputStream for a file on disk. Similar to createFromFile, but if this file is cloned, only one file handle will be open at a time. This is slower, but is useful if many users are likely to want to have this file open at the same file. On some systems, having too many files open may result in errors. Performance should be acceptable providing there is not a great deal of contention and simultaneous access.
- static EDL_API IRAInputStreamPtr createFromFileShared (IEDLClassFactory *pFactory, const EDLString &path)

  Creation function for an IInputStream for a file on disk. Similar to createFromFile, but if this stream is cloned, only one file handle will be open at a time. This is slower, but is useful if many users are likely to want to have this file open at the same file. On some systems, having too many files open may result in errors. Performance should be acceptable providing there is not a great deal of contention and simultaneous access.
- static EDL_API IRAInputStreamPtr createSharedFromStream (IEDLClassFactory *pFactory, const IRAInputStreamPtr &stream)

  Creation function for a shared stream overlaying an existing stream. If this stream is cloned, the underlying file will not be cloned. This is useful if many users are likely to want to have this file open at the same time. On some systems, having too many files open may result in errors. Performance should be acceptable providing there is not a great deal of contention and simultaneous access. Only possible for random access streams. If the stream is already shared, the stream will be returned as is.
- static EDL_API IRAInputStreamPtr createFromMemory (IEDLClassFactory *pFactory, const void *mem, uint32 length, bool copy=false, bool free=true)

  Creation function for an IInputStream for data in memory. Throws an IEDLError exception on failure.
- static EDL_API IInputStreamPtr createFromUserReadFunc (IEDLClassFactory *pFactory, UserStream↩ ReadFunc readFunc, void *priv)

  Creation function for an IInputStream from a user function that provides data. Throws an IEDLError exception on failure.
- static EDL_API IRAInputStreamPtr createFromRAUserFunc (IEDLClassFactory *pFactory, int64 length, UserRAReadFunc readFunc, void *priv)

  Creation function for an IRAInputStream from a user function that provides random access. Throws an IEDLError exception on failure.
- static EDL_API IRAInputStreamPtr createRandomAccessFromNonRandomAccess (IEDLClassFactory *p↩ Factory, const IInputStreamPtr &stream)

  Creation function to make a random access stream from a stream that is not random access. Currently, the temporary store will be used to store a copy of the source data, but a concrete implementation method should not be assumed. If the source stream is random access, it will be returned as is. Throws an IEDLError exception on failure.
- static EDL_API IRAInputStreamPtr createFromNewFileWithContents (IEDLClassFactory *pFactory, const EDLSysString &path, const IInputStreamPtr &stream)

  Creation function for an IInputStream for a file on disk created with the contents of an existing stream. Throws an IEDLError exception on failure.
- static EDL_API IRAInputStreamPtr createFromNewFileWithContents (IEDLClassFactory *pFactory, const EDLString &path, const IInputStreamPtr &stream)

Creation function for an IInputStream for a file on disk created with the contents of an existing stream. Throws an IEDLError exception on failure.

- static EDL_API IInputStreamPtr createSubFile (IEDLClassFactory *pFactory, const IInputStreamPtr &stream, int64 offset, int64 length)

  Creation routine for a stream representing a portion of a file on disk. If the source file is random access, then the created file shall be also. Throws an IEDLError exception on failure.
- static EDL_API IInputStreamPtr createFromFlateCompressed (IEDLClassFactory *pFactory, const IInput↩ StreamPtr &stream, bool raw=true, bool ignoreChecksums=false)

  Creation routine for a input stream for decompressing a flate stream. Throws an IEDLError exception on failure.
- static EDL_API IInputStreamPtr createFromLz4Compressed (IEDLClassFactory *pFactory, const IInput↩ StreamPtr &stream)

  Creation routine for a input stream for decompressing an lz4 block compressed stream. Throws an IEDLError exception on failure. Note: This is not intended for interoperability with other LZ4 formats, but is useful for things like temporary storage.
- static EDL_API IInputStreamPtr createFromPredictorCompressed (IEDLClassFactory *pFactory, const IInputStreamPtr &stream, uint8 predictor, uint8 colors, uint8 bitsPerComponent, uint32 columns)

  Creation routine for a input stream for applying a predictor algorithm to a compressed stream. Throws an IEDLError exception on failure.
- static EDL_API IInputStreamPtr createCompositeStream (IEDLClassFactory *pFactory, const CIInput↩ StreamVect &streams)

  Creation routine for creating a composite input stream representing the concatenation of a series of streams.
- static EDL_API IInputPushbackStreamPtr createPushbackStream (IEDLClassFactory *pFactory, const IInputStreamPtr &sourceStream, bool clone=true)

  Creation routine for creating a push-back stream using a non-pushback stream as a data source. If the source stream is random access, the resulting stream will also be random access.
- static EDL_API IInputStreamPtr createUelStream (IEDLClassFactory *pFactory, const IInputStreamPtr &sourceStream)

  Creation routine for a stream that emulates an end-of-file condition should a Universal End of Language (UEL) sequence be encountered in an underlying data source. The resulting stream is not random access. The underlying stream is neither opened nor closed. This stream type is useful for restricting parsing of substreams inside a print stream. Streams of this type are not clonable. Please also note, that this stream must read ahead of read requests in order to search for UEL sequences. However it will never read beyond the UEL sequence.
- static EDL_API IInputStreamPtr createTbcpStream (IEDLClassFactory *pFactory, const IInputStreamPtr &sourceStream, bool openSource=false)

  Creation routine for a stream that reads data from an underlying stream encoded using Adobe's Tagged Binary Communication Protocol (TBCP). The resulting stream is not random access. Streams of this type are not clonable.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.323.1 Detailed Description

Generic input stream. Abstract base class for all input streams.

7.323.2 Member Function Documentation

completeRead()

```
virtual bool IInputStream::completeRead (
            void * buffer,
            int32 count )  [virtual]
```

Perform a complete read.

Parameters

buffer	Buffer to accept the data.
count	Number of bytes to be read into the buffer.

Returns

bool True if the read operation was successful, or false if the read could not be completely fulfilled.

completeReadE()

```
virtual void IInputStream::completeReadE (
            void * buffer,
            int32 count )  [virtual]
```

As completeRead(), but throws an exception if the operation fails.

Parameters

buffer	Address of buffer.
count	Number of bytes to be read.

createCompositeStream()

```
static EDL_API IInputStreamPtr IInputStream::createCompositeStream (
            IEDLClassFactory * pFactory,
            const CIInputStreamVect & streams )  [static]
```

Creation routine for creating a composite input stream representing the concatenation of a series of streams.

Parameters

pFactory	The class factory.
streams	The vector of streams to use.

Returns

IInputStreamPtr The new input stream.

createFromFile() [1/2]

```
static EDL_API IRAInputStreamPtr IInputStream::createFromFile (
            IEDLClassFactory * pFactory,
            const EDLString & path )  [static]
```

Creation function for an IInputStream for a file on disk. Throws an IEDLError exception on failure.

Parameters

pFactory	The class factory.
path	Path to the file.

Returns

IRAInputStreamPtr The new input stream.

createFromFile() [2/2]

```
static EDL_API IRAInputStreamPtr IInputStream::createFromFile (
            IEDLClassFactory * pFactory,
            const EDLSysString & path )  [static]
```

Creation function for an IInputStream for a file on disk. Throws an IEDLError exception on failure.

Parameters

pFactory	The class factory.
path	Path to the file.

Returns

IRAInputStreamPtr The new input stream.

createFromFileShared() [1/2]

```
static EDL_API IRAInputStreamPtr IInputStream::createFromFileShared (
            IEDLClassFactory * pFactory,
            const EDLString & path )  [static]
```

Creation function for an IInputStream for a file on disk. Similar to createFromFile, but if this stream is cloned, only one file handle will be open at a time. This is slower, but is useful if many users are likely to want to have this file open at the same file. On some systems, having too many files open may result in errors. Performance should be acceptable providing there is not a great deal of contention and simultaneous access.

Parameters

pFactory	The class factory.
path	Path to the file.

Returns

IRAInputStreamPtr The new input stream.

createFromFileShared() [2/2]

```
static EDL_API IRAInputStreamPtr IInputStream::createFromFileShared (
            IEDLClassFactory * pFactory,
            const EDLSysString & path )  [static]
```

Creation function for an IInputStream for a file on disk. Similar to createFromFile, but if this file is cloned, only one file handle will be open at a time. This is slower, but is useful if many users are likely to want to have this file open at the same file. On some systems, having too many files open may result in errors. Performance should be acceptable providing there is not a great deal of contention and simultaneous access.

Parameters

pFactory	The class factory.
path	Path to the file.

Returns

IRAInputStreamPtr The new input stream.

createFromFlateCompressed()

```
static EDL_API IInputStreamPtr IInputStream::createFromFlateCompressed (
            IEDLClassFactory * pFactory,
            const IInputStreamPtr & stream,
            bool raw = true,
            bool ignoreChecksums = false )  [static]
```

Creation routine for a input stream for decompressing a flate stream. Throws an IEDLError exception on failure.

Parameters

pFactory	The class factory.
stream	The compressed stream.
raw	Pass to true if the flate stream has no zlib header.
ignoreChecksums	Pass true if checksum errors should be ignored.

Returns

IInputStreamPtr The new input stream.

createFromLz4Compressed()

```
static EDL_API IInputStreamPtr IInputStream::createFromLz4Compressed (
            IEDLClassFactory * pFactory,
            const IInputStreamPtr & stream )  [static]
```

Creation routine for a input stream for decompressing an lz4 block compressed stream. Throws an IEDLError exception on failure. Note: This is not intended for interoperability with other LZ4 formats, but is useful for things like temporary storage.

Parameters

pFactory	The class factory.
stream	The compressed stream.

Returns

IInputStreamPtr The new input stream.

createFromMemory()

```
static EDL_API IRAInputStreamPtr IInputStream::createFromMemory (
            IEDLClassFactory * pFactory,
            const void * mem,
            uint32 length,
            bool copy = false,
            bool free = true )  [static]
```

Creation function for an IInputStream for data in memory. Throws an IEDLError exception on failure.

Parameters

pFactory	The class factory.
mem	A pointer to the memory to be used as the source data for the stream.
length	The length of the buffer pointed to by mem.
copy	If true, a copy of the memory pointed to by mem will be made and will be managed and released by the stream.
free	If copy is false and free is true, then when the stream is destroyed, mem will be deallocated using using stdlib free(). Ignored if copy is true.

Returns

IInputStreamPtr The new input stream.

createFromNewFileWithContents() [1/2]

```
static EDL_API IRAInputStreamPtr IInputStream::createFromNewFileWithContents (
            IEDLClassFactory * pFactory,
            const EDLString & path,
            const IInputStreamPtr & stream )  [static]
```

Creation function for an IInputStream for a file on disk created with the contents of an existing stream. Throws an IEDLError exception on failure.

Parameters

pFactory	The class factory.
path	Path to the file.
stream	The stream to populate the new file with.

Returns

IInputStreamPtr The new input stream.

createFromNewFileWithContents() [2/2]

```
static EDL_API IRAInputStreamPtr IInputStream::createFromNewFileWithContents (
            IEDLClassFactory * pFactory,
            const EDLSysString & path,
            const IInputStreamPtr & stream )  [static]
```

Creation function for an IInputStream for a file on disk created with the contents of an existing stream. Throws an IEDLError exception on failure.

Parameters

pFactory	The class factory.
path	Path to the file.
stream	The stream to populate the new file with.

Returns

IInputStreamPtr The new input stream.

createFromPredictorCompressed()

```
static EDL_API IInputStreamPtr IInputStream::createFromPredictorCompressed (
            IEDLClassFactory * pFactory,
            const IInputStreamPtr & stream,
            uint8 predictor,
            uint8 colors,
            uint8 bitsPerComponent,
            uint32 columns )  [static]
```

Creation routine for a input stream for applying a predictor algorithm to a compressed stream. Throws an IEDLError exception on failure.

Parameters

pFactory	The class factory.
stream	The compressed stream.
predictor	The predictor algorithm to apply. Valid values are 2 (TIFF), 10 (PNG None), 11 (PNG Sub), 12 (PNG Up), 13 (PNG average) or 14 (PNG Paeth).
colors	The number of interleaved color components per sample, which must be greater or equal to 1.
bitsPerComponent	The number of bits used to represent each color component in a sample. Valid values are 1, 2, 4, 8, and 16.
columns	The number of samples in each row.

Returns

IInputStreamPtr The new input stream.

createFromRAUserFunc()

```
static EDL_API IRAInputStreamPtr IInputStream::createFromRAUserFunc (
            IEDLClassFactory * pFactory,
            int64 length,
            UserRAReadFunc readFunc,
            void * priv )  [static]
```

Creation function for an IRAInputStream from a user function that provides random access. Throws an IEDLError exception on failure.

Parameters

pFactory	The class factory.
length	The stream length.
readFunc	A function that when called, will provide the data for the stream.
priv	An opaque private pointer that is passed to the readFunc on each call.

Returns

IInputStreamPtr The new input stream.

createFromUserReadFunc()

```
static EDL_API IInputStreamPtr IInputStream::createFromUserReadFunc (
            IEDLClassFactory * pFactory,
            UserStreamReadFunc readFunc,
            void * priv )  [static]
```

Creation function for an IInputStream from a user function that provides data. Throws an IEDLError exception on failure.

Parameters

pFactory	The class factory.
readFunc	A function that when called, will provide the data for the stream.
priv	An opaque private pointer that is passed to the readFunc on each call.

Returns

IInputStreamPtr The new input stream.

createPushbackStream()

```
static EDL_API IInputPushbackStreamPtr IInputStream::createPushbackStream (
            IEDLClassFactory * pFactory,
```

```
            const IInputStreamPtr & sourceStream,
            bool clone = true )  [static]
```

Creation routine for creating a push-back stream using a non-pushback stream as a data source. If the source stream is random access, the resulting stream will also be random access.

Parameters

pFactory	The class factory.
sourceStream	The source stream to use.
clone	If true, the input stream is cloned. If false, the existing stream is used in place.

Returns

IInputStreamPtr The new input stream.

createRandomAccessFromNonRandomAccess()

```
static EDL_API IRAInputStreamPtr IInputStream::createRandomAccessFromNonRandomAccess (
            IEDLClassFactory * pFactory,
            const IInputStreamPtr & stream )  [static]
```

Creation function to make a random access stream from a stream that is not random access. Currently, the temporary store will be used to store a copy of the source data, but a concrete implementation method should not be assumed. If the source stream is random access, it will be returned as is. Throws an IEDLError exception on failure.

Parameters

pFactory	The class factory. Must be a Mako instance.
stream	The stream to use for source.

Returns

IRAInputStreamPtr A random access stream.

createSharedFromStream()

```
static EDL_API IRAInputStreamPtr IInputStream::createSharedFromStream (
            IEDLClassFactory * pFactory,
            const IRAInputStreamPtr & stream )  [static]
```

Creation function for a shared stream overlaying an existing stream. If this stream is cloned, the underlying file will not be cloned. This is useful if many users are likely to want to have this file open at the same time. On some systems, having too many files open may result in errors. Performance should be acceptable providing there is not a great deal of contention and simultaneous access. Only possible for random access streams. If the stream is already shared, the stream will be returned as is.

Parameters

pFactory	The class factory.
stream	The source stream to share. Will be cloned.

Returns

IRAInputStreamPtr The new input stream.

createSubFile()

```
static EDL_API IInputStreamPtr IInputStream::createSubFile (
            IEDLClassFactory * pFactory,
            const IInputStreamPtr & stream,
            int64 offset,
            int64 length )  [static]
```

Creation routine for a stream representing a portion of a file on disk. If the source file is random access, then the created file shall be also. Throws an IEDLError exception on failure.

Parameters

pFactory	The class factory.
stream	The stream to use for source.
offset	The offset within the source stream of the start of the data to use.
length	The length of source data to use.

Returns

IInputStreamPtr The new input stream.

createTbcpStream()

```
static EDL_API IInputStreamPtr IInputStream::createTbcpStream (
            IEDLClassFactory * pFactory,
            const IInputStreamPtr & sourceStream,
            bool openSource = false )  [static]
```

Creation routine for a stream that reads data from an underlying stream encoded using Adobe's Tagged Binary Communication Protocol (TBCP). The resulting stream is not random access. Streams of this type are not clonable.

Parameters

pFactory	The class factory.
sourceStream	The source stream to use.
openSource	If true, opening the stream will open the underlying stream, and the underlying stream will be closed when the stream closes. When set to false the underlying stream is neither opened nor closed. The default is false.

Returns

IInputStreamPtr The new input stream.

createUelStream()

```
static EDL_API IInputStreamPtr IInputStream::createUelStream (
            IEDLClassFactory * pFactory,
            const IInputStreamPtr & sourceStream )  [static]
```

Creation routine for a stream that emulates an end-of-file condition should a Universal End of Language (UEL) sequence be encountered in an underlying data source. The resulting stream is not random access. The underlying stream is neither opened nor closed. This stream type is useful for restricting parsing of substreams inside a print stream. Streams of this type are not clonable. Please also note, that this stream must read ahead of read requests in order to search for UEL sequences. However it will never read beyond the UEL sequence.

Parameters

pFactory	The class factory.
sourceStream	The source stream to use.

Returns

IInputStreamPtr The new input stream.

eof()

```
virtual bool IInputStream::eof ( ) const  [pure virtual]
```

Determine if the stream has exhausted.

Returns

bool True if the stream has exhausted.

getCanonicalSourceFilePath()

```
virtual bool IInputStream::getCanonicalSourceFilePath (
            EDLSysString & sourceCanonicalPath )  [pure virtual]
```

If available, find the canonical file path of the file that this stream references, or for streams that take their input from other streams, the ultimate source stream. For canonical names to be discovered on all platforms, the underlying file must exist. If the canonical file path cannot be determined, false will be returned. In which case, you may wish to try getSourceFilePath() above.

Parameters

sourceCanonicalPath	Reference to receive the UTF-8 path to the ultimate source file, if available.

Returns

bool True if a canonical path could be retrieved.

getSourceFilePath()

```
virtual bool IInputStream::getSourceFilePath (
            EDLSysString & sourcePath )  [pure virtual]
```

If available, find the file path of the file that this stream references, or for streams that take their input from other streams, the ultimate source stream. If at all possible, the source path will be canonicalised, but this is not guaranteed. If a canonical name is absolutely required, check getCanonicalSourceFilePath().

Parameters

sourcePath	Reference to receive the UTF-8 path to the ultimate source file, if available.

Returns

bool True if a path could be retrieved.

hash()

```
virtual bool IInputStream::hash (
            uint64 & hash )  [virtual]
```

Obtain a 64-bit hash of the stream. Please note that this requires reading the stream and is therefore not thread safe. If thread safety is desired, make a clone of the stream first.

Parameters

hash	A reference to receive the hash value.

Returns

bool True if method succeeded.

Implements IDOMHashable.

read() [1/2]

```
virtual int8 IInputStream::read ( )  [pure virtual]
```

Read single byte from a stream.

Returns

int8 Number of bytes actually read.

read() [2/2]

```
virtual int32 IInputStream::read (
            void * buffer,
            int32 count )  [pure virtual]
```

Read specified number of bytes from a stream into buffer.

Parameters

buffer	Buffer to accept the data.
count	Number of bytes to read.

Returns

int32 The number of bytes actually read.

skip()

```
virtual int64 IInputStream::skip (
             int64 count )  [inline], [virtual]
```

Skip a specified number of bytes.

Parameters

count	Number of bytes to skip.

Returns

int64 The number of bytes actually skipped.

The documentation for this class was generated from the following file:

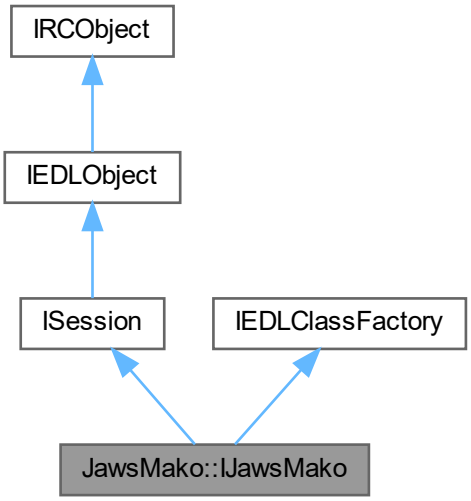
- edlstream.h

7.324 JawsMako::IJawsMako Class Reference

An instance of the IJawsMako library. Only one instance of this object is currently allowed. This class may also be used as both an EDL factory and an EDL session, and passed to any EDL API that requires these objects.

```
#include <jawsmako.h>
```

Inheritance diagram for JawsMako::IJawsMako:



Public Member Functions

- virtual void **freeMemory** ()=0

  Inform JawsMako that system memory is being exhausted. This will cause JawsMako to free up cached information and discard any unmodified pages.
- virtual IDOMFontPtr findFont (const U8String &fontName, uint32_t &fontIndex)=0

  Find a font from the system fonts, and the font index if applicable.

Public Member Functions inherited from ISession

- virtual bool setFactory (IEDLClassFactory *pFactory)=0

  initializes Session by setting EDL class Factory
- virtual IEDLClassFactory * getFactory ()=0

  EDL Class Factory getter method.
- virtual IMessageHandlerPtr getMessageHandler ()=0

  Obtain the session's message handler.
- virtual ILiteMessageHandlerPtr getLiteMessageHandler ()=0

  Obtain the session's litemessage handler.
- virtual bool setTemporaryDirectory (const EDLSysString &sTempDirectory)=0

  Set the temporary directory to be used by EDL and filters for this session.
- virtual bool getTemporaryDirectory (EDLSysString &sTempDirectory)=0

  Get the temporary directory to be used by EDL and filters for this session.
- virtual IEDLTempStorePtr getTempStore ()=0

  Get the temporary store for this session. The temporary directory must be set before calling this member. Throws an IEDLError on failure.
- virtual bool setStartupDirectory (const EDLSysString &sStartupDirectory)=0

  Set the startup directory. This is meaningfull for executable environments. Note the client must set this during initalisation for it to be valid.
- virtual bool getStartupDirectory (EDLSysString &sStartupDirectory)=0

  Get the startup directory.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IEDLClassFactory

- virtual bool registerNamedClass (const EDLSysString &strName, const CClassID &id, bool overwrite=true)=0

  Register a GUID under a string name.
- virtual bool findNamedClass (const EDLSysString &strName, CClassID &id)=0

  Retrieve GUID registered under the string name.
- virtual bool registerClass (const CClassID &id, creatorFunc f)=0

  Register a GUID with creator function.
- virtual IEDLObjectPtr createInstance (const CClassID &id, CClassParams *pParams=NULL)=0

  Creates the registered class by CClassID.
- virtual IEDLObjectPtr createInstanceJawsMako (const CClassID &id, CClassParams *pParams=NULL, IEDLClassFactory *factory=NULL)=0

  Creates the registered class by CClassID.
- virtual IEDLObjectPtr getSingleton (const CClassID &id)=0

  Creates the EDL singleton (for example FontLibrary or ColorManager)

Static Public Member Functions

- static JAWSMAKO_API IJawsMakoPtr create (const U8String &tempDir=U8String(""), const U8String &cacheDir=U8String(""), const CTemporaryStoreParameters &tempStoreParams=CTemporaryStoreParameters(), const std::map< std::string, std::string > &initialParams=std::map< std::string, std::string >())

  Create an IJawsMako instance.
  Only one may be created at any one time.
- static JAWSMAKO_API void **enablePDFInput** (const IJawsMakoPtr &jawsMako)

  Enable PDF Input for JawsMako.
- static JAWSMAKO_API void **enablePDFOutput** (const IJawsMakoPtr &jawsMako)

  Enable PDF Output for JawsMako.
- static JAWSMAKO_API void enablePSInput (const IJawsMakoPtr &jawsMako)

  Enable PostScript Input for JawsMako.
- static JAWSMAKO_API void **enablePSOutput** (const IJawsMakoPtr &jawsMako)

  Enable PostScript Output for JawsMako.
- static JAWSMAKO_API void **enableAllFeatures** (const IJawsMakoPtr &jawsMako)

  Enable all JawsMako features.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()
  Virtual destructor.

7.324.1 Detailed Description

An instance of the IJawsMako library. Only one instance of this object is currently allowed. This class may also be used as both an EDL factory and an EDL session, and passed to any EDL API that requires these objects.

7.324.2 Member Function Documentation

enablePSInput()

```
static JAWSMAKO_API void JawsMako::IJawsMako::enablePSInput (
            const IJawsMakoPtr & jawsMako )  [static]
```

Enable PostScript Input for JawsMako.

Note that enabling PostScript will also enable PDF Input, as for PostScript the input is converted and then opened as PDF.

findFont()

```
virtual IDOMFontPtr JawsMako::IJawsMako::findFont (
            const U8String & fontName,
            uint32_t & fontIndex )  [pure virtual]
```

Find a font from the system fonts, and the font index if applicable.

Only TrueType, TrueType collections or OpenType/CFF fonts are supported. Matching is currently exact; there is no attempt to find a font that is similarly named. Fonts are matched based on their Full or PostScript font names.

The list of fonts on the system is cached and refreshed once per JawsMako instance creation.

An IError of code JM_ERR_FONT_NOT_FOUND will be thrown if the font could not be found.

Parameters

fontName	The font name of the desired font.
fontIndex	If the requested font is part of a TrueType collection, this will be set to the index of the font within the collection, or 0 otherwise.

Returns

IDOMFontPtr The found font.

The documentation for this class was generated from the following file:

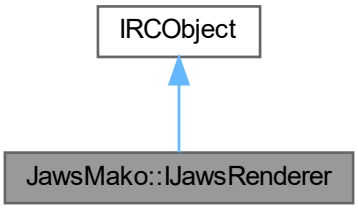
- jawsmako.h

7.325 JawsMako::IJawsRenderer Class Reference

A renderer that uses the Jaws RIP to create images from arbitrary DOM.

`#include <jawsmako.h>`

Inheritance diagram for JawsMako::IJawsRenderer:



Classes

- class CColorSpotHalftone

  Description of spot halftones, using Jaws's default spot function. Used for color halftoned rendering. Analogous to a PostScript Type 2 Halftone; please refer to section 7.4.6 of the PostScript language reference manual. When rendering to RGB, there must be three angles. (Red, Green and Blue respectively). For CMYK output, there must be four (Cyan, Magenta, Yellow and Black respectively).
- class CColorThresholdArrayHalftone

  As per CThresholdArrayHalftone, but for use with color rendering. There is a single width and height, but a threshold array for each color being rendering. One threshold array must be specified for each color being rendered. When rendering to RGB, there must be three thresholds (Red, Green and Blue respectively). For CMYK output, there must be four thresholds (Cyan, Magenta, Yellow and Black respectively).
- class CEDSHalftone

  A halftone representing an error diffusion screen. Allows the production of results containing a variable number of gray levels per channel using a range of error diffusion screens.
- class CFrameBufferInfo

  Description of a frame buffer for use with renderSeparationsToFrameBuffers.
- class CSpotHalftone

  Description of a simple spot halftone, at 45 degrees, using Jaws's default spot function. Used for monochrome rendering.
- class CThresholdArrayHalftone

  Description of a Type 3 8-bit threshold array halftone for use with monochrome rendering. Please refer to section 7.4.5 of the PostScript language reference manual, 3rd edition.
- class CThresholdHalftone

  A halftone representing a simple threshold. Used for monochrome rendering.

Public Member Functions

- virtual IDOMImagePtr render (const IDOMNodePtr &node, uint32 dpi=72, uint8 depth=8, const IDOMColorSpacePtr &colorSpace=IDOMColorSpacePtr(), const FRect &bounds=FRect(), bool maskToInteresting Nodes=false, const CSpotColorNames &retainedSpotColors=CSpotColorNames(), const IOptionalContent Ptr &optionalContent=IOptionalContentPtr(), eOptionalContentEvent optionalContentUsage=eOCEView)=0

  Render a node to an image.
- virtual CEDLVector< IDOMImagePtr > renderSeparations (const IDOMNodePtr &node, const uint8 depth=8, const IDOMColorSpacePtr &colorSpace=IDOMColorSpacePtr(), uint8 antiAliased=0, const FRect &bounds=FRect(), uint32 destWidth=0, uint32 destHeight=0, const CSpotColorNames &retained SpotColors=CSpotColorNames(), const IOptionalContentPtr &optionalContent=IOptionalContentPtr(), eOptionalContentEvent optionalContentUsage=eOCEView, const CU8StringVect &processColor Names=CU8StringVect(), bool alphaGeneration=false, uint64 bandMemorySize=0, const CSpotColorNames &ignoreSpotColors=CSpotColorNames())=0

  Render a node to a set of separated images, with optional anti-aliasing. For ease of use, the output is scaled to fit the destination size, described in pixels.
- virtual void renderSeparationsToFrameBuffers (const IDOMNodePtr &node, uint8 depth, bool host Endian, uint32 destWidth, uint32 destHeight, const IDOMColorSpacePtr &colorSpace, CFrameBuffer InfoVect &frameBuffers, uint8 antiAliased=0, const FRect &bounds=FRect(), const CSpotColorNames &retainedSpotColors=CSpotColorNames(), const IOptionalContentPtr &optionalContent=IOptionalContent Ptr(), eOptionalContentEvent optionalContentUsage=eOCEView, const CU8StringVect &processColor Names=CU8StringVect(), bool alphaGeneration=false, uint64 bandMemorySize=0, const CSpotColorNames &ignoreSpotColors=CSpotColorNames())=0

  As per renderSeparations(), but targeting a set of frame buffers, described by a vector CFrameBufferInfoVect.
- virtual void renderToFrameBuffer (const IDOMNodePtr &node, void *buffer, uint32 stride, uint32 dest Width, uint32 destHeight, const IDOMColorSpacePtr &colorSpace=IDOMColorSpacePtr(), const FRect &bounds=FRect(), const IOptionalContentPtr &optionalContent=IOptionalContentPtr(), eOptionalContentEvent optionalContentUsage=eOCEView)=0

  Render a node to a supplied frame buffer, at 8 bits per component.
- virtual void renderToFrameBufferPadAndReverse (const IDOMNodePtr &node, void *buffer, uint32 stride, uint32 pixelStride, bool reverseComponents, uint32 destWidth, uint32 destHeight, const IDOMColorSpacePtr &colorSpace=IDOMColorSpacePtr(), const FRect &bounds=FRect(), const IOptionalContentPtr &optional Content=IOptionalContentPtr(), eOptionalContentEvent optionalContentUsage=eOCEView)=0

  As per renderToFrameBuffer, but allowing for pixel byte order and padding control.
- virtual IDOMImagePtr renderAntiAliased (const IDOMNodePtr &node, uint32 dpi=72, uint8 quality=3, const FRect &bounds=FRect())=0

  Render a node to an antialiased, DeviceRGB image suitable for display, using supersampling.
- virtual CEDLVector< IDOMImagePtr > renderScreened (const IDOMNodePtr &node, uint32 dpi, const IHalftone *halftone, const IDOMColorSpacePtr &colorSpace=IDOMColorSpacePtr(), const FRect &bounds=FRect(), const bool useCC=true, const IOptionalContentPtr &optionalContent=IOptionalContent Ptr(), eOptionalContentEvent optionalContentUsage=eOCEView)=0

  Render a node to a screened result, in monochrome or CMYK returning one image per colorant.
- IDOMImagePtr renderMonochrome (const IDOMNodePtr &node, uint32 dpi=72, const IHalftone *halftone=NULL, const IDOMColorSpacePtr &colorSpace=IDOMColorSpacePtr(), const FRect &bounds=FRect(), const bool useCC=false)

  Convenience - Render a node to a 1-bit monochrome image.
- virtual void renderScreenedToFrameBuffers (const IDOMNodePtr &node, void **buffers, uint32 *strides, uint32 destWidth, uint32 destHeight, const IHalftone *halftone, const IDOMColorSpace Ptr &colorSpace=IDOMColorSpacePtr(), const FRect &bounds=FRect(), const bool useCC=true, const IOptionalContentPtr &optionalContent=IOptionalContentPtr(), eOptionalContentEvent optionalContent Usage=eOCEView)=0

  Render a node to a 1-bit monochrome frame buffer, a series of three 1-bit frame buffers for RGB, or four 1-bit frame buffers representing CMYK.
- void renderMonochromeToFrameBuffer (const IDOMNodePtr &node, void *buffer, uint32 stride, uint32 dest Width, uint32 destHeight, const IHalftone *halftone, const IDOMColorSpacePtr colorSpace=IDOMColor SpacePtr(), const FRect &bounds=FRect(), const bool useCC=false, const IOptionalContentPtr &optional Content=IOptionalContentPtr(), eOptionalContentEvent optionalContentUsage=eOCEView)

Convenience Render a node to a 1-bit monochrome frame buffer.

- virtual void setEnableTrueTypeNotDef (bool enable)=0

  Enable the use of a True Type font's /.notdef glyph in the output.
- virtual void setEnableRasterCompression (bool enable)=0

  Enable the use of compression for compressed output.
- virtual void setBlackPreservation (bool preserveForText, bool preserveForOther)=0

  Enable/control 100% black preservation.
- virtual void setIgnoreMatchingDeviceIntercept (bool ignoreDeviceIntercept)=0

  Allow intercept color spaces for a matching Device space to be ignored.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IJawsRendererPtr create (const IJawsMakoPtr &jawsMako, const IProgressMonitor↩ Ptr &progressMonitor=IProgressMonitorPtr())

  Create a Jaws renderer instance.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.325.1 Detailed Description

A renderer that uses the Jaws RIP to create images from arbitrary DOM.

7.325.2 Member Function Documentation

create()

```
static JAWSMAKO_API IJawsRendererPtr JawsMako::IJawsRenderer::create (
            const IJawsMakoPtr & jawsMako,
            const IProgressMonitorPtr & progressMonitor = IProgressMonitorPtr() )  [static]
```

Create a Jaws renderer instance.

Parameters

jawsMako	The JawsMako instance.
progressMonitor	A smart pointer to an IProgressMonitor object or NULL of no object was passed.

Returns

IJawsRendererPtr A jaws renderer instance.

render()

```
virtual IDOMImagePtr JawsMako::IJawsRenderer::render (
            const IDOMNodePtr & node,
            uint32 dpi = 72,
            uint8 depth = 8,
            const IDOMColorSpacePtr & colorSpace = IDOMColorSpacePtr(),
            const FRect & bounds = FRect(),
            bool maskToInterestingNodes = false,
            const CSpotColorNames & retainedSpotColors = CSpotColorNames(),
            const IOptionalContentPtr & optionalContent = IOptionalContentPtr(),
            eOptionalContentEvent optionalContentUsage = eOCEView )  [pure virtual]
```

Render a node to an image.

Parameters

node	The node to render.
dpi	The desired resolution.
depth	The desired depth - either 8 or 16 bits per component.
colorSpace	The desired color space. If NULL, DeviceRGB will be used. If provided, this must be an instance of one of the following colorspace types: • DeviceRGB, • sRGB, • DeviceGray, • sGray, • DeviceCMYK, • ICCBased,
bounds	The desired rendering area. If the rect is empty an appropriate size is chosen based on the input node.
maskToInterestingNodes	If true, a mask channel will be created to render only nodes that are marked as interesting. All other nodes will be masked out. Nodes may be marked as interesting by setting the IDOMNodeFlags::eNodeInteresting flag on the nodes to render.
retainedSpotColors	Used only if the color space is a three or four channel type (ie some form of RGB or CMYK color space). If provided, the renderer will provide channels in the output image for these spot colors and generate a DeviceN color space for the image as a whole. Note that the generated DeviceN color space is a best guess and is suitable for separated output only.

Parameters

optionalContent	The optional content properties of the document the content is taken from. If provided, the renderer will apply optional content rules to the content during rendering. If not provided, all content will be rendered.
optionalContentUsage	The desired usage for optional content. Ignored if #optionalContent is NULL. eOCEUnknown must not be used.

Returns

IDOMImagePtr the rendered image.

renderAntiAliased()

```
virtual IDOMImagePtr JawsMako::IJawsRenderer::renderAntiAliased (
            const IDOMNodePtr & node,
            uint32 dpi = 72,
            uint8 quality = 3,
            const FRect & bounds = FRect() )  [pure virtual]
```

Render a node to an antialiased, DeviceRGB image suitable for display, using supersampling.

Parameters

node	The node to render.
dpi	The desired resolution.
quality	The desired anti-aliasing quality, from 0 (no antialiasing) to 15 (maximum).

Returns

IDOMImagePtr the rendered image.

renderMonochrome()

```
IDOMImagePtr JawsMako::IJawsRenderer::renderMonochrome (
            const IDOMNodePtr & node,
            uint32 dpi = 72,
            const IHalftone * halftone = NULL,
            const IDOMColorSpacePtr & colorSpace = IDOMColorSpacePtr(),
            const FRect & bounds = FRect(),
            const bool useCC = false )  [inline]
```

Convenience - Render a node to a 1-bit monochrome image.

Parameters

node	The node to render.
dpi	The desired resolution.
halftone	The Halftone to use. If NULL, a 60lpi spot function will be used. If provided, this must be either a CSpotHalftone, CThresholdArrayHalftone, or CThresholdHalftone.

Parameters

colorSpace	The desired color space. If NULL, DeviceGray will be used. If provided, only DeviceGray sGray, or a single component ICC space may be used.
bounds	The desired rendering area. If the rect is empty an appropriate size is chosen based on the input node.
useCC	When this parameter is true then color correction is applied during rendering.

Returns

IDOMImagePtr the rendered image.

renderMonochromeToFrameBuffer()

```
void JawsMako::IJawsRenderer::renderMonochromeToFrameBuffer (
            const IDOMNodePtr & node,
            void * buffer,
            uint32 stride,
            uint32 destWidth,
            uint32 destHeight,
            const IHalftone * halftone,
            const IDOMColorSpacePtr colorSpace = IDOMColorSpacePtr(),
            const FRect & bounds = FRect(),
            const bool useCC = false,
            const IOptionalContentPtr & optionalContent = IOptionalContentPtr(),
            eOptionalContentEvent optionalContentUsage = eOCEView )  [inline]
```

Convenience Render a node to a 1-bit monochrome frame buffer.

Parameters

in	*node*	The node to render.
in	*buffer*	The target frame buffer.
in	*stride*	The distance in bytes between the start of one scanline in the buffer to the next scanline in y.
in	*halftone*	The Halftone to use. Must not be NULL. Spot function (CSpotHalftone) and threshold (CThresholdHalftone) supported. (Adjust the frequency of spot function so that it works in 48 dpi. For example, to achieve 60 lpi at 600dpi, set a frequency of 60 ∗ 48 / 600 = 4.8 lpi).
in	*colorSpace*	The desired color space. If NULL, DeviceGray will be used. If provided, only DeviceGray or sGray may be used.
in	*bounds*	The desired rendering area. If the rect is empty an appropriate size is chosen based on the input node.
in	*useCC*	When this parameter is true then color correction is applied during rendering.
in	*optionalContentUsage*	The desired usage for optional content. Ignored if optionalContent is NULL. eOCEUnknown must not be used.

renderScreened()

```
virtual CEDLVector< IDOMImagePtr > JawsMako::IJawsRenderer::renderScreened (
            const IDOMNodePtr & node,
```

```
            uint32 dpi,
            const IHalftone * halftone,
            const IDOMColorSpacePtr & colorSpace = IDOMColorSpacePtr(),
            const FRect & bounds = FRect(),
            const bool useCC = true,
            const IOptionalContentPtr & optionalContent = IOptionalContentPtr(),
            eOptionalContentEvent optionalContentUsage = eOCEView )  [pure virtual]
```

Render a node to a screened result, in monochrome or CMYK returning one image per colorant.

Parameters

in	*node*	The node to render.
in	*dpi*	The desired resolution.
in	*halftone*	to use. Must be provided. When rendering to a CMYK or RGB result, this must be either a CColorSpotHalftone or a CColorThresholdArrayHalftone. When rendering to a monochrome result this must be either a CSpotHalftone, CThresholdArrayHalftone, or CThresholdHalftone.
in	*colorSpace*	The desired color space. May be one of: • DeviceGray • sGray • DeviceRGB • sRGB • DeviceCMYK • One, three or four-component ICC color space
in	*bounds*	The desired rendering area. If the rect is empty an appropriate size is chosen based on the input node.
in	*useCC*	When this parameter is true then color correction is applied during rendering.
in	*optionalContentUsage*	The desired usage for optional content. Ignored if optionalContent is NULL. eOCEUnknown must not be used.

renderScreenedToFrameBuffers()

```
virtual void JawsMako::IJawsRenderer::renderScreenedToFrameBuffers (
            const IDOMNodePtr & node,
            void ** buffers,
            uint32 * strides,
            uint32 destWidth,
            uint32 destHeight,
            const IHalftone * halftone,
            const IDOMColorSpacePtr & colorSpace = IDOMColorSpacePtr(),
            const FRect & bounds = FRect(),
            const bool useCC = true,
            const IOptionalContentPtr & optionalContent = IOptionalContentPtr(),
            eOptionalContentEvent optionalContentUsage = eOCEView )  [pure virtual]
```

Render a node to a 1-bit monochrome frame buffer, a series of three 1-bit frame buffers for RGB, or four 1-bit frame buffers representing CMYK.

Parameters

in	*node*	The node to render.
in	*buffers*	The target frame buffers. Provide one for monochrome, or four for CMYK.
in	*strides*	The distance in bytes between the start of one scanline in each buffer to the next scanline in y, one per frame buffer.
in	*halftone*	to use. Must be provided. When rendering to a CMYK or RGB result, this must be either a CColorSpotHalftone or a CColorThresholdArrayHalftone. When rendering to a monochrome result this must be either a CSpotHalftone, CThresholdArrayHalftone, or CThresholdHalftone. For spot functions, adjust the frequency to assume 48 dpi. For example, to achieve 60 lpi at 600dpi, set a frequency of 60 ∗ 48 / 600 = 4.8 lpi.
in	*colorSpace*	The desired color space. May be one of: • DeviceGray • sGray • DeviceRGB • sRGB • DeviceCMYK • One, three or four-component ICC color space If NULL, DeviceGray is used.
in	*bounds*	The desired rendering area. If the rect is empty an appropriate size is chosen based on the input node.
in	*useCC*	When this parameter is true then color correction is applied during rendering.
in	*optionalContentUsage*	The desired usage for optional content. Ignored if optionalContent is NULL. eOCEUnknown must not be used.

renderSeparations()

```
virtual CEDLVector< IDOMImagePtr > JawsMako::IJawsRenderer::renderSeparations (
            const IDOMNodePtr & node,
            const uint8 depth = 8,
            const IDOMColorSpacePtr & colorSpace = IDOMColorSpacePtr(),
            uint8 antiAliased = 0,
            const FRect & bounds = FRect(),
            uint32 destWidth = 0,
            uint32 destHeight = 0,
            const CSpotColorNames & retainedSpotColors = CSpotColorNames(),
            const IOptionalContentPtr & optionalContent = IOptionalContentPtr(),
            eOptionalContentEvent optionalContentUsage = eOCEView,
            const CU8StringVect & processColorNames = CU8StringVect(),
            bool alphaGeneration = false,
            uint64 bandMemorySize = 0,
            const CSpotColorNames & ignoreSpotColors = CSpotColorNames() )  [pure virtual]
```

Render a node to a set of separated images, with optional anti-aliasing. For ease of use, the output is scaled to fit the destination size, described in pixels.

Parameters

node	The node to render.

Parameters

depth	The desired depth - either 8 or 16 bits per component.
colorSpace	The desired color space. If NULL, DeviceRGB will be used. If provided, this must be an instance of one of the following colorspace types: • DeviceRGB, • sRGB, • DeviceGray, • sGray, • DeviceCMYK, • ICCBased Multi-channel ICC based color spaces based on CMYK are supported, such as CMYKOG (Cyan Magenta Yellow Black Orange Green) hifi color profiles and the like. If such a profile is used, then processColorNames must be provided.
antiAliased	Anti-aliasing factor. If zero, anti-aliasing is not performed.
bounds	The desired rendering area. If the rect is empty an appropriate size is chosen based on the input node.
destWidth	The desired output width, in pixels. If zero, the input size is used.
destHeight	The desired output height, in pixels. If zero, the input size is used.
retainedSpotColors	Used only if the color space is a three or four channel type (ie some form of RGB or CMYK color space). If provided, the renderer will produce separate images for these colorants. A maximum of 4096 total separations are supported (process + spot + alpha).
optionalContent	The optional content properties of the document the content is taken from. If provided, the renderer will apply optional content rules to the content during rendering. If not provided, all content will be rendered.
optionalContentUsage	The desired usage for optional content. Ignored if #optionalContent is NULL. eOCEUnknown must not be used.
processColorNames	Required only if a multi-channel target profile target space is provided, ignored otherwise. The names must be specified in the order that they are generated by the profile. For example, a CMYKOK profile will normally require this to be set to "Cyan", "Magenta", "Yellow", "Black", "Orange", "Green".
alphaGeneration	Used only if the color space is Gray, RGB or CMYK based (including ICC Gray, RGB or CMYK).
bandMemorySize	Explicitly set the amount of memory that should be used for band buffers in the renderer. Useful for some situations where either very large pages or very large numbers of separations. This parameter is ignored on 32-bit platforms.
ignoreSpotColors	If provided, the renderer will ignore these colorants.

Returns

CEDLVector<IDOMImagePtr> the rendered separations. The main channels (as determined from the target space) will be first, followed by spot components (if requested) in the same order as retainedSpotColors.

renderSeparationsToFrameBuffers()

```
virtual void JawsMako::IJawsRenderer::renderSeparationsToFrameBuffers (
            const IDOMNodePtr & node,
```

```
uint8 depth,
bool hostEndian,
uint32 destWidth,
uint32 destHeight,
const IDOMColorSpacePtr & colorSpace,
CFrameBufferInfoVect & frameBuffers,
uint8 antiAliased = 0,
const FRect & bounds = FRect(),
const CSpotColorNames & retainedSpotColors = CSpotColorNames(),
const IOptionalContentPtr & optionalContent = IOptionalContentPtr(),
eOptionalContentEvent optionalContentUsage = eOCEView,
const CU8StringVect & processColorNames = CU8StringVect(),
bool alphaGeneration = false,
uint64 bandMemorySize = 0,
const CSpotColorNames & ignoreSpotColors = CSpotColorNames() )  [pure virtual]
```

As per renderSeparations(), but targeting a set of frame buffers, described by a vector CFrameBufferInfoVect.

Parameters

node	The node to render.
depth	The desired depth - either 8 or 16 bits per component.
hostEndian	Whether or not to use host-endianness for 16 bit components. If false, the PDF format is used (Big endian). If true, the host endianness (typically little endian) is used. Ignored for 8bpc.
destWidth	The desired output width, in pixels.
destHeight	The desired output height, in pixels.
colorSpace	The desired color space. If provided, this must be an instance of one of the following colorspace types: • DeviceRGB, • sRGB, • DeviceGray, • sGray, • DeviceCMYK, • ICCBased Multi-channel ICC based color spaces based on CMYK are supported, such as CMYKOG (Cyan Magenta Yellow Black Orange Green) hifi color profiles and the like. If such a profile is used, then processColorNames must be provided.
frameBuffers	A vector of frame buffers where the results will be written. There must be a CFrameBufferInfo for every channel that will be generated (including alpha if alphaGeneration is true). The row and pixel strides in the CFrameBufferInfo may be negative. The overlapping of buffers is allowed, and therefore it is possible to create a series of CFrameBufferInfo that result in interleaved output to a single block of target memory. Padding and reversal is also possible.
antiAliased	Anti-aliasing factor. If zero, anti-aliasing is not performed.
bounds	The desired rendering area. If the rect is empty an appropriate size is chosen based on the input node.
retainedSpotColors	Used only if the color space is a three or four channel type (ie some form of RGB or CMYK color space). If provided, the renderer will produce separate images for these colorants. A maximum of 4096 total separations are supported (process + spot + alpha).

Parameters

optionalContent	The optional content properties of the document the content is taken from. If provided, the renderer will apply optional content rules to the content during rendering. If not provided, all content will be rendered.
optionalContentUsage	The desired usage for optional content. Ignored if #optionalContent is NULL. eOCEUnknown must not be used.
processColorNames	Required only if a multi-channel target profile target space is provided, ignored otherwise. The names must be specified in the order that they are generated by the profile. For example, a CMYKOK profile will normally require this to be set to "Cyan", "Magenta", "Yellow", "Black", "Orange", "Green".
alphaGeneration	Used only if the color space is Gray, RGB or CMYK based (including ICC Gray, RGB or CMYK).
bandMemorySize	Explicitly set the amount of memory that should be used for band buffers in the renderer. Useful for some situations where either very large pages or very large numbers of separations. This parameter is ignored on 32-bit platforms.
ignoreSpotColors	If provided, the renderer will ignore these colorants.

renderToFrameBuffer()

```
virtual void JawsMako::IJawsRenderer::renderToFrameBuffer (
            const IDOMNodePtr & node,
            void * buffer,
            uint32 stride,
            uint32 destWidth,
            uint32 destHeight,
            const IDOMColorSpacePtr & colorSpace = IDOMColorSpacePtr(),
            const FRect & bounds = FRect(),
            const IOptionalContentPtr & optionalContent = IOptionalContentPtr(),
            eOptionalContentEvent optionalContentUsage = eOCEView )  [pure virtual]
```

Render a node to a supplied frame buffer, at 8 bits per component.

Parameters

node	The node to render.
buffer	The target frame buffer.
destWidth	The desired output width in pixels. The input content will be scaled to fit this size.
destHeight	The desired output height in pixels. The input content will be scaled to fit this size.
stride	The distance in bytes between the start of one scanline in the buffer to the next scanline in y.
colorSpace	The desired color space. If NULL, DeviceRGB will be used. If provided, this must be an instance of one of the following colorspace types: • DeviceRGB, • sRGB, • DeviceGray, • sGray, • DeviceCMYK, • ICCBased,

Parameters

bounds	The desired rendering area. If the rect is empty an appropriate size is chosen based on the input node.
optionalContent	The optional content properties of the document the content is taken from. If provided, the renderer will apply optional content rules to the content during rendering. If not provided, all content will be rendered.
optionalContentUsage	The desired usage for optional content. Ignored if #optionalContent is NULL. eOCEUnknown must not be used.

renderToFrameBufferPadAndReverse()

```
virtual void JawsMako::IJawsRenderer::renderToFrameBufferPadAndReverse (
            const IDOMNodePtr & node,
            void * buffer,
            uint32 stride,
            uint32 pixelStride,
            bool reverseComponents,
            uint32 destWidth,
            uint32 destHeight,
            const IDOMColorSpacePtr & colorSpace = IDOMColorSpacePtr(),
            const FRect & bounds = FRect(),
            const IOptionalContentPtr & optionalContent = IOptionalContentPtr(),
            eOptionalContentEvent optionalContentUsage = eOCEView )  [pure virtual]
```

As per renderToFrameBuffer, but allowing for pixel byte order and padding control.

Amongst other things this allows the use of XRGB, RGBX, XBGR or BGRX frame buffers. For example, to render to:

- XRGB: Set pixelStride to 4 and pass the address of the second frame buffer byte
- RGBX: Set pixelStride to 4 and pass the buffer address as usual
- XBGR: Set pixelStride to 4, pass the address of the second frame buffer byte, and set reverseComponents to true
- BGRX: Set pixelStride to 4, pass the buffer address as usual, and set reverseComponents to true

Note that any pad bytes will not be altered during rendering.

Parameters

node	The node to render.
buffer	The target frame buffer.
destWidth	The desired output width in pixels. The input content will be scaled to fit this size.
destHeight	The desired output height in pixels. The input content will be scaled to fit this size.
stride	The distance in bytes between the start of one scanline in the buffer to the next scanline in y.
pixelStride	The distance in bytes between one pixel in x and the next. Must be at least as large as the number of components in the color space.
reverseComponents	If true, the order of samples within a pixel will be reversed. For example, if you are rendering RGB to a BGR frame buffer, set this to true.

Parameters

colorSpace	The desired color space. If NULL, DeviceRGB will be used. If provided, this must be an instance of one of the following colorspace types: • DeviceRGB, • sRGB, • DeviceGray, • sGray, • DeviceCMYK, • ICCBased,
bounds	The desired rendering area. If the rect is empty an appropriate size is chosen based on the input node.
optionalContent	The optional content properties of the document the content is taken from. If provided, the renderer will apply optional content rules to the content during rendering. If not provided, all content will be rendered.
optionalContentUsage	The desired usage for optional content. Ignored if #optionalContent is NULL. eOCEUnknown must not be used.

setBlackPreservation()

```
virtual void JawsMako::IJawsRenderer::setBlackPreservation (
            bool preserveForText,
            bool preserveForOther )  [pure virtual]
```

Enable/control 100% black preservation.

```
100% black preservation preserves a 100% black output when rendering
when enabled for certain solid colors.

A 100% black object is an object painted with one of the following
colors:

- 0 0 0 1 in any CMYK color space (including ICC)
- 0 0 0 in any RGB color space (including ICC)
- 0 in any Gray colorspace (including ICC)
- 1 in a single channel DeviceN (aka Separation) Black color space
- 1 in a multi-channel DeviceN colorspace where only the Black channel
  is set.

If black preservation applies then the resulting color will be either
(depending on the target color space):

- c m y 1 in any CMYK colorspace (including ICC)
- 0 0 0 in any RGB colorspace (including ICC)
- 0 in any Gray colorspace (including ICC)

Note that the CMY values in a CMYK result may not be zero if color
managed white conversion would not produce pure white.

Note also that where transparency is involved this conversion only
applies when converting to the blending colorspace, and has no
further effect for further conversions.

May be enabled separately for text and vector art. Applies to only
solid color brushes (IDOMSolidColorBrush) and uncolored tiling
patterns only.
```

```
The default is set by #IColorManager::setDefaultBlackPreservation(),
which defaults to false, false. The default is applied at the time
of renderer creation.
```

Parameters

preserveForText	Whether or not to apply black preservation for text.
preserveForOther	Whether or not to apply for non-text objects.

setEnableRasterCompression()

```
virtual void JawsMako::IJawsRenderer::setEnableRasterCompression (
            bool enable )  [pure virtual]
```

Enable the use of compression for compressed output.

```
If true, a fast compression is applied to raster data produced by the renderer.
If false, the image raster data is stored uncompressed.

In general, the compression is very fast and does not introduce a large CPU cost
but reduces the storage requirements. However, it does mean that IImageFrame::readScanline()
will be slower than if the data was uncompressed.

The default is true.
```

setEnableTrueTypeNotDef()

```
virtual void JawsMako::IJawsRenderer::setEnableTrueTypeNotDef (
            bool enable )  [pure virtual]
```

Enable the use of a True Type font's /.notdef glyph in the output.

```
If true, a character that does not have a glyph or that has been linked to the /.notdef glyph
will be rendered.

If false, a character that does not have a glyph or that has been linked to the /.notdef glyph
will be ignored.

The default is true.
```

setIgnoreMatchingDeviceIntercept()

```
virtual void JawsMako::IJawsRenderer::setIgnoreMatchingDeviceIntercept (
            bool ignoreDeviceIntercept )  [pure virtual]
```

Allow intercept color spaces for a matching Device space to be ignored.

```
In the renderer, when a Device color space (DeviceRGB, DeviceCMYK, or
DeviceGray) is seen, it is interpreted according to the intercept
color space set in the IColorManager instance via the setDevice*Intercept()
routines.

This may not always be desirable. Consider the case where a CMYK ICC profile
is used as the the output color space for rendering. If the DeviceCMYK
intercept does not match that profile, then any DeviceCMYK content would be
color converted from the intercept color space to the output profile.

This behaviour is contrary to what would ordinarily be seen in a standalone
RIP. There, it is expected that if the output is a CMYK profile, then any
DeviceCMYK content would be passed through without conversion.

The same result can be achieved in Mako by using
IColorManager::setDeviceCMYKIntercept() to apply the output ICC profile as
the DeviceCMYK intercept. However, this is a global setting, and it may not
always be possible to do so.

Here, if ignoreDeviceIntercept is set, then the renderer will do the following:

- If the output is a CMYK color space, then any DeviceCMYK content will be
  interpreted as matching that space.
- If the output is an RGB color space, then any DeviceRGB content will be
  interpreted as matching that space.
- If the output is a Gray color space, then any DeviceGray content will be
  interpreted as matching that space.

The default is false.
```

The documentation for this class was generated from the following file:

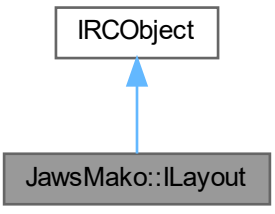
- jawsmako.h

7.326 JawsMako::ILayout Class Reference

An instance of a Layout engine that can layout and flow text into one or more frames, resulting in DOM that describes the final laid-out text.

```
#include <layout.h>
```

Inheritance diagram for JawsMako::ILayout:



Public Member Functions

- virtual void addFrame (const ILayoutFramePtr &frame)=0

  Add a frame to the list of frames into which the engine will flow text. Text will be laid out into the first frame, and any that does not fit will flow into the second, and so forth.
- virtual IDOMNodePtr layout (const CLayoutParagraphVect &paragraphs)=0

  Lay out the given paragraphs into the frames, returning a DOM representation of the result.
- virtual IDOMNodePtr layout (const ILayoutParagraphPtr &paragraph)=0

  Lay out a single paragraph.
- virtual IDOMNodePtr layout (const ILayoutRunPtr &run, ILayoutParagraph::eHorizontalAlignment horizontal↩ Alignment, double spacingAfter=0.0, double spacingBefore=0.0, double leading=1.0)=0

  Lay out a run of content.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API ILayoutPtr create (const IJawsMakoPtr &jawsMako)

  Create an instance of the layout engine.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.326.1 Detailed Description

An instance of a Layout engine that can layout and flow text into one or more frames, resulting in DOM that describes the final laid-out text.

7.326.2 Member Function Documentation

addFrame()

```
virtual void JawsMako::ILayout::addFrame (
            const ILayoutFramePtr & frame )  [pure virtual]
```

Add a frame to the list of frames into which the engine will flow text. Text will be laid out into the first frame, and any that does not fit will flow into the second, and so forth.

Parameters

frame	The frame to add.

create()

```
static JAWSMAKO_API ILayoutPtr JawsMako::ILayout::create (
            const IJawsMakoPtr & jawsMako )  [static]
```

Create an instance of the layout engine.

Parameters

jawsMako	The IJawsMako object.

Returns

ILayoutPtr The new layout engine object.

layout() [1/3]

```
virtual IDOMNodePtr JawsMako::ILayout::layout (
            const CLayoutParagraphVect & paragraphs )  [pure virtual]
```

Lay out the given paragraphs into the frames, returning a DOM representation of the result.

Parameters

paragraphs	The paragraphs to lay out.

Returns

IDOMNodePtr The resulting DOM.

layout() [2/3]

```
virtual IDOMNodePtr JawsMako::ILayout::layout (
            const ILayoutParagraphPtr & paragraph )  [pure virtual]
```

Lay out a single paragraph.

Convenience function to lay out a single paragraph into frames.

Parameters

paragraph	The paragraph to lay out.

Returns

IDOMNodePtr The resulting DOM.

layout() [3/3]

```
virtual IDOMNodePtr JawsMako::ILayout::layout (
            const ILayoutRunPtr & run,
            ILayoutParagraph::eHorizontalAlignment horizontalAlignment,
            double spacingAfter = 0.0,
            double spacingBefore = 0.0,
            double leading = 1.0 )  [pure virtual]
```

Lay out a run of content.

Convenience function to lay out a run of content into frames. Internally the run will be added to a newly created ILayoutParagraph instance.

Parameters

run	The run of content to lay out.
horizontalAlignment	The horizontal alignment to use for the ILayoutParagraph instance.
spacingAfter	The spacing after the paragraph, in default DOM units (96ths of an inch).
spacingBefore	The spacing before the paragraph, in default DOM units (96ths of an inch).
leading	The leading (line spacing).

Returns

IDOMNodePtr The resulting DOM.

The documentation for this class was generated from the following file:

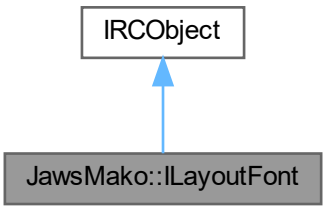
- layout.h

7.327 JawsMako::ILayoutFont Class Reference

ILayoutFont interface.

`#include <layout.h>`

Inheritance diagram for JawsMako::ILayoutFont:



Classes

- class CLayoutFontItem

  An association of a font and it's font index.

Public Types

- enum eFontStyle { eFSAny , eFSItalic = 1 , eFSRegular = 64 }

  Font style.
- enum eFontProportion { eFPAny , eFPMonoSpaced , eFPProportional }

  Font proportion.
- enum eFontSerifStyle { eFSSAny , eFSSSerif , eFSSSansSerif }

  Font serif style.

Public Member Functions

- virtual CLayoutFontVect findFonts () const =0

  Find matching fonts.
- virtual void findFont (IDOMFontOpenTypePtr &font, uint32 &fontIndex, bool exact=true) const =0

  Find matching font.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API ILayoutFontPtr create (const IJawsMakoPtr &jawsMako, const ILayoutFont↩ WeightPtr &weight=ILayoutFontWeightPtr(), eFontStyle style=eFSAny, eFontProportion proportion=eFPAny, eFontSerifStyle serifStyle=eFSSAny, uint64 codePage=UINT64_MAX)

  Create a ILayoutFont instance.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.327.1 Detailed Description

ILayoutFont interface.

A interface that enumerates system fonts with information drawn from OpenType tables to allow font selection based on characteristics such as font weight, style, proportion, serif style and code page ranges.

7.327.2 Member Enumeration Documentation

eFontProportion

`enum JawsMako::ILayoutFont::eFontProportion`

Font proportion.

Enumerator

eFPAny	Match against any proportion type.
eFPMonoSpaced	Monospace (fixed-width) font.
eFPProportional	Proportional (variable-width) font.

eFontSerifStyle

`enum JawsMako::ILayoutFont::eFontSerifStyle`

Font serif style.

Enumerator

eFSSAny	Match against any style.
eFSSSerif	Serif.
eFSSSansSerif	Sans Serif.

eFontStyle

`enum JawsMako::ILayoutFont::eFontStyle`

Font style.

Enumerator

eFSAny	Match against any style.
eFSItalic	Italic.
eFSRegular	Regular.

7.327.3 Member Function Documentation

create()

```
static JAWSMAKO_API ILayoutFontPtr JawsMako::ILayoutFont::create (
            const IJawsMakoPtr & jawsMako,
            const ILayoutFontWeightPtr & weight = ILayoutFontWeightPtr(),
            eFontStyle style = eFSAny,
            eFontProportion proportion = eFPAny,
            eFontSerifStyle serifStyle = eFSSAny,
            uint64 codePage = UINT64_MAX )  [static]
```

Create a ILayoutFont instance.

Parameters

jawsMako	The IJawsMako object.
weight	Weights used when finding fonts. When comparing weights, the usWeightClass field from the font OS/2 table will be used. The default value is NULL, which returns fonts of any weight.
style	The style to be used when finding fonts. The style will be compared with the bit values from the fsSelection field in the OS/2 table. The default value is eFSAny, which returns fonts of any style.
proportion	The proportion type used when finding fonts. The proportion type is determined from both the proportion value in the OS/2 table panose field, and the isFixedPitch field in the post table header, and they are expected to be consistent. The default value is eFPAny, which return both proportional and monospace fonts.
serifStyle	The Serif style used when finding fonts. This is determined using both the OS/2 table sFamilyClass and panose fields, and they are expected to be consistent. The default value is eFSSAny, which returns both Serif and Sans Serif fonts.
codePage	The code page character ranges used used when finding fonts. This is equivalent to the OS/2 ulCodePageRange1 and ulCodePageRange2 fields and can be used to find fonts that set specific ranges. The default value sets all bits to 1 (all code page ranges).

Returns

ILayoutFontPtr A smart pointer to the ILayoutFont object.

findFont()

```
virtual void JawsMako::ILayoutFont::findFont (
            IDOMFontOpenTypePtr & font,
            uint32 & fontIndex,
            bool exact = true ) const  [pure virtual]
```

Find matching font.

Returns the first font found to match the selection criteria, and the font index.

Parameters

font	Reference to the font to be returned, which will be set NULL if there were no matching fonts.
fontIndex	Reference to the index of the font within the font file.
exact	When true only exact font matches will be returned. When false the font determined to be the closest match will be returned. The default is true.

findFonts()

```
virtual CLayoutFontVect JawsMako::ILayoutFont::findFonts ( ) const  [pure virtual]
```

Find matching fonts.

Returns

CLayoutFontVect A vector of CLayoutFontItem.

The documentation for this class was generated from the following file:

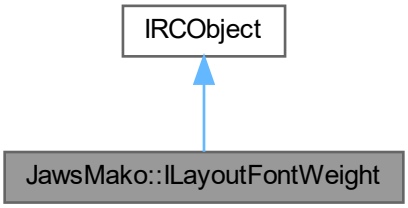
- layout.h

7.328 JawsMako::ILayoutFontWeight Class Reference

Font weights used by ILayoutFont for font selection.

```
#include <layout.h>
```

Inheritance diagram for JawsMako::ILayoutFontWeight:



Public Types

- enum eFontWeight {
  eAny = 0 , eThin = 100 , eExtraLight = 200 , eLight = 300 ,
  eSemiLight = 350 , eNormal = 400 , eMedium = 500 , eSemiBold = 600 ,
  eBold = 700 , eExtraBold = 800 , eBlack = 900 , eExtraBlack = 950 }

  Font weight.

Public Member Functions

- virtual bool match (uint16 weight) const =0

  Test for a matching font weight.
- virtual uint16 difference (uint16 weight) const =0

  Find the font weight difference.
- virtual bool equals (const ILayoutFontWeightPtr &weight) const =0

  Determines if the given weight instance is equivalent to this instance.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API ILayoutFontWeightPtr create (uint16 weight=eAny, uint16 allowedError=0)

  Create a font weight instance using a single weight value.
- static JAWSMAKO_API ILayoutFontWeightPtr createFromVect (const CUInt16Vect &weights, uint16 allowedError=0)

  Create a font weight instance using a vector of weight values.
- static JAWSMAKO_API ILayoutFontWeightPtr createFromRange (uint16 weightLow, uint16 weightHigh)

  Create a font weight instance using a range of weight values.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.328.1 Detailed Description

Font weights used by ILayoutFont for font selection.

7.328.2 Member Enumeration Documentation

eFontWeight

```
enum JawsMako::ILayoutFontWeight::eFontWeight
```

Font weight.

Enumeration of common weights intended to be used with ILayoutFontWeight::create().

Enumerator

eAny	Match against any weight.
eThin	Thin.
eExtraLight	Extra Light.
eLight	Light.
eSemiLight	Semi Light.
eNormal	Normal.
eMedium	Medium.
eSemiBold	Semi Bold.
eBold	Bold.
eExtraBold	Extra Bold.

7.328.3 Member Function Documentation

create()

```
static JAWSMAKO_API ILayoutFontWeightPtr JawsMako::ILayoutFontWeight::create (
            uint16 weight = eAny,
            uint16 allowedError = 0 )  [static]
```

Create a font weight instance using a single weight value.

Parameters

weight	The weight to use when matching. The default value of eAny will select fonts of any weight.
allowedError	Optional allowed error margin. The default value is 0.

Returns

ILayoutFontWeightPtr A smart pointer to the ILayoutFontWeight object.

createFromRange()

```
static JAWSMAKO_API ILayoutFontWeightPtr JawsMako::ILayoutFontWeight::createFromRange (
            uint16 weightLow,
            uint16 weightHigh )  [static]
```

Create a font weight instance using a range of weight values.

Parameters

weightLow	The lower weight range value to use when matching.
weightHigh	The upper weight range value to use when matching.

Returns

ILayoutFontWeightPtr A smart pointer to the ILayoutFontWeight object.

createFromVect()

```
static JAWSMAKO_API ILayoutFontWeightPtr JawsMako::ILayoutFontWeight::createFromVect (
            const CUInt16Vect & weights,
            uint16 allowedError = 0 )  [static]
```

Create a font weight instance using a vector of weight values.

Parameters

weights	The weights to use when matching.
allowedError	Optional allowed error margin. The default value is 0.

Returns

ILayoutFontWeightPtr A smart pointer to the ILayoutFontWeight object.

difference()

```
virtual uint16 JawsMako::ILayoutFontWeight::difference (
            uint16 weight ) const  [pure virtual]
```

Find the font weight difference.

For instances created with a range of weights, the return value will be a non-zero difference if the weight is outside the specified range range. For instances created with a vector of weight values, the return value will be the difference from the closest weight in the vector.

Parameters

weight	The weight to test.

Returns

uint16 The difference as an absolute value.

equals()

```
virtual bool JawsMako::ILayoutFontWeight::equals (
            const ILayoutFontWeightPtr & weight ) const  [pure virtual]
```

Determines if the given weight instance is equivalent to this instance.

Parameters

weight	The weight instance to compare with.

Returns

bool True if equal, false otherwise.

match()

```
virtual bool JawsMako::ILayoutFontWeight::match (
            uint16 weight ) const  [pure virtual]
```

Test for a matching font weight.

Parameters

weight	The weight to test. The value will be tested against all weight values, or range of weights specified at creation.

Returns

bool True if weight matched a weight value, or was within a range of weight values.

The documentation for this class was generated from the following file:

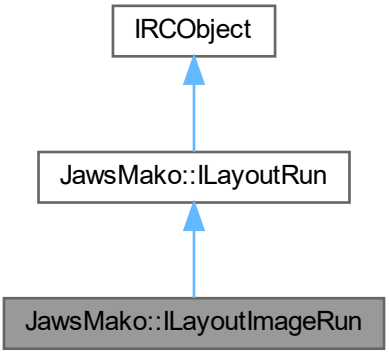
- layout.h

7.329 JawsMako::ILayoutImageRun Class Reference

A run of image content to be added to an ILayoutParagraph.

`#include <layout.h>`

Inheritance diagram for JawsMako::ILayoutImageRun:



Public Member Functions

- virtual const IDOMImagePtr & getImage () const =0

  Get the image.
- virtual double getWidth () const =0

  Get the width of the image.
- virtual double getHeight () const =0

  Get the height of the image.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API ILayoutImageRunPtr create (const IJawsMakoPtr &jawsMako, const IDOMImagePtr &image, double width=0.0, double height=0.0)

  Create an image run.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.329.1 Detailed Description

A run of image content to be added to an ILayoutParagraph.

7.329.2 Member Function Documentation

create()

```
static JAWSMAKO_API ILayoutImageRunPtr JawsMako::ILayoutImageRun::create (
            const IJawsMakoPtr & jawsMako,
            const IDOMImagePtr & image,
            double width = 0.0,
            double height = 0.0 )  [static]
```

Create an image run.

Parameters

jawsMako	The IJawsMako object.
image	The desired image for the run.
width	The desired image width. The default value of 0 will scale the image uniformly with the height. When both width and height are set to 0 the image width and height will be used.
height	The desired image height. The default value of 0 will scale the image uniformly with the width. When both width and height are set to 0 the image width and height will be used.

Returns

ILayoutImageRunPtr The new run.

getHeight()

```
virtual double JawsMako::ILayoutImageRun::getHeight ( ) const  [pure virtual]
```

Get the height of the image.

Returns

The height of the image.

getImage()

```
virtual const IDOMImagePtr & JawsMako::ILayoutImageRun::getImage ( ) const  [pure virtual]
```

Get the image.

Returns

IDOMImagePtr The image.

getWidth()

```
virtual double JawsMako::ILayoutImageRun::getWidth ( ) const  [pure virtual]
```

Get the width of the image.

Returns

The width of the image.

The documentation for this class was generated from the following file:

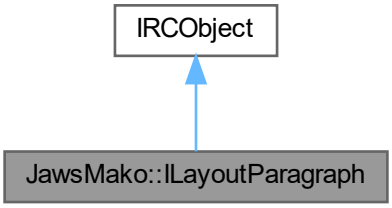
- layout.h

7.330 JawsMako::ILayoutParagraph Class Reference

A paragraph, consisting of a number of runs of content.

```
#include <layout.h>
```

Inheritance diagram for JawsMako::ILayoutParagraph:



Public Types

- enum eHorizontalAlignment { eHALeft , eHACenter , eHARight , eHAJustified }

  Horizontal alignment or justification for the paragraph.

Public Member Functions

- virtual ILayoutParagraphPtr clone () const =0

  Clone the paragraph.
- virtual void addRun (const ILayoutRunPtr &run)=0

  Add a run to the end of the paragraph.
- virtual const CLayoutRunVect & getRuns () const =0

  Get the runs present in this paragraph.
- virtual eHorizontalAlignment getHorizontalAlignment () const =0

  Get the horizontal alignment for this paragraph.
- virtual void setSpacing (double spacing, bool before=false)=0

  Set the paragraph spacing.
- virtual double getSpacing (bool before=false) const =0

  Get the paragraph spacing.
- virtual void setLeading (double leading)=0

  Set the leading (line spacing).
- virtual double getLeading () const =0

  Get the leading (line spacing).
- virtual void setDefaultFont (const ILayoutFontPtr &defaultFont, bool exact=true)=0

  Set an ILayoutFont instance to select a default font for the the paragraph, or nullptr to clear the default.
- virtual void setDefaultFont (const IDOMFontOpenTypePtr &defaultFont, uint32 defaultFontIndex)=0

  Set an the default font for the the paragraph, or nullptr to clear the default.
- virtual const IDOMFontOpenTypePtr & getDefaultFont () const =0

  Get the default font for the paragraph.
- virtual uint32 getDefaultFontIndex () const =0

  Get the font index within the default font file.
- virtual void setDefaultFontSize (double defaultFontSize)=0

  Set the default font size of the paragraph.
- virtual double getDefaultFontSize () const =0

  Get the default font size on DOM units.
- virtual void setDefaultColor (const IDOMColorPtr &defaultColor)=0

  Set a default color for the paragraph, or nullptr to clear the default.
- virtual const IDOMColorPtr & getDefaultColor () const =0

  Get the default color for the paragraph.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API ILayoutParagraphPtr create (eHorizontalAlignment horizontalAlignment=eHALeft, double spacingAfter=0.0, double spacingBefore=0.0, double leading=1.0, const IDOMColorPtr &defaultColor=IDOMColorPtr(), const ILayoutFontPtr &defaultFont=ILayoutFontPtr(), bool exact=true, double defaultFontSize=-1.0)

  Create a paragraph.
- static JAWSMAKO_API ILayoutParagraphPtr create (eHorizontalAlignment horizontalAlignment, double spacingAfter, double spacingBefore, double leading, const IDOMColorPtr &defaultColor, const IDOMFontOpenTypePtr &defaultFont, uint32 defaultFontIndex=0, double defaultFontSize=-1.0)

  Create a paragraph.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.330.1 Detailed Description

A paragraph, consisting of a number of runs of content.

7.330.2 Member Enumeration Documentation

eHorizontalAlignment

```
enum JawsMako::ILayoutParagraph::eHorizontalAlignment
```

Horizontal alignment or justification for the paragraph.

Enumerator

eHALeft	Align the content to the left boundary of the enclosing frame.
eHACenter	Align the content to the center of the enclosing frame.
eHARight	Align the content to the right boundary of the enclosing frame.
eHAJustified	Justify the content left and right.

7.330.3 Member Function Documentation

addRun()

```
virtual void JawsMako::ILayoutParagraph::addRun (
            const ILayoutRunPtr & run )  [pure virtual]
```

Add a run to the end of the paragraph.

Parameters

run	The run to add.

clone()

```
virtual ILayoutParagraphPtr JawsMako::ILayoutParagraph::clone ( ) const  [pure virtual]
```

Clone the paragraph.

Note that this will make a shallow clone of any runs added to the paragraph.

Returns

ILayoutParagraphPtr A smart pointer to the cloned ILayoutParagraph object.

create() [1/2]

```
static JAWSMAKO_API ILayoutParagraphPtr JawsMako::ILayoutParagraph::create (
            eHorizontalAlignment horizontalAlignment,
            double spacingAfter,
            double spacingBefore,
            double leading,
            const IDOMColorPtr & defaultColor,
            const IDOMFontOpenTypePtr & defaultFont,
            uint32 defaultFontIndex = 0,
            double defaultFontSize = -1.0 )  [static]
```

Create a paragraph.

Parameters

horizontalAlignment	The horizontal alignment for this paragraph.
spacingAfter	The spacing after the paragraph, in default DOM units (96ths of an inch).
spacingBefore	The spacing before the paragraph, in default DOM units (96ths of an inch).
leading	The leading (line spacing).
defaultColor	The default color to use if an enclosed run does not specify a color. If null, a DeviceGray black will be used.
defaultFont	An IDOMOpenTypeFont instance used as a default if a if an enclosed run does not specify a font.
defaultFontIndex	The index of the font within the default font file. Required if the default font is specified and is in a TrueType Collection, ignored otherwise.
defaultFontSize	The default font size to use if an enclosed font does not specify a font size. Optional.

Returns

ILayoutParagraphPtr The newly created paragraph.

create() [2/2]

```
static JAWSMAKO_API ILayoutParagraphPtr JawsMako::ILayoutParagraph::create (
            eHorizontalAlignment horizontalAlignment = eHALeft,
            double spacingAfter = 0.0,
            double spacingBefore = 0.0,
            double leading = 1.0,
            const IDOMColorPtr & defaultColor = IDOMColorPtr(),
            const ILayoutFontPtr & defaultFont = ILayoutFontPtr(),
            bool exact = true,
            double defaultFontSize = -1.0 )  [static]
```

Create a paragraph.

Parameters

horizontalAlignment	The horizontal alignment for this paragraph.
spacingAfter	The spacing after the paragraph, in default DOM units (96ths of an inch).
spacingBefore	The spacing before the paragraph, in default DOM units (96ths of an inch).
leading	The leading (line spacing).
defaultColor	The default color to use if an enclosed run does not specify a color. If null, a DeviceGray black will be used.
defaultFont	An ILayoutFont instance used to select a font to use as a default if an enclosed run does not specify a font. Optional.
exact	When true only exact font matches will be returned when selecting the default font (if provided). When false the font determined to be the closest match will be returned. The default is true.
defaultFontSize	The default font size to use if an enclosed font does not specify a font size. Optional.

Returns

ILayoutParagraphPtr The newly created paragraph.

getDefaultColor()

```
virtual const IDOMColorPtr & JawsMako::ILayoutParagraph::getDefaultColor ( ) const  [pure
virtual]
```

Get the default color for the paragraph.

Returns

IDOMColorPtr The default color.

getDefaultFont()

```
virtual const IDOMFontOpenTypePtr & JawsMako::ILayoutParagraph::getDefaultFont ( ) const  [pure
virtual]
```

Get the default font for the paragraph.

Returns

IDOMFontOpenTypePtr The font.

getDefaultFontIndex()

```
virtual uint32 JawsMako::ILayoutParagraph::getDefaultFontIndex ( ) const  [pure virtual]
```

Get the font index within the default font file.

Returns

uint32 The font index.

getDefaultFontSize()

```
virtual double JawsMako::ILayoutParagraph::getDefaultFontSize ( ) const  [pure virtual]
```

Get the default font size on DOM units.

Returns

double The font size, or a negative value if no default has been set.

getHorizontalAlignment()

```
virtual eHorizontalAlignment JawsMako::ILayoutParagraph::getHorizontalAlignment ( ) const
[pure virtual]
```

Get the horizontal alignment for this paragraph.

Returns

eHorizontalAlignment The horizontal alignment.

getLeading()

```
virtual double JawsMako::ILayoutParagraph::getLeading ( ) const  [pure virtual]
```

Get the leading (line spacing).

Returns

double The leading value as a fraction of the default, for example 0.5 for half of the default leading.

getRuns()

```
virtual const CLayoutRunVect & JawsMako::ILayoutParagraph::getRuns ( ) const  [pure virtual]
```

Get the runs present in this paragraph.

Returns

CLayoutRunVect The runs.

getSpacing()

```
virtual double JawsMako::ILayoutParagraph::getSpacing (
            bool before = false ) const  [pure virtual]
```

Get the paragraph spacing.

Parameters

before	true to set spacing before the paragraph, or false after. The default is false.

Returns

double The spacing in default DOM units (96ths of an inch).

setDefaultColor()

```
virtual void JawsMako::ILayoutParagraph::setDefaultColor (
            const IDOMColorPtr & defaultColor )  [pure virtual]
```

Set a default color for the paragraph, or nullptr to clear the default.

Parameters

defaultColor	The default color to use. Optional.

setDefaultFont() [1/2]

```
virtual void JawsMako::ILayoutParagraph::setDefaultFont (
            const IDOMFontOpenTypePtr & defaultFont,
            uint32 defaultFontIndex )  [pure virtual]
```

Set an the default font for the the paragraph, or nullptr to clear the default.

Parameters

defaultFont	The font to use as a default if an enclosed run does not specify a font. Optional.
defaultFontIndex	The index of the font within the font file. Required if the font is specified and is in a TrueType Collection, ignored otherwise.

setDefaultFont() [2/2]

```
virtual void JawsMako::ILayoutParagraph::setDefaultFont (
            const ILayoutFontPtr & defaultFont,
            bool exact = true )  [pure virtual]
```

Set an ILayoutFont instance to select a default font for the the paragraph, or nullptr to clear the default.

Parameters

defaultFont	An ILayoutFont instance used to select a font to use as a default if an enclosed run does not specify a font. Optional.
exact	When true only exact font matches will be returned when selecting the default font (if provided). When false the font determined to be the closest match will be returned. The default is true.

setDefaultFontSize()

```
virtual void JawsMako::ILayoutParagraph::setDefaultFontSize (
            double defaultFontSize )  [pure virtual]
```

Set the default font size of the paragraph.

Parameters

defaultFontSize	The default font size to use if an enclosed font does not specify a font size. Set a negative value to remove the setting.

setLeading()

```
virtual void JawsMako::ILayoutParagraph::setLeading (
            double leading )  [pure virtual]
```

Set the leading (line spacing).

Parameters

leading	The leading, which must be greater than or equal to zero. The value is a fraction of the default leading, for example a value of 0.5 would set leading to half of the default, 2 to double the default etc. The default is 1.

setSpacing()

```
virtual void JawsMako::ILayoutParagraph::setSpacing (
            double spacing,
            bool before = false )  [pure virtual]
```

Set the paragraph spacing.

Parameters

spacing	The spacing in default DOM units (96ths of an inch).
before	true to set spacing before the paragraph, or false after. The default is false.

The documentation for this class was generated from the following file:

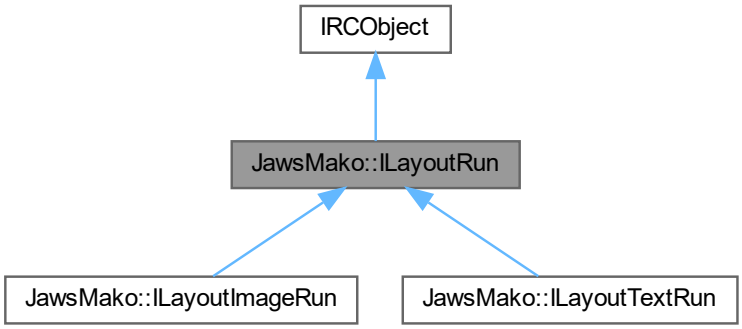
- layout.h

7.331 JawsMako::ILayoutRun Class Reference

A run of content to be added to an ILayoutParagraph.

```
#include <layout.h>
```

Inheritance diagram for JawsMako::ILayoutRun:



Additional Inherited Members

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.331.1 Detailed Description

A run of content to be added to an ILayoutParagraph.

The documentation for this class was generated from the following file:

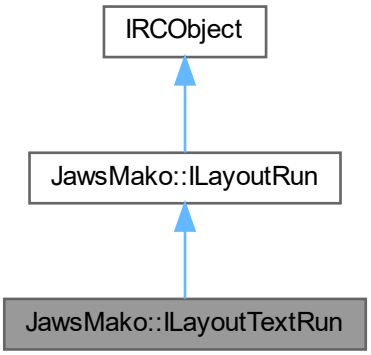
- layout.h

7.332 JawsMako::ILayoutTextRun Class Reference

A run of text content to be added to an ILayoutParagraph.

`#include <layout.h>`

Inheritance diagram for JawsMako::ILayoutTextRun:



Public Member Functions

- virtual const U8String & getText () const =0
  Get the text in the run as UTF-8.
- virtual const IDOMFontOpenTypePtr & getFont () const =0
  Get the font.
- virtual uint32 getFontIndex () const =0
  Get the font index within the font file.
- virtual double getSize () const =0
  Get the em size of the font in default DOM units (96ths of an inch)
- virtual void setColor (const IDOMColorPtr &color)=0
  Set the text color.
- virtual const IDOMColorPtr & getColor () const =0
  Get the text color.
- virtual void setLetterSpacing (double letterSpacing)=0
  Set the letter spacing.
- virtual double getLetterSpacing () const =0
  Get the letter spacing.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0
  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0
  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0
  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API ILayoutTextRunPtr create (const U8String &text, const IDOMFontOpenTypePtr &font, uint32 fontIndex, double size, const IDOMColorPtr &color=IDOMColorPtr(), double letterSpacing=0.0)

  Create a text run.
- static JAWSMAKO_API ILayoutTextRunPtr create (const String &text, const IDOMFontOpenTypePtr &font, uint32 fontIndex, double size, const IDOMColorPtr &color=IDOMColorPtr(), double letterSpacing=0.0)

  Create a text run.
- static JAWSMAKO_API ILayoutTextRunPtr create (const U8String &text, const ILayoutFontPtr &font, double size, bool exact=true, const IDOMColorPtr &color=IDOMColorPtr(), double letterSpacing=0.0)

  Create a text run.
- static JAWSMAKO_API ILayoutTextRunPtr create (const String &text, const ILayoutFontPtr &font, double size, bool exact=true, const IDOMColorPtr &color=IDOMColorPtr(), double letterSpacing=0.0)

  Create a text run.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.332.1 Detailed Description

A run of text content to be added to an ILayoutParagraph.

7.332.2 Member Function Documentation

create() [1/4]

```
static JAWSMAKO_API ILayoutTextRunPtr JawsMako::ILayoutTextRun::create (
            const String & text,
            const IDOMFontOpenTypePtr & font,
            uint32 fontIndex,
            double size,
            const IDOMColorPtr & color = IDOMColorPtr(),
            double letterSpacing = 0.0 )  [static]
```

Create a text run.

Parameters

text	The text for the run as a wide character Unicode string.
font	The desired font for the run. If nullptr, the paragraph default font (which must be set in this circumstance) will be used.
fontIndex	The index of the font within the font file. Required if the font is specified and is in a TrueType Collection, ignored otherwise.
size	The desired em size of the text, in default DOM units (96ths of an inch). If negative, the default paragraph font size (which must be set in this circumstance) will be used.
color	The color to use for the text. If null, the paragraph default font color will be used.
letterSpacing	The desired letter spacing, in ems. A positive value expands, and a negative value condenses the spacing between letters.

Returns

ILayoutTextRunPtr The new run.

create() [2/4]

```
static JAWSMAKO_API ILayoutTextRunPtr JawsMako::ILayoutTextRun::create (
            const String & text,
            const ILayoutFontPtr & font,
            double size,
            bool exact = true,
            const IDOMColorPtr & color = IDOMColorPtr(),
            double letterSpacing = 0.0 )  [static]
```

Create a text run.

Will throw an exception if the supplied ILayoutFont fails to select a font.

Parameters

text	The text for the run as a wide character Unicode string.
font	An ILayoutFont instance used to select a font for the run. If nullptr, the paragraph default font (which must be set) will be used.
size	The desired em size of the text, in default DOM units (96ths of an inch). If negative, the default paragraph font size (which must be set in this circumstance) will be used.
exact	When true only exact font matches will be returned when selecting fonts. When false the font determined to be the closest match will be returned. The default is true.
color	The color to use for the text.
letterSpacing	The desired letter spacing, in ems. A positive value expands, and a negative value condenses the spacing between letters.

Returns

ILayoutTextRunPtr The new run.

create() [3/4]

```
static JAWSMAKO_API ILayoutTextRunPtr JawsMako::ILayoutTextRun::create (
            const U8String & text,
            const IDOMFontOpenTypePtr & font,
            uint32 fontIndex,
            double size,
            const IDOMColorPtr & color = IDOMColorPtr(),
            double letterSpacing = 0.0 )  [static]
```

Create a text run.

Parameters

text	The text for the run as UTF-8.
font	The desired font for the run. If nullptr, the paragraph default font (which must be set) will be used.

Parameters

fontIndex	The index of the font within the font file. Required if the font is specified and is in a TrueType Collection, ignored otherwise.
size	The desired em size of the text, in default DOM units (96ths of an inch).
color	The color to use for the text.
letterSpacing	The desired letter spacing, in ems. A positive value expands, and a negative value condenses the spacing between letters.

Returns

ILayoutTextRunPtr The new run.

create() [4/4]

```
static JAWSMAKO_API ILayoutTextRunPtr JawsMako::ILayoutTextRun::create (
            const U8String & text,
            const ILayoutFontPtr & font,
            double size,
            bool exact = true,
            const IDOMColorPtr & color = IDOMColorPtr(),
            double letterSpacing = 0.0 )  [static]
```

Create a text run.

Will throw an exception if the supplied ILayoutFont fails to select a font.

Parameters

text	The text for the run as UTF-8.
font	An ILayoutFont instance used to select a font for the run. If nullptr, the paragraph default font (which must be set) will be used.
size	The desired em size of the text, in default DOM units (96ths of an inch). If negative, the default paragraph font size (which must be set in this circumstance) will be used.
exact	When true only exact font matches will be returned when selecting fonts. When false the font determined to be the closest match will be returned. The default is true.
color	The color to use for the text.
letterSpacing	The desired letter spacing, in ems. A positive value expands, and a negative value condenses the spacing between letters.

Returns

ILayoutTextRunPtr The new run.

getColor()

```
virtual const IDOMColorPtr & JawsMako::ILayoutTextRun::getColor ( ) const  [pure virtual]
```

Get the text color.

Returns

The text color, or NULL if no color was set.

getFont()

```
virtual const IDOMFontOpenTypePtr & JawsMako::ILayoutTextRun::getFont ( ) const  [pure virtual]
```

Get the font.

Returns

IDOMFontOpenTypePtr The font. NULL if not font is set.

getFontIndex()

```
virtual uint32 JawsMako::ILayoutTextRun::getFontIndex ( ) const  [pure virtual]
```

Get the font index within the font file.

Returns

uint32 The font index.

getLetterSpacing()

```
virtual double JawsMako::ILayoutTextRun::getLetterSpacing ( ) const  [pure virtual]
```

Get the letter spacing.

Returns

The letter spacing.

getSize()

```
virtual double JawsMako::ILayoutTextRun::getSize ( ) const  [pure virtual]
```

Get the em size of the font in default DOM units (96ths of an inch)

Returns

The size of the font.

getText()

```
virtual const U8String & JawsMako::ILayoutTextRun::getText ( ) const  [pure virtual]
```

Get the text in the run as UTF-8.

Returns

U8String The text.

setColor()

```
virtual void JawsMako::ILayoutTextRun::setColor (
            const IDOMColorPtr & color )  [pure virtual]
```

Set the text color.

Parameters

color	The text color.

setLetterSpacing()

```
virtual void JawsMako::ILayoutTextRun::setLetterSpacing (
            double letterSpacing )  [pure virtual]
```

Set the letter spacing.

Parameters

letterSpacing	The letter spacing, in ems.

The documentation for this class was generated from the following file:

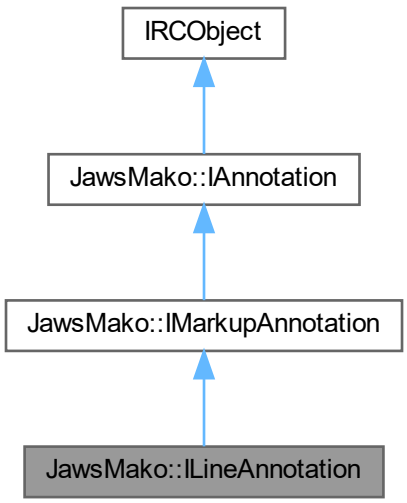
- layout.h

7.333 JawsMako::ILineAnnotation Class Reference

An interface class for a line annotation. It is intended that future releases of JawsMako will extend this interface.

```
#include <interactive.h>
```

Inheritance diagram for JawsMako::ILineAnnotation:



Public Member Functions

- virtual void getLineEndpoints (FPoint &start, FPoint &end) const =0

  Get the line end points The points are relative to the annotation rect.
- virtual void setLineEndpoints (const FPoint &start, const FPoint &end)=0

  Set the line end points. The points are relative to the annotation rect.
- virtual float getLeaderLineLength () const =0

  Get the length of the leader line.
- virtual float getLeaderLineExtensionsLength () const =0

  Get the length of the leader line extensions.
- virtual float getLeaderLineOffset () const =0

  Get the leader line offset.
- virtual void getCaptionOffset (float &xOffset, float &yOffset) const =0

  Get the caption offset.
- virtual IDOMColorPtr getInteriorColor () const =0

  Get the interior color of the fill used for the line endings.
- virtual void setInteriorColor (const IDOMColorPtr &color)=0

  Set the interior color to be used to fill the line endings.

Public Member Functions inherited from JawsMako::IMarkupAnnotation

- virtual U8String getAuthor () const =0

  Get the Author of the markup annotation. This is the "T" entry in the annotation, which by convention is the author.
- virtual void setAuthor (const U8String &author)=0

  Set the Author of the markup annotation. This is the "T" entry in the annotation, which by convention is the author.
- virtual IEDLTimePtr getCreationTime () const =0

  Get the creation date and time of the annotation, if present.
- virtual void setCreationTime (const IEDLTimePtr &creationTime)=0

  Set the creation date and time of the annotation.
- virtual float getOpacity () const =0

  Get the opacity of the markup annotation.
- virtual void setOpacity (float opacity)=0

  Set the opacity of the markup annotation.
- virtual IAnnotationReferencePtr getPopupReference () const =0

  Get a reference to the popup, if present.
- virtual void setPopup (const IAnnotationReferencePtr &popupReference)=0

  Set or clear the popup, by reference.
- virtual void setPopup (const IPopupAnnotationPtr &popup)=0

  Set a reference to a popup, if present.
- virtual IAnnotationAppearancePtr createNormalAppearance () const =0

  Create a basic appearance given the current annotation's parameters. This can then be installed by using IAnnotation::addAppearance().

Public Member Functions inherited from JawsMako::IAnnotation

- virtual eAnnotationType getType () const =0

  Get the type of the annotation.
- virtual const FRect & getRect () const =0

  Get the rect in which the appearances should be displayed.
- virtual void setRect (const FRect &rect)=0

  Set the rect in which the appearances should be displayed.
- virtual U8String getContents () const =0

  Get the Contents entry in UTF-8.
- virtual void setContents (const U8String &contents)=0

  Set the Contents entry in UTF-8.
- virtual IDOMColorPtr getColor () const =0

  Get the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.
- virtual void setColor (const IDOMColorPtr &color)=0

  Set the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.
- virtual IEDLTimePtr getModificationTime () const =0

  Get the Modification date and time of the annotation, if present.
- virtual void setModificationTime (const IEDLTimePtr &modificationTime)=0

  Set the Modification date and time of the annotation.
- virtual CAnnotationBorder getBorder () const =0

  Get the annotation's border. See CAnnotationBorder for details.
- virtual void setBorder (const CAnnotationBorder &border)=0

  Set the annotation's border.
- virtual uint32 getFlags () const =0

  Get the annotation flags. To interpret these flags please see section 8.4.2 "Annotation Flags" in the PDF 1.7 specification.
- virtual void setFlags (uint32 flags)=0

  Set the annotation flags. Please see section 8.4.2 "Annotation Flags" for details about these flags.
- virtual void rotate (uint16 rotate, double pageWidth, double pageHeight)=0

  Rotate the annotation clockwise as if the page was rotated by the same amount.
- virtual CAnnotationAppearanceVect getAppearances () const =0

  Return all the annotation appearances in a vector.
- virtual void **removeAppearances** ()=0

  Remove all annotation appearances.
- virtual U8String getState () const =0

  Get the current annotation state.
- virtual void setState (const U8String &state)=0

  Set the current annotation state.
- virtual IAnnotationAppearancePtr getAppearance (eAppearanceUsage usage, const U8String &state=U8String()) const =0

  Fetch the annotation appearance that should be used for the given annotation usage and state according to the following rules:
- virtual void addAppearance (const IAnnotationAppearancePtr &appearance)=0

  Add or replace an appearance.
- virtual bool hasNormalAppearance () const =0

  Does the annotation have a normal appearance? Convenience utility function.
- virtual IAnnotationPtr clone () const =0

  Clone the annotation. This is a deep clone. The annotation reference will remain the same.
- virtual bool matchesReference (const IAnnotationReferencePtr &reference) const =0

  Does this annotation match the given IAnnotationReference?
- virtual IAnnotationReferencePtr getReference () const =0

  Get a reference that can be used to refer to this annotation.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Public Types inherited from JawsMako::IAnnotation

- enum eAnnotationType {
  eAT3D , eATCaret , eATCircle , eATFileAttachment ,
  eATFreeText , eATHighlight , eATInk , eATLine ,
  eATLink , eATMovie , eATPolygon , eATPolyLine ,
  eATPopup , eATPrinterMark , eATProjection , eATRedact ,
  eATRichMedia , eATScreen , eATSound , eATSquare ,
  eATSquiggly , eATStamp , eATStrikeOut , eATText ,
  eATTrapNet , eATUnderline , eATWatermark , eATWidget ,
  eATOther }

  Types of annotations, listed with the earliest version of PDF that supported them.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.333.1 Detailed Description

An interface class for a line annotation. It is intended that future releases of JawsMako will extend this interface.

7.333.2 Member Function Documentation

getCaptionOffset()

```
virtual void JawsMako::ILineAnnotation::getCaptionOffset (
            float & xOffset,
            float & yOffset ) const  [pure virtual]
```

Get the caption offset.

Parameters

xOffset	The xOffset value
yOffset	The yOffset value

getInteriorColor()

```
virtual IDOMColorPtr JawsMako::ILineAnnotation::getInteriorColor ( ) const  [pure virtual]
```

Get the interior color of the fill used for the line endings.

Returns

IDOMColorPtr The fill color. NULL is returned if there is no such color

getLeaderLineExtensionsLength()

```
virtual float JawsMako::ILineAnnotation::getLeaderLineExtensionsLength ( ) const  [pure virtual]
```

Get the length of the leader line extensions.

Returns

float The leader line extensions length

getLeaderLineLength()

```
virtual float JawsMako::ILineAnnotation::getLeaderLineLength ( ) const  [pure virtual]
```

Get the length of the leader line.

Returns

float The leader line length

getLeaderLineOffset()

```
virtual float JawsMako::ILineAnnotation::getLeaderLineOffset ( ) const  [pure virtual]
```

Get the leader line offset.

Returns

float The leader line offset

getLineEndpoints()

```
virtual void JawsMako::ILineAnnotation::getLineEndpoints (
            FPoint & start,
            FPoint & end ) const  [pure virtual]
```

Get the line end points The points are relative to the annotation rect.

Parameters

start	Start point
end	End point

setInteriorColor()

```
virtual void JawsMako::ILineAnnotation::setInteriorColor (
            const IDOMColorPtr & color )  [pure virtual]
```

Set the interior color to be used to fill the line endings.

Parameters

color	The color definition, or pass NULL to remove the color

setLineEndpoints()

```
virtual void JawsMako::ILineAnnotation::setLineEndpoints (
            const FPoint & start,
            const FPoint & end )  [pure virtual]
```

Set the line end points. The points are relative to the annotation rect.

Parameters

start	Start point
end	End point

The documentation for this class was generated from the following file:

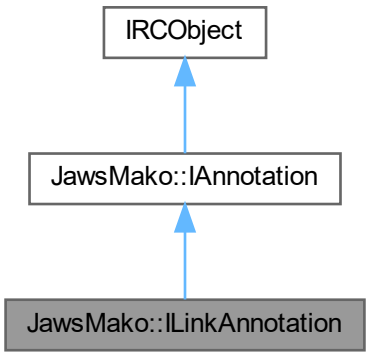
- interactive.h

7.334 JawsMako::ILinkAnnotation Class Reference

A generic interface class for a link annotation It is intended that future releases of JawsMako will extend this interface.

```
#include <interactive.h>
```

Inheritance diagram for JawsMako::ILinkAnnotation:



Public Member Functions

- virtual CQuadPointVect getQuadPoints () const =0

  Get the link annotation's quad points if present. The points are relative to the annotation rect.
- virtual void setQuadPoints (const CQuadPointVect &quadPoints)=0

  Set the link annotation's quad points.
- virtual void setActionOrDest (const IPDFObjectPtr &actionOrDest)=0

  Set the Action dictionary or Dest object for this annotation.
- virtual IPDFObjectPtr getActionOrDest () const =0

  Get the link annotation's Action or Dest object. An Action will be an IPDFDictionary object while a Dest can be IPDFArray, IPDFName, or IPDFString.
- virtual void setHighlightMode (const IPDFNamePtr &highlightMode)=0

  Set the Highlight mode for this annotation.
- virtual IPDFNamePtr getHighlightMode () const =0

  Get the link annotation's Highlight mode object.
- virtual void setPA (const IPDFDictionaryPtr &pa)=0

  Set the previous Action object for this annotation.
- virtual IPDFDictionaryPtr getPA () const =0

  Get the link annotation's previous Action object.

Public Member Functions inherited from JawsMako::IAnnotation

- virtual eAnnotationType getType () const =0

  Get the type of the annotation.
- virtual const FRect & getRect () const =0

  Get the rect in which the appearances should be displayed.
- virtual void setRect (const FRect &rect)=0

  Set the rect in which the appearances should be displayed.
- virtual U8String getContents () const =0

  Get the Contents entry in UTF-8.
- virtual void setContents (const U8String &contents)=0

Set the Contents entry in UTF-8.

- virtual IDOMColorPtr getColor () const =0

  Get the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.
- virtual void setColor (const IDOMColorPtr &color)=0

  Set the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.
- virtual IEDLTimePtr getModificationTime () const =0

  Get the Modification date and time of the annotation, if present.
- virtual void setModificationTime (const IEDLTimePtr &modificationTime)=0

  Set the Modification date and time of the annotation.
- virtual CAnnotationBorder getBorder () const =0

  Get the annotation's border. See CAnnotationBorder for details.
- virtual void setBorder (const CAnnotationBorder &border)=0

  Set the annotation's border.
- virtual uint32 getFlags () const =0

  Get the annotation flags. To interpret these flags please see section 8.4.2 "Annotation Flags" in the PDF 1.7 specification.
- virtual void setFlags (uint32 flags)=0

  Set the annotation flags. Please see section 8.4.2 "Annotation Flags" for details about these flags.
- virtual void rotate (uint16 rotate, double pageWidth, double pageHeight)=0

  Rotate the annotation clockwise as if the page was rotated by the same amount.
- virtual CAnnotationAppearanceVect getAppearances () const =0

  Return all the annotation appearances in a vector.
- virtual void **removeAppearances** ()=0

  Remove all annotation appearances.
- virtual U8String getState () const =0

  Get the current annotation state.
- virtual void setState (const U8String &state)=0

  Set the current annotation state.
- virtual IAnnotationAppearancePtr getAppearance (eAppearanceUsage usage, const U8String &state=U8String()) const =0

  Fetch the annotation appearance that should be used for the given annotation usage and state according to the following rules:
- virtual void addAppearance (const IAnnotationAppearancePtr &appearance)=0

  Add or replace an appearance.
- virtual bool hasNormalAppearance () const =0

  Does the annotation have a normal appearance? Convenience utility function.
- virtual IAnnotationPtr clone () const =0

  Clone the annotation. This is a deep clone. The annotation reference will remain the same.
- virtual bool matchesReference (const IAnnotationReferencePtr &reference) const =0

  Does this annotation match the given IAnnotationReference?
- virtual IAnnotationReferencePtr getReference () const =0

  Get a reference that can be used to refer to this annotation.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API ILinkAnnotationPtr create (const IJawsMakoPtr &jawsMako, const FRect &rect, const IDOMColorPtr &color, const IPDFObjectPtr &actionOrDest=IPDFObjectPtr(), const IPDFNamePtr &highlightMode=IPDFNamePtr(), const IPDFDictionaryPtr &pa=IPDFDictionaryPtr())

  Create a link annotation with the given parameters.

Additional Inherited Members

Public Types inherited from JawsMako::IAnnotation

- enum eAnnotationType {
  eAT3D , eATCaret , eATCircle , eATFileAttachment ,
  eATFreeText , eATHighlight , eATInk , eATLine ,
  eATLink , eATMovie , eATPolygon , eATPolyLine ,
  eATPopup , eATPrinterMark , eATProjection , eATRedact ,
  eATRichMedia , eATScreen , eATSound , eATSquare ,
  eATSquiggly , eATStamp , eATStrikeOut , eATText ,
  eATTrapNet , eATUnderline , eATWatermark , eATWidget ,
  eATOther }

  Types of annotations, listed with the earliest version of PDF that supported them.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.334.1 Detailed Description

A generic interface class for a link annotation It is intended that future releases of JawsMako will extend this interface.

7.334.2 Member Function Documentation

create()

```
static JAWSMAKO_API ILinkAnnotationPtr JawsMako::ILinkAnnotation::create (
            const IJawsMakoPtr & jawsMako,
            const FRect & rect,
            const IDOMColorPtr & color,
            const IPDFObjectPtr & actionOrDest = IPDFObjectPtr(),
            const IPDFNamePtr & highlightMode = IPDFNamePtr(),
            const IPDFDictionaryPtr & pa = IPDFDictionaryPtr() )  [static]
```

Create a link annotation with the given parameters.

Parameters

jawsMako	The JawsMako instance
rect	The annotations bounds. Must not be empty.
color	Optional; The desired annotation color. If NULL, no line will be visible.
actionOrDest	Optional; The Action dictionary or Dest object to be invoked for this annotation. If NULL, no
	action is performed.
highlightMode	Optional; The highlight mode for the annotation when the annotation is clicked. If NULL, it uses the default which is Invert.
pa	Optional; The previous action to be attached to this annotation. This can be NULL.

Returns

ILinkAnnotationPtr A smart pointer to the new link annotation

getActionOrDest()

```
virtual IPDFObjectPtr JawsMako::ILinkAnnotation::getActionOrDest ( ) const  [pure virtual]
```

Get the link annotation's Action or Dest object. An Action will be an IPDFDictionary object while a Dest can be IPDFArray, IPDFName, or IPDFString.

Returns

IPDFObjectPtr The link annotation's Action or Dest object

getHighlightMode()

```
virtual IPDFNamePtr JawsMako::ILinkAnnotation::getHighlightMode ( ) const  [pure virtual]
```

Get the link annotation's Highlight mode object.

Returns

IPDFNamePtr The link annotation's highlight mode

getPA()

```
virtual IPDFDictionaryPtr JawsMako::ILinkAnnotation::getPA ( ) const  [pure virtual]
```

Get the link annotation's previous Action object.

Returns

IPDFDictionaryPtr The link annotation's previous action dictionary object

getQuadPoints()

```
virtual CQuadPointVect JawsMako::ILinkAnnotation::getQuadPoints ( ) const  [pure virtual]
```

Get the link annotation's quad points if present. The points are relative to the annotation rect.

Returns

CQuadPointVect The link annotation's quad points

setActionOrDest()

```
virtual void JawsMako::ILinkAnnotation::setActionOrDest (
            const IPDFObjectPtr & actionOrDest )  [pure virtual]
```

Set the Action dictionary or Dest object for this annotation.

Parameters

actionOrDest	The action or dest object.

setHighlightMode()

```
virtual void JawsMako::ILinkAnnotation::setHighlightMode (
            const IPDFNamePtr & highlightMode )  [pure virtual]
```

Set the Highlight mode for this annotation.

Parameters

highlightMode	The highlight mode name object. This can be N, I, O, or P.

setPA()

```
virtual void JawsMako::ILinkAnnotation::setPA (
            const IPDFDictionaryPtr & pa )  [pure virtual]
```

Set the previous Action object for this annotation.

Parameters

pa	The previous action dictionary object.

setQuadPoints()

```
virtual void JawsMako::ILinkAnnotation::setQuadPoints (
            const CQuadPointVect & quadPoints )  [pure virtual]
```

Set the link annotation's quad points.

Parameters

quadPoints	The link annotation's quad points

The documentation for this class was generated from the following file:

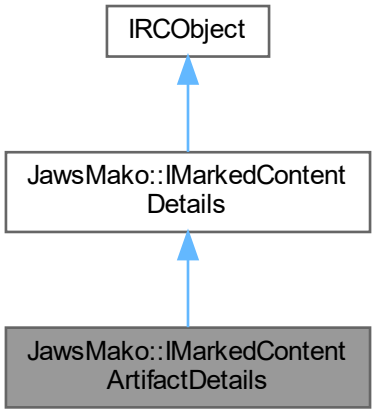
- interactive.h

7.335 JawsMako::IMarkedContentArtifactDetails Class Reference

A subclass of IMarkedContentDetails that is created when the content is a logical structure Artifact.

`#include <structure.h>`

Inheritance diagram for JawsMako::IMarkedContentArtifactDetails:



Public Member Functions

- virtual U8String **getType** () const =0

  Get the type of the artifact, if present. Returns an empty string if no Type information is provided.
- virtual U8String **getSubtype** () const =0

  Get the subtype of the artifact, if present. Returns an empty string if no Type information is provided.

Public Member Functions inherited from JawsMako::IMarkedContentDetails

- virtual U8String **getTag** () const =0

  Obtain the marked content's tag.
- virtual IPDFObjectPtr **getProperties** () const =0

  Obtain the properties, if present, as a PDF Object. Do not edit this object.
- virtual bool **getIsPoint** () const =0

  Do the marked content details represent a single point?

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IMarkedContentArtifactDetailsPtr **create** (const IJawsMakoPtr &jawsMako, const U8String &tag, const IRCObjectPtr &properties, bool isPoint)

  Basic marked content artifact details creation.
- static JAWSMAKO_API IMarkedContentArtifactDetailsPtr **create** (const IJawsMakoPtr &jawsMako, const U8String &type, const U8String &subType)

  Create marked content details for a logical structure Artifact In this context, an artifact is any object that are not relevant for the understanding of the content.

Static Public Member Functions inherited from JawsMako::IMarkedContentDetails

- static JAWSMAKO_API IMarkedContentDetailsPtr **create** (const IJawsMakoPtr &jawsMako, const U8String &tag, const IRCObjectPtr &properties, bool isPoint)

  Create general-purpose marked content details.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.335.1 Detailed Description

A subclass of IMarkedContentDetails that is created when the content is a logical structure Artifact.

The documentation for this class was generated from the following file:

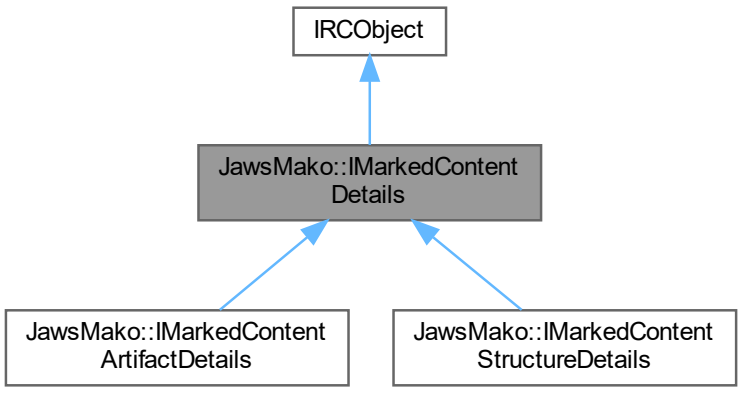
- structure.h

7.336 JawsMako::IMarkedContentDetails Class Reference

Details of Marked Content applied to an IDOMGroup.

`#include <structure.h>`

Inheritance diagram for JawsMako::IMarkedContentDetails:



Public Member Functions

- virtual U8String **getTag** () const =0

  Obtain the marked content's tag.
- virtual IPDFObjectPtr **getProperties** () const =0

  Obtain the properties, if present, as a PDF Object. Do not edit this object.
- virtual bool **getIsPoint** () const =0

  Do the marked content details represent a single point?

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IMarkedContentDetailsPtr **create** (const IJawsMakoPtr &jawsMako, const U8String &tag, const IRCObjectPtr &properties, bool isPoint)

  Create general-purpose marked content details.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.336.1 Detailed Description

Details of Marked Content applied to an IDOMGroup.

The documentation for this class was generated from the following file:

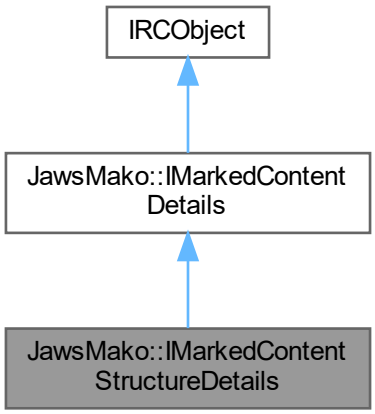
- structure.h

7.337 JawsMako::IMarkedContentStructureDetails Class Reference

A subclass of IMarkedContentDetails that is created when the marked content is associated with the document's structure.

`#include <structure.h>`

Inheritance diagram for JawsMako::IMarkedContentStructureDetails:



Public Member Functions

- virtual IStructureElementReferencePtr **getStructureElementReference** () const =0

  Obtain a reference for the structure element that this content refers to. Will never return NULL.

Public Member Functions inherited from JawsMako::IMarkedContentDetails

- virtual U8String **getTag** () const =0

  Obtain the marked content's tag.
- virtual IPDFObjectPtr **getProperties** () const =0

  Obtain the properties, if present, as a PDF Object. Do not edit this object.
- virtual bool **getIsPoint** () const =0

  Do the marked content details represent a single point?

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IMarkedContentStructureDetailsPtr **create** (const IJawsMakoPtr &jawsMako, const U8String &tag, const IRCObjectPtr &properties, bool isPoint, const IStructureElementReference↩ Ptr &elementReference)

  Create structure marked content details.

Static Public Member Functions inherited from JawsMako::IMarkedContentDetails

- static JAWSMAKO_API IMarkedContentDetailsPtr **create** (const IJawsMakoPtr &jawsMako, const U8String &tag, const IRCObjectPtr &properties, bool isPoint)

  Create general-purpose marked content details.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.337.1 Detailed Description

A subclass of IMarkedContentDetails that is created when the marked content is associated with the document's structure.

The documentation for this class was generated from the following file:

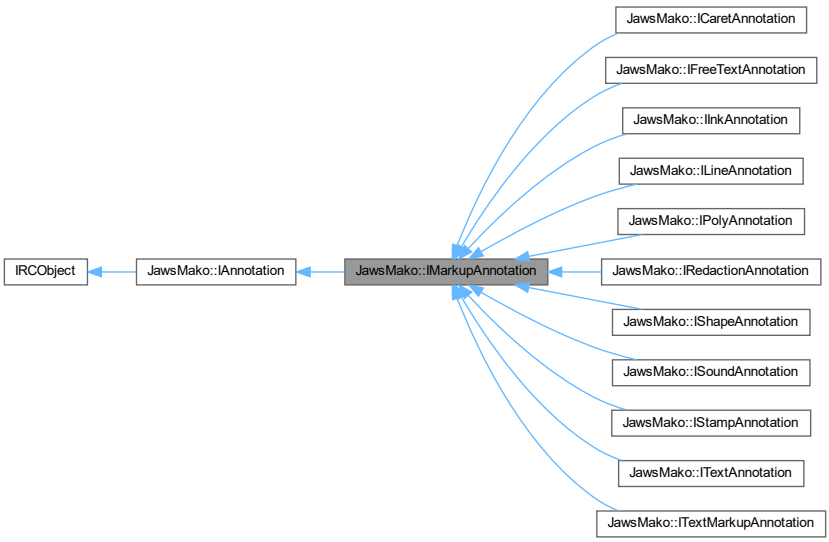
- structure.h

7.338 JawsMako::IMarkupAnnotation Class Reference

An interface class for markup annotations. It is intended that future releases of JawsMako will extend this interface.

`#include <interactive.h>`

Inheritance diagram for JawsMako::IMarkupAnnotation:



Public Member Functions

- virtual U8String getAuthor () const =0

  Get the Author of the markup annotation. This is the "T" entry in the annotation, which by convention is the author.
- virtual void setAuthor (const U8String &author)=0

  Set the Author of the markup annotation. This is the "T" entry in the annotation, which by convention is the author.
- virtual IEDLTimePtr getCreationTime () const =0

  Get the creation date and time of the annotation, if present.
- virtual void setCreationTime (const IEDLTimePtr &creationTime)=0

  Set the creation date and time of the annotation.
- virtual float getOpacity () const =0

  Get the opacity of the markup annotation.
- virtual void setOpacity (float opacity)=0

  Set the opacity of the markup annotation.
- virtual IAnnotationReferencePtr getPopupReference () const =0

  Get a reference to the popup, if present.
- virtual void setPopup (const IAnnotationReferencePtr &popupReference)=0

  Set or clear the popup, by reference.
- virtual void setPopup (const IPopupAnnotationPtr &popup)=0

  Set a reference to a popup, if present.
- virtual IAnnotationAppearancePtr createNormalAppearance () const =0

  Create a basic appearance given the current annotation's parameters. This can then be installed by using IAnnotation::addAppearance().

Public Member Functions inherited from JawsMako::IAnnotation

- virtual eAnnotationType getType () const =0

  Get the type of the annotation.
- virtual const FRect & getRect () const =0

  Get the rect in which the appearances should be displayed.
- virtual void setRect (const FRect &rect)=0

  Set the rect in which the appearances should be displayed.
- virtual U8String getContents () const =0

  Get the Contents entry in UTF-8.
- virtual void setContents (const U8String &contents)=0

  Set the Contents entry in UTF-8.
- virtual IDOMColorPtr getColor () const =0

  Get the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.
- virtual void setColor (const IDOMColorPtr &color)=0

  Set the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.
- virtual IEDLTimePtr getModificationTime () const =0

  Get the Modification date and time of the annotation, if present.
- virtual void setModificationTime (const IEDLTimePtr &modificationTime)=0

  Set the Modification date and time of the annotation.
- virtual CAnnotationBorder getBorder () const =0

  Get the annotation's border. See CAnnotationBorder for details.
- virtual void setBorder (const CAnnotationBorder &border)=0

  Set the annotation's border.
- virtual uint32 getFlags () const =0

  Get the annotation flags. To interpret these flags please see section 8.4.2 "Annotation Flags" in the PDF 1.7 specification.

- virtual void setFlags (uint32 flags)=0

  Set the annotation flags. Please see section 8.4.2 "Annotation Flags" for details about these flags.
- virtual void rotate (uint16 rotate, double pageWidth, double pageHeight)=0

  Rotate the annotation clockwise as if the page was rotated by the same amount.
- virtual CAnnotationAppearanceVect getAppearances () const =0

  Return all the annotation appearances in a vector.
- virtual void **removeAppearances** ()=0

  Remove all annotation appearances.
- virtual U8String getState () const =0

  Get the current annotation state.
- virtual void setState (const U8String &state)=0

  Set the current annotation state.
- virtual IAnnotationAppearancePtr getAppearance (eAppearanceUsage usage, const U8String &state=U8String()) const =0

  Fetch the annotation appearance that should be used for the given annotation usage and state according to the following rules:
- virtual void addAppearance (const IAnnotationAppearancePtr &appearance)=0

  Add or replace an appearance.
- virtual bool hasNormalAppearance () const =0

  Does the annotation have a normal appearance? Convenience utility function.
- virtual IAnnotationPtr clone () const =0

  Clone the annotation. This is a deep clone. The annotation reference will remain the same.
- virtual bool matchesReference (const IAnnotationReferencePtr &reference) const =0

  Does this annotation match the given IAnnotationReference?
- virtual IAnnotationReferencePtr getReference () const =0

  Get a reference that can be used to refer to this annotation.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Public Types inherited from JawsMako::IAnnotation

- enum eAnnotationType {
  eAT3D , eATCaret , eATCircle , eATFileAttachment ,
  eATFreeText , eATHighlight , eATInk , eATLine ,
  eATLink , eATMovie , eATPolygon , eATPolyLine ,
  eATPopup , eATPrinterMark , eATProjection , eATRedact ,
  eATRichMedia , eATScreen , eATSound , eATSquare ,
  eATSquiggly , eATStamp , eATStrikeOut , eATText ,
  eATTrapNet , eATUnderline , eATWatermark , eATWidget ,
  eATOther }

  Types of annotations, listed with the earliest version of PDF that supported them.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()
  Virtual destructor.

7.338.1 Detailed Description

An interface class for markup annotations. It is intended that future releases of JawsMako will extend this interface.

7.338.2 Member Function Documentation

createNormalAppearance()

```
virtual IAnnotationAppearancePtr JawsMako::IMarkupAnnotation::createNormalAppearance ( ) const
[pure virtual]
```

Create a basic appearance given the current annotation's parameters. This can then be installed by using IAnnotation::addAppearance().

Replaces any already present. Not yet supported for all markup annotation types. Supported for:

- Shape annotations
- Text annotations
- Text markup annotations

Inset, Beveled, Underline and "Cloud" borders will currently result in plain borders.

Additional annotation types may be supported in future releases.

An IError with error code JM_ERR_UNSUPPORTED will be thrown for such appearances.

Returns

IAnnotationAppearancePtr A smart pointer to the resulting appearance

getAuthor()

```
virtual U8String JawsMako::IMarkupAnnotation::getAuthor ( ) const  [pure virtual]
```

Get the Author of the markup annotation. This is the "T" entry in the annotation, which by convention is the author.

Returns

U8String The author. An empty string is returned if the entry is missing.

getCreationTime()

```
virtual IEDLTimePtr JawsMako::IMarkupAnnotation::getCreationTime ( ) const  [pure virtual]
```

Get the creation date and time of the annotation, if present.

Returns

IEDLTimePtr The annotation's creation date and time.

If the date is not of the expected format, or is missing, a NULL object will be returned.

getOpacity()

```
virtual float JawsMako::IMarkupAnnotation::getOpacity ( ) const  [pure virtual]
```

Get the opacity of the markup annotation.

Returns

float The opacity value

getPopupReference()

```
virtual IAnnotationReferencePtr JawsMako::IMarkupAnnotation::getPopupReference ( ) const  [pure
virtual]
```

Get a reference to the popup, if present.

Returns

IAnnotationReferencePtr The annotation reference to the popup. If there is no popup, a NULL reference will be returned.

setAuthor()

```
virtual void JawsMako::IMarkupAnnotation::setAuthor (
            const U8String & author )  [pure virtual]
```

Set the Author of the markup annotation. This is the "T" entry in the annotation, which by convention is the author.

Parameters

author	The author to set. An empty string will cause the author to be removed entirely.

setCreationTime()

```
virtual void JawsMako::IMarkupAnnotation::setCreationTime (
            const IEDLTimePtr & creationTime )  [pure virtual]
```

Set the creation date and time of the annotation.

Parameters

creationTime	The creation date and time. Must not be NULL.

setOpacity()

```
virtual void JawsMako::IMarkupAnnotation::setOpacity (
            float opacity )  [pure virtual]
```

Set the opacity of the markup annotation.

Parameters

opacity	A value between 0.0 (fully transparent) and 1.0 (fully opaque)

setPopup() [1/2]

```
virtual void JawsMako::IMarkupAnnotation::setPopup (
            const IAnnotationReferencePtr & popupReference )  [pure virtual]
```

Set or clear the popup, by reference.

Parameters

popupReference	A reference to the popup. Set to NULL to remove the current popup.

For full functionality, the popup must be added to the current page's annotations.

setPopup() [2/2]

```
virtual void JawsMako::IMarkupAnnotation::setPopup (
            const IPopupAnnotationPtr & popup )  [pure virtual]
```

Set a reference to a popup, if present.

Parameters

popup	The popup. Set to NULL to remove the current popup.

For full functionality, the popup must be added to the current page's annotations.

The documentation for this class was generated from the following file:

- interactive.h

7.339 JawsMako::IMediaHandler Class Reference

Interface allowing users of supported input types to monitor/handle unsatisfied media requests. For input interfaces that support this feature, use setMediaHandler() to install subclasses of this type.

```
#include <jawsmako.h>
```

Public Member Functions

- virtual void sizeRequest (const PValue &size, double &pageWidth, double &pageHeight)=0

  Invoked by the input when the media size is unknown.

7.339.1 Detailed Description

Interface allowing users of supported input types to monitor/handle unsatisfied media requests. For input interfaces that support this feature, use setMediaHandler() to install subclasses of this type.

7.339.2 Member Function Documentation

sizeRequest()

```
virtual void JawsMako::IMediaHandler::sizeRequest (
            const PValue & size,
            double & pageWidth,
            double & pageHeight )  [pure virtual]
```

Invoked by the input when the media size is unknown.

Parameters

size	The requested media size, which may be an integer or a string.
pageWidth	The default width that will be set for the requested media size. Users may set this value to override the default page width.
pageHeight	The default height that will be set for the requested media size. Users may set this value to override the default page height.

The documentation for this class was generated from the following file:

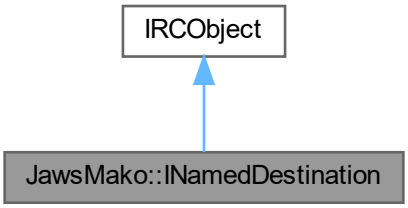
- jawsmako.h

7.340 JawsMako::INamedDestination Class Reference

A named destination in a PDF Document.

`#include <interactive.h>`

Inheritance diagram for JawsMako::INamedDestination:



Public Member Functions

- virtual INamedDestinationPtr clone () const =0

  Clone the named destination.
- virtual const U8String & getName () const =0

  Get the name of this named destination.
- virtual IDOMPageRectTargetPtr getTarget () const =0

  Get the destination.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API INamedDestinationPtr create (const IJawsMakoPtr &jawsMako, const U8String &name, const IDOMPageRectTargetPtr &target)

  Create a named destination with the given name to the given target.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.340.1 Detailed Description

A named destination in a PDF Document.

7.340.2 Member Function Documentation

clone()

```
virtual INamedDestinationPtr JawsMako::INamedDestination::clone ( ) const  [pure virtual]
```

Clone the named destination.

Returns

INamedDestinationPtr A smart pointer to the cloned named destination

create()

```
static JAWSMAKO_API INamedDestinationPtr JawsMako::INamedDestination::create (
            const IJawsMakoPtr & jawsMako,
            const U8String & name,
            const IDOMPageRectTargetPtr & target )  [static]
```

Create a named destination with the given name to the given target.

Parameters

jawsMako	The JawsMako instance
name	The name of the destination
target	The target of the destination

Returns

INamedDestinationPtr A smart pointer to the new named destination

getName()

```
virtual const U8String & JawsMako::INamedDestination::getName ( ) const  [pure virtual]
```

Get the name of this named destination.

Returns

U8String The name of this named destination

getTarget()

```
virtual IDOMPageRectTargetPtr JawsMako::INamedDestination::getTarget ( ) const  [pure virtual]
```

Get the destination.

Returns

IDOMPageRectTargetPtr A smart pointer to a page rect target that represents the destination of this named destination, or NULL if the destination is not understood

Do not edit this destination

The documentation for this class was generated from the following file:

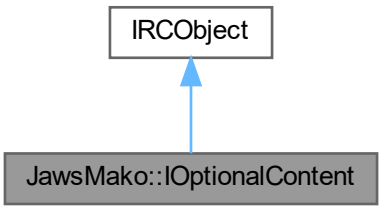
- interactive.h

7.341 JawsMako::IOptionalContent Class Reference

Root level optional content information for an entire document.

```
#include <optionalcontent.h>
```

Inheritance diagram for JawsMako::IOptionalContent:



Public Member Functions

- virtual IOptionalContentPtr **clone** ()=0

  Clone the optional content. This is a deep clone.
- virtual IOptionalContentGroupPtr **makeNewGroup** (const U8String &name, bool visible)=0

  Convenience to create a new optional content group with the given visibility status, and add. The group will also be added to the ui by adding to the default configuration's order.
- virtual IDOMNodePtr makeNodeOptional (const IDOMNodePtr &node, const IOptionalContentGroupPtr &group)=0

  Make the given DOM node optional as part of an existing optional content group.
- IDOMNodePtr **makeNodeOptionalWithNewGroup** (const IDOMNodePtr &node, const U8String &new↩ GroupName, bool visible)

Make the given DOM node optional with a new optional content group.
Convenience.

- virtual bool groupIsVisible (const IOptionalContentGroupPtr &group, eOptionalContentEvent event=eOCEView, IOptionalContentConfigurationPtr configuration=IOptionalContentConfigurationPtr())=0

  Determine if the given optional content group is visible in the given circumstances.
- virtual bool groupIsVisible (const IOptionalContentGroupReferencePtr &groupRef, eOptionalContentEvent event=eOCEView, IOptionalContentConfigurationPtr configuration=IOptionalContentConfigurationPtr())=0

  Determine if the given optional content group is visible in the given circumstances, using the group's reference.
- virtual bool groupShouldBeIgnored (const IOptionalContentGroupPtr &group, eOptionalContentEvent event=eOCEView, IOptionalContentConfigurationPtr configuration=IOptionalContentConfigurationPtr())=0

  Determine if the given optional content group should be ignored for visibility purposes in the given circumstances. Groups that have an intent that does not match the configuration's intent are not to be used to determine visibility, necessitating this distinction.
- virtual bool groupShouldBeIgnored (const IOptionalContentGroupReferencePtr &groupRef, eOptionalContentEvent event=eOCEView, IOptionalContentConfigurationPtr configuration=IOptionalContentConfigurationPtr())=0

  Determine if the given optional content group (by reference) should be ignored for visibility purposes in the given circumstances, using the group's reference. Groups that have an intent that does not match the configuration's intent are not to be used to determine visibility, necessitating this distinction.
- virtual bool **groupIsPresent** (const IOptionalContentGroupReferencePtr &groupRef) const =0

  Is a group with the given reference present?
- virtual IOptionalContentGroupPtr **getGroup** (const IOptionalContentGroupReferencePtr &groupRef) const =0

  Get the optional content group with the given reference. An exception will be thrown if the group is not found.
- virtual void forceGroupState (const IOptionalContentGroupPtr &group, bool visible)=0

  Force the given group to be always on or off in the default configuration. Convenience method.
- virtual COptionalContentGroupVect **getGroups** () const =0

  Get all the document's optional content groups in a vector.
- virtual void **addGroup** (const IOptionalContentGroupPtr &group)=0

  Add an optional content group. An error will be thrown if a group with the same reference is already present.
- virtual void **addGroup** (const IOptionalContentGroupPtr &group, const IDocumentPtr &sourceDocument)=0

  Add an optional content group from another document.
  This will attempt to maintain the context of the optional content group from the source document, adding to the default configuration in order to maintain its user-visible behavior as closely as possible.
- virtual void **removeGroup** (const IOptionalContentGroupPtr &group)=0

  Remove an optional content group. This will also purge the group from any mention in the default or alternate configurations.
- virtual void setDefaultConfiguration (const IOptionalContentConfigurationPtr &configuration)=0

  Set the default optional content configuration.
- virtual IOptionalContentConfigurationPtr **getDefaultConfiguration** () const =0

  Get the default optional content configuration.
- virtual void **addConfiguration** (const IOptionalContentConfigurationPtr &configuration)=0

  Add to the list of configurations for this optional content. If the configuration is already present, no action will be taken.
- virtual void **removeConfiguration** (const IOptionalContentConfigurationPtr &configuration)=0

  Remove the given configuration. This configuration must not be the current default or an exception will be thrown.
- virtual COptionalContentConfigurationVect **getConfigurations** () const =0

  Get all the configurations.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IOptionalContentPtr **create** (const IJawsMakoPtr &jawsMako, const IOptional↩ ContentConfigurationPtr &defaultConfiguration=IOptionalContentConfigurationPtr())

  Create a new optional content object. If no default configuration is provided, one will be created.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.341.1 Detailed Description

Root level optional content information for an entire document.

7.341.2 Member Function Documentation

forceGroupState()

```
virtual void JawsMako::IOptionalContent::forceGroupState (
            const IOptionalContentGroupPtr & group,
            bool visible )  [pure virtual]
```

Force the given group to be always on or off in the default configuration. Convenience method.

This will:

- ensure the group is in (or removed from) the default visible and invisible list
- ensure the group has the same intent as the configuration
- remove any usage information from the group

groupIsVisible() [1/2]

```
virtual bool JawsMako::IOptionalContent::groupIsVisible (
            const IOptionalContentGroupPtr & group,
            eOptionalContentEvent event = eOCEView,
            IOptionalContentConfigurationPtr configuration = IOptionalContentConfigurationPtr()
)  [pure virtual]
```

Determine if the given optional content group is visible in the given circumstances.

Parameters

group	The group to check.
event	The event representing the intended use. Pass eOCEUnknown if the automatic state IOptionalContentGroupUsageApplication should not be checked.
configuration	The configuration to use. Pass NULL to use the default configuration.

groupIsVisible() [2/2]

```
virtual bool JawsMako::IOptionalContent::groupIsVisible (
            const IOptionalContentGroupReferencePtr & groupRef,
            eOptionalContentEvent event = eOCEView,
            IOptionalContentConfigurationPtr configuration = IOptionalContentConfigurationPtr()
)  [pure virtual]
```

Determine if the given optional content group is visible in the given circumstances, using the group's reference.

Parameters

groupRef	Reference to the group to check.
event	The event representing the intended use. Pass eOCEUnknown if the automatic state IOptionalContentGroupUsageApplication should not be checked.
configuration	The configuration to use. Pass NULL to use the default configuration.

groupShouldBeIgnored() [1/2]

```
virtual bool JawsMako::IOptionalContent::groupShouldBeIgnored (
            const IOptionalContentGroupPtr & group,
            eOptionalContentEvent event = eOCEView,
            IOptionalContentConfigurationPtr configuration = IOptionalContentConfigurationPtr()
)  [pure virtual]
```

Determine if the given optional content group should be ignored for visibility purposes in the given circumstances. Groups that have an intent that does not match the configuration's intent are not to be used to determine visibility, necessitating this distinction.

Parameters

group	The group to check.
event	The event representing the intended use. Pass eOCEUnknown if the automatic state IOptionalContentGroupUsageApplication should not be checked.
configuration	The configuration to use. Pass NULL to use the default configuration.

groupShouldBeIgnored() [2/2]

```
virtual bool JawsMako::IOptionalContent::groupShouldBeIgnored (
            const IOptionalContentGroupReferencePtr & groupRef,
            eOptionalContentEvent event = eOCEView,
            IOptionalContentConfigurationPtr configuration = IOptionalContentConfigurationPtr()
)  [pure virtual]
```

Determine if the given optional content group (by reference) should be ignored for visibility purposes in the given circumstances, using the group's reference. Groups that have an intent that does not match the configuration's intent are not to be used to determine visibility, necessitating this distinction.

Parameters

groupRef	Reference to the group to check.

Parameters

event	The event representing the intended use. Pass eOCEUnknown if the automatic state IOptionalContentGroupUsageApplication should not be checked.
configuration	The configuration to use. Pass NULL to use the default configuration.

makeNodeOptional()

```
virtual IDOMNodePtr JawsMako::IOptionalContent::makeNodeOptional (
            const IDOMNodePtr & node,
            const IOptionalContentGroupPtr & group )  [pure virtual]
```

Make the given DOM node optional as part of an existing optional content group.

If the node is an IDOMGroup, the group will have it's optional content information set accordingly and the returned result will be that node. However, if the node is some other graphical node, it will be moved inside an IDOMGroup with the optional content information set. If the node is in a tree, the node will be replaced with the group. The group will be returned.

Convenience.

setDefaultConfiguration()

```
virtual void JawsMako::IOptionalContent::setDefaultConfiguration (
            const IOptionalContentConfigurationPtr & configuration )  [pure virtual]
```

Set the default optional content configuration.

Note

There are restrictions on the intent attribute of an optional content configuration for it to be the default, which must be "View". Also the base state must be visible. This routine will modify the configuration to suit.

If the optional content configuration is not present in the configuration list, it will be added.

The documentation for this class was generated from the following file:

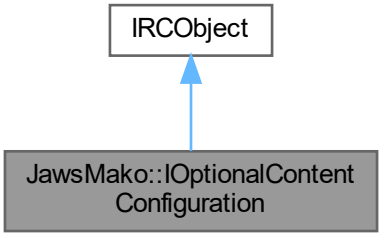
- optionalcontent.h

7.342 JawsMako::IOptionalContentConfiguration Class Reference

A configuration for optional content.

`#include <optionalcontent.h>`

Inheritance diagram for JawsMako::IOptionalContentConfiguration:



Classes

- class COrderEntry

  Class for presenting the order that groups should be displayed in a user interface. May be arranged in a tree.

Public Member Functions

- virtual IOptionalContentConfigurationPtr **clone** () const =0

  Clone the configuration.
- virtual void **setName** (const U8String &name)=0

  Set the name of the configuration. May be an empty string.
- virtual U8String **getName** () const =0

  Get the name of the configuration. An empty string is returned if no name is present.
- virtual void **setCreator** (const U8String &creator)=0

  Set the "creator" of the configuration. May be an empty string.
- virtual U8String **getCreator** () const =0

  Get the "creator" of the configuration. An empty string is returned if no name is present.
- virtual void **setBaseState** (eOptionalContentVisibility visibility)=0

  Get the base visibility state of the optional content groups in the document.
- virtual eOptionalContentVisibility **getBaseState** () const =0

  Get the base visibility state of the optional content groups in the document.
- virtual void **setDefaultVisibleGroups** (const COptionalContentGroupReferenceVect &groups)=0

  Set the vector of optional content group references that should be visible by default.
- virtual COptionalContentGroupReferenceVect **getDefaultVisibleGroups** () const =0

  Get the vector of optional content groups references that should be visible by default.
- virtual bool **groupInVisibleGroups** (const IOptionalContentGroupReferencePtr &groupRef) const =0

  Is the given group present in the visible groups?
- virtual void **setDefaultInvisibleGroups** (const COptionalContentGroupReferenceVect &groups)=0

  Set the vector of optional content group references that should be invisible by default.
- virtual COptionalContentGroupReferenceVect **getDefaultInvisibleGroups** () const =0

  Get the vector of optional content groups references that should be invisible by default.
- virtual bool **groupInInvisibleGroups** (const IOptionalContentGroupReferencePtr &groupRef) const =0

  Is the given group present in the invisible groups?
- virtual void **setIntent** (const U8String &intent)=0

  Set the intent for this configuration Pass an empty string to remove the intent.
- virtual void **setIntents** (const CU8StringVect &intents)=0

  Set an array of intents for this configuration. If set to an empty array, all groups will be considered visible, regardless of any other status information. To remove the intents entry, use setIntent above with an empty string.
- virtual CU8StringVect **getIntents** () const =0

  Get the intents for this configuration. Usually this will be a single entry for most real world cases. A default of "View" will be returned if the entry is not present.
- virtual bool **intentInIntents** (const U8String &intent) const =0

  Determine if the given intent is present in the list of intents. If the intents specify All then all intents will match. intent must not be an empty string.
- virtual void **setAutoStates** (const COptionalContentGroupUsageApplicationVect &applications)=0

  Set the usage applications that apply to this configuration.
- virtual COptionalContentGroupUsageApplicationVect **getAutoStates** () const =0

  Get the usage applications that apply to this configuraiton.
- virtual void **setOrder** (const COrderEntryVect &order)=0

  Set the order that groups should be shown in a user interface.
- virtual COrderEntryVect **getOrder** () const =0

Get the order that groups should be shown in a user interface. An independent copy will be returned.

- virtual void **setListMode** (eListMode listMode)=0

  Set which optional content groups should be shown in the consuming application's user interface.

- virtual eListMode **getListMode** () const =0

  Get which optional content groups should be shown in the consuming application's user interface.

- virtual void **setRadioButtonGroups** (const CEDLVector< COptionalContentGroupReferenceVect > &rb↩ Groups)=0

  Set the "Radio Button" optional content groups for this configuration. This is a vector of vectors of group references. For each of the vectors of references, setting a group to Visible should result in all of the others to become invisible. Pass an empty vector to remove this feature.

- virtual CEDLVector< COptionalContentGroupReferenceVect > **getRadioButtonGroups** () const =0

  Get the "Radio Button" optional content groups for this configuration. This is a vector of vectors of group references. For each of the vectors of references, setting a group to Visible should result in all of the others to become invisible.

- virtual void **setLockedGroups** (const COptionalContentGroupReferenceVect &lockedGroups)=0

  Set the optional content groups that should be locked in the user interface.

- virtual COptionalContentGroupReferenceVect **getLockedGroups** () const =0

  Get the optional content groups that should be locked in the user interface.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.

- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.

- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IOptionalContentConfigurationPtr **create** (const IJawsMakoPtr &jawsMako)

  Create a new configuration.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.342.1 Detailed Description

A configuration for optional content.

The documentation for this class was generated from the following file:

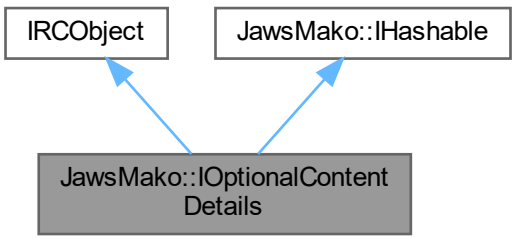
- optionalcontent.h

7.343 JawsMako::IOptionalContentDetails Class Reference

Interface for objects used to tag content as optional. Instances of this class are set in IDOMGroup instances to make those objects optional, linking them to one or more optional content groups.

`#include <optionalcontent.h>`

Inheritance diagram for JawsMako::IOptionalContentDetails:



Public Types

- enum eVisibilityPolicy { eVPAllOn , eVPAnyOn , eVPAnyOff , eVPAllOff }

  The visibility policy controlling how the dependent optional content groups affect the visibility of this content.

Public Member Functions

- virtual IOptionalContentDetailsPtr **clone** () const =0

  Clone.
- virtual bool getIsVisible (eOptionalContentEvent event=eOCEView, const IOptionalContentPtr &optional↩ Content=IOptionalContentPtr()) const =0

  Determine the visibility of this content. Pass in the Optional Content for this document to obtain current results. Otherwise, the visibility returned will be either:
- virtual void **addGroup** (const IOptionalContentGroupReferencePtr &groupRef)=0

  Make the optional content dependent on the given group (if it is not already so). Optional content can be a member of multiple groups. The visibility policy or visibility expression (see below) controls how the visibility of this piece of optional content is calculated based on the groups upon which it depends.
- virtual void **removeGroup** (const IOptionalContentGroupReferencePtr &groupRef)=0

  Remove the dependence of the content on the given group.
- virtual COptionalContentGroupReferenceVect **getGroupReferences** () const =0

  Get all the groups that this content depends on.
- virtual void **setVisibilityPolicy** (eVisibilityPolicy policy)=0

  Set the visibility policy. If present, the visibility expression will take precedence. The default is eAnyOn.
- virtual eVisibilityPolicy **getVisibilityPolicy** () const =0

  Get the visibility policy.
- virtual void **setVisibilityExpression** (const IOptionalContentVisibilityExpressionPtr &expression)=0

  Set the visibility expression, which affords more powerful control of the visibilty of this content via a logical expression. This takes precedence over the visibility policy for consumers that support visibility expressions. Pass NULL to remove any existing expression.
- virtual IOptionalContentVisibilityExpressionPtr **getVisibilityExpression** ()=0

  Get the visibility expression.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from JawsMako::IHashable

- virtual uint64 hash () const

  Obtain a 64-bit hash of the receiving object.
- virtual void updateHash (uint64 &hash) const =0

  Update the given hash to include the receiver.

Static Public Member Functions

- static JAWSMAKO_API IOptionalContentDetailsPtr **create** (const IJawsMakoPtr jawsMako, const IOptional↩ ContentGroupReferencePtr &groupRef=IOptionalContentGroupReferencePtr(), const IOptionalContentPtr &optionalContent=IOptionalContentPtr())

  Create using the given (optional) group.
- static JAWSMAKO_API IOptionalContentDetailsPtr **create** (const IJawsMakoPtr jawsMako, const COptionalContentGroupReferenceVect &groups, eVisibilityPolicy policy, const IOptionalContentVisibility↩ ExpressionPtr &visibilityExpression=IOptionalContentVisibilityExpressionPtr(), const IOptionalContentPtr &optionalContent=IOptionalContentPtr())

  Create using the given groups, policy, and (optional) visibility expression.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.343.1 Detailed Description

Interface for objects used to tag content as optional. Instances of this class are set in IDOMGroup instances to make those objects optional, linking them to one or more optional content groups.

7.343.2 Member Enumeration Documentation

eVisibilityPolicy

```
enum JawsMako::IOptionalContentDetails::eVisibilityPolicy
```

The visibility policy controlling how the dependent optional content groups affect the visibility of this content.

Enumerator

eVPAllOn	This content is visible if all the dependent groups are visible.
eVPAnyOn	This content is visible if any dependent group is visible.
eVPAnyOff	This content is visible if any dependent group is invisible.
eVPAllOff	This content is visible if all the dependent groups are invisible.

7.343.3 Member Function Documentation

getIsVisible()

```
virtual bool JawsMako::IOptionalContentDetails::getIsVisible (
            eOptionalContentEvent event = eOCEView,
            const IOptionalContentPtr & optionalContent = IOptionalContentPtr() ) const  [pure
virtual]
```

Determine the visibility of this content. Pass in the Optional Content for this document to obtain current results. Otherwise, the visibility returned will be either:

- The optional content state at the time this object was created, if the IOptionalContent was passed at the time of creation (always the case for objects created directly from reading from a PDF, or if it was created using an IOptionalContent API function)
- Otherwise, true is returned.

The documentation for this class was generated from the following file:

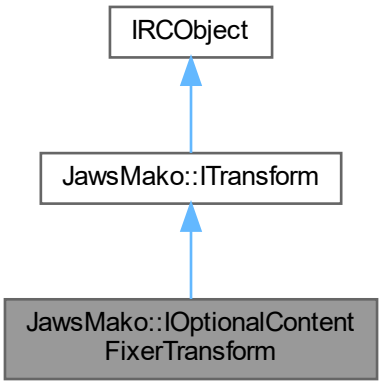
- optionalcontent.h

7.344 JawsMako::IOptionalContentFixerTransform Class Reference

A simple transform that strips the DOM of any PDF optional content that is not visible for the given document use. This transform also selects from any alternate images, if present.

```
#include <transforms.h>
```

Inheritance diagram for JawsMako::IOptionalContentFixerTransform:



Public Member Functions

- virtual void **setOptionalContent** (const IOptionalContentPtr &optionalContent)=0

  Sets the optional content data to use when making decisions. Without this, the status of the optional content at the time of creation will be used.
- virtual void **setOptionalContentUsage** (eOptionalContentUsage usage)=0

  Sets the usage of the optional content items that should be retained. The default is eOCUView.

Public Member Functions inherited from JawsMako::ITransform

- virtual IDOMBrushPtr transform (const IDOMBrushPtr &brush, eBrushUsage usage=eBUGeneral, const CTransformState &state=CTransformState())=0

  Apply the transform to the given brush, if applicable. These transforms are thread safe.
- virtual IDOMImagePtr transform (const IDOMImagePtr &image, const CTransformState &state=CTransformState())=0

  Apply the transform to the given image, if applicable. These transforms are thread safe.
- virtual IDOMColorPtr transform (const IDOMColorPtr &color, const CTransformState &state=CTransformState())=0

  Apply the transform to the given color, if applicable. These transforms are thread safe.
- virtual IDOMColorSpacePtr transform (const IDOMColorSpacePtr &colorSpace, const CTransformState &state=CTransformState())=0

  Apply the transform to the given colorspace, if applicable. These transforms are thread safe.
- virtual IDOMNodePtr transform (const IDOMNodePtr &node, bool &changed, bool transformChildren=true, const CTransformState &state=CTransformState())=0

  Apply the transform to the given node, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time.
- virtual void transformPage (const IPagePtr &page, bool transformContent=true, bool transform↩ Annotations=true)=0

  Apply the transform to the given page, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time. The transform will also apply to the annotations appearances.
- virtual void **flushCaches** ()=0

  Flush the caches used by the transform. Most transforms cache recently transformed results to improve the performance of repeated transformations of equivalent results. However, it is possible that some cached results may point to entities that no longer exist, such as content inside an XPS file that no longer exists. If you are deleting or replacing files where transforms have been used, it is advisable to invoke this routine to clear the caches.
- virtual void setProgressMonitor (const IProgressMonitorPtr &progressMonitor)=0

  Set the IProgressMonitor object for this transform to allow for monitoring the progress of the transform.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IOptionalContentFixerTransformPtr create (const IJawsMakoPtr &jawsMako, const IAbortPtr &abort=IAbortPtr())

  Create the transform.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.344.1 Detailed Description

A simple transform that strips the DOM of any PDF optional content that is not visible for the given document use. This transform also selects from any alternate images, if present.

7.344.2 Member Function Documentation

create()

```
static JAWSMAKO_API IOptionalContentFixerTransformPtr JawsMako::IOptionalContentFixerTransform↩
::create (
            const IJawsMakoPtr & jawsMako,
            const IAbortPtr & abort = IAbortPtr() )  [static]
```

Create the transform.

Parameters

jawsMako	The JawsMako instance.
abort	An abort callback handler.

Returns

The new instance.

The documentation for this class was generated from the following file:

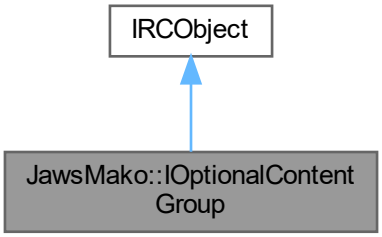
- transforms.h

7.345 JawsMako::IOptionalContentGroup Class Reference

Interface for an optional content group.

```
#include <optionalcontent.h>
```

Inheritance diagram for JawsMako::IOptionalContentGroup:



Public Member Functions

- virtual IOptionalContentGroupPtr **clone** ()=0

  Clone the group. Note that cloning the group will not result in a new reference being created.
- virtual IOptionalContentGroupReferencePtr **getReference** () const =0

  Get the reference to this group. This is used with several APIs to maintain a link to this group.
- virtual void **setName** (const U8String &name)=0

  Set the name of this group. Must not be an empty string.
- virtual U8String **getName** () const =0

  Get the name of this group.
- virtual void **setIntent** (const U8String &intent)=0

  Set the intent of the group, or an empty string to reset to the default.
- virtual void **setIntents** (const CU8StringVect &intents)=0

  Set an array of intents for this group.
- virtual CU8StringVect **getIntents** () const =0

  Get the intents for this group. Usually this will be a single entry for most real world cases. A default of "View" will be returned if the entry is not present.
- virtual void **setUsage** (const IOptionalContentGroupUsagePtr &usage)=0

  Set the Usage information for this group. Pass NULL to remove the usage information.
- virtual IOptionalContentGroupUsagePtr **getUsage** () const =0

  Get the Usage information for this group. May be NULL.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IOptionalContentGroupPtr **create** (const IJawsMakoPtr &jawsMako, const U8String &name)

  Create an optional content group. A non-zero length name must be provided.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.345.1 Detailed Description

Interface for an optional content group.

The documentation for this class was generated from the following file:

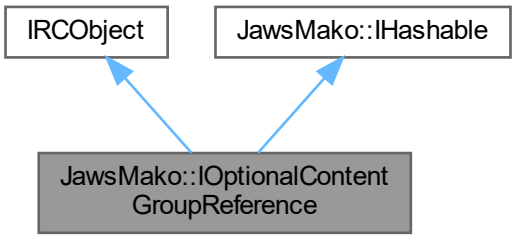
- optionalcontent.h

7.346 JawsMako::IOptionalContentGroupReference Class Reference

A reference to an optional content group.

`#include <optionalcontent.h>`

Inheritance diagram for JawsMako::IOptionalContentGroupReference:



Public Member Functions

- virtual bool **equals** (const IOptionalContentGroupReferencePtr &reference) const =0

  Does the given reference point to the same group as this reference?

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from JawsMako::IHashable

- virtual uint64 hash () const

  Obtain a 64-bit hash of the receiving object.
- virtual void updateHash (uint64 &hash) const =0

  Update the given hash to include the receiver.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.346.1 Detailed Description

A reference to an optional content group.

The documentation for this class was generated from the following file:

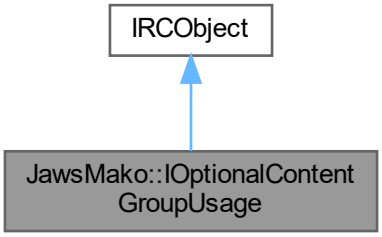
- optionalcontent.h

7.347 JawsMako::IOptionalContentGroupUsage Class Reference

Usage information for an optional content group, providing context that an application can use to automatically show or hide content in the optional content group. This is optional.

`#include <optionalcontent.h>`

Inheritance diagram for JawsMako::IOptionalContentGroupUsage:



Public Member Functions

- virtual IOptionalContentGroupUsagePtr **clone** () const =0

  Clone the usage information.
- virtual void setLanguage (const U8String &language, bool preferred=false)=0

  Set the language of the content controlled by this optional content group, and whether or not it should be considered "preferred" if there is no exact match for the language used by the viewer's.
- virtual U8String **getLanguage** () const =0

  Get the language for the group, which may be an empty string if no language has been set.
- virtual bool **getLanguageIsPreferred** () const =0

  Get whether or not the language of this optional content group should be considered "preferred".
- virtual void **setExportVisibility** (eOptionalContentVisibility visibility)=0

  Set if the content in the group should be present when the content is "exported" to a format that does not support optional content.
- virtual eOptionalContentVisibility **getExportVisibility** () const =0

  Get whether the content in the group should be present when the content is "exported" to a format that does not support optional content.
- virtual void **setZoomVisibility** (float minimumZoom, float maximumZoom)=0

  Set if the content should be visible between the given zoom levels. Setting maximumZoom to a negative value indicates infinity.
- virtual void **getZoomVisibility** (float &minimumZoom, float &maximumZoom) const =0

  Get the zoom levels between which this content is visible. A negative maximumZoom indicates infinity.
- virtual void **setPrintVisibility** (eOptionalContentVisibility visibility, const U8String &description=U8String())=0

  Set if the content in the group should be present when the content is printed, and optionally an informative string describing the kind of content.
- virtual eOptionalContentVisibility **getPrintVisibility** () const =0

  Get whether the content in the group should be present when the content is printed.
- virtual U8String **getPrintVisibilityDescription** () const =0

  Get the print visibility description, which may be an empty string.
- virtual void **setViewVisibility** (eOptionalContentVisibility visibility)=0

  Set if the content in the group should be visible when the content is viewed.
- virtual eOptionalContentVisibility **getViewVisibility** () const =0

  Get whether the content in the group should be visible when the content is viewed.
- virtual void setUsers (const U8String &type, const CU8StringVect &users)=0

  Get the type and users for which this content is primarily intended.
- virtual CU8StringVect getUsers (U8String &type) const =0

  Get the type and users for which this content is primarily intended.
- virtual void setPageElement (const U8String &type)=0

  Sets a description of the page element that is represented by the optional content group.
- virtual U8String **getPageElement** () const =0

  Gets a description of the page element that is represented by the optional content group. May be an empty string.
- virtual eOptionalContentVisibility recommendedVisibilityForCategories (const COCCategoryVect &categories) const =0

  Deteremine the recommended state from this usage information for the given usage categories. Uses the algorithm on page 383 of the PDF 1.7 specification, with the following modifications:

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IOptionalContentGroupUsagePtr **create** (const IJawsMakoPtr &jawsMako)

  Create usage information.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.347.1 Detailed Description

Usage information for an optional content group, providing context that an application can use to automatically show or hide content in the optional content group. This is optional.

These are only consulted when making visibility decisions if the document's IOptionalContent specifies that the usage for this group should be used, and even then only some attributes may be consulted depending on how the usages are configured. See #IOptionalContentGroupUsageApplication below.

7.347.2 Member Function Documentation

getUsers()

```
virtual CU8StringVect JawsMako::IOptionalContentGroupUsage::getUsers (
            U8String & type ) const  [pure virtual]
```

Get the type and users for which this content is primarily intended.

Parameters

type	Reference to receive the type of user.

Returns

The user names, or an empty array if none are present.

recommendedVisibilityForCategories()

```
virtual eOptionalContentVisibility JawsMako::IOptionalContentGroupUsage::recommendedVisibility↩
ForCategories (
            const COCCategoryVect & categories ) const  [pure virtual]
```

Deteremine the recommended state from this usage information for the given usage categories. Uses the algorithm on page 383 of the PDF 1.7 specification, with the following modifications:

- Zoom is not considered and always assumed ON
- User is not considered and always assumed ON
- Language is considered ON if the language is preferred. May return Unchanged.

setLanguage()

```
virtual void JawsMako::IOptionalContentGroupUsage::setLanguage (
            const U8String & language,
            bool preferred = false )  [pure virtual]
```

Set the language of the content controlled by this optional content group, and whether or not it should be considered "preferred" if there is no exact match for the language used by the viewer's.

Parameters

language	A "Natural Language Specification" string representing the language and possible a locale, such as "en-US". Pass an empty string to specify no particular language.
preferred	Whether this should be considered preferred if no exact match for the user's viewing language is found.

setPageElement()

```
virtual void JawsMako::IOptionalContentGroupUsage::setPageElement (
            const U8String & type )  [pure virtual]
```

Sets a description of the page element that is represented by the optional content group.

Parameters

type	Should be either "HF" (Header or footer), "FG" (foreground content), "BG" (Background content), "L" (Logo) or an empty string indicating no particular page element is represented.

setUsers()

```
virtual void JawsMako::IOptionalContentGroupUsage::setUsers (
            const U8String & type,
            const CU8StringVect & users )  [pure virtual]
```

Get the type and users for which this content is primarily intended.

Parameters

type	The type of user, which should be either "Ind" (Individual), "Ttl" (Title), or "Org" (Organisation). Must not be empty unless the users vector is also empty (which will cause the user information to be ignored for visibility determination)
users	The user names.

The documentation for this class was generated from the following file:

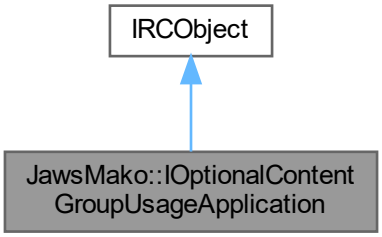
- optionalcontent.h

7.348 JawsMako::IOptionalContentGroupUsageApplication Class Reference

Interface for controlling how IOptionalContentGroupUsage is applied, and for what groups.

`#include <optionalcontent.h>`

Inheritance diagram for JawsMako::IOptionalContentGroupUsageApplication:



Public Member Functions

- virtual IOptionalContentGroupUsageApplicationPtr **clone** () const =0

  Clone the usage application information.
- virtual void **setEvent** (eOptionalContentEvent event)=0

  Set the event for which the the optional content usage should be applied. Must not be eOCEUnknown.
- virtual eOptionalContentEvent **getEvent** () const =0

  Get the event for which the optional content usage should be applied.
- virtual void **setAffectedOptionalContentGroups** (const COptionalContentGroupReferenceVect &groups)=0

  Set the optional content groups (by reference) for which the usage information should be applied.
- virtual COptionalContentGroupReferenceVect **getAffectedOptionalContentGroups** () const =0

  Get the optional content groups for which the usage information should be applied.
- virtual void **addGroup** (const IOptionalContentGroupReferencePtr &groupRef)=0

  Add the given group to the list of affected groups. It should not already be present.
- virtual bool **groupIsAffected** (const IOptionalContentGroupReferencePtr &groupRef) const =0

  Is the given group affected by this usage application?
- virtual void **setCategories** (const COCCategoryVect &categories)=0

  Set the categories from the optional content groups that should be consulted to determine visibility.
- virtual COCCategoryVect **getCategories** () const =0

  Get the categories from the optional content groups that should be consulted to determine visibility.
- virtual bool **categoryInCategories** (eOptionalContentCategory category) const =0

  Does the usage categories include the given category?
- virtual bool **categoriesInCategories** (const COCCategoryVect &categories) const =0

  Does the usage categories include all the given categories? Convenience.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IOptionalContentGroupUsageApplicationPtr **create** (const IJawsMakoPtr &jaws↩Mako, eOptionalContentEvent event=eOCEView)

  Create usage application information.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.348.1 Detailed Description

Interface for controlling how IOptionalContentGroupUsage is applied, and for what groups.

The documentation for this class was generated from the following file:

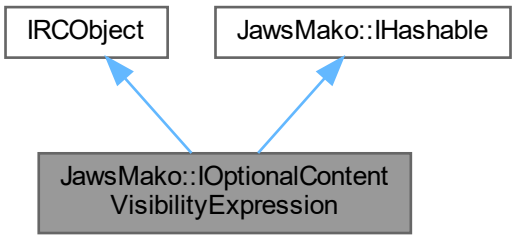
- optionalcontent.h

7.349 JawsMako::IOptionalContentVisibilityExpression Class Reference

An interface representing a PDF 1.6+ visibility expression. Please refer to table 4.4.9 of the PDF 1.7 specification for background and detail.

```
#include <optionalcontent.h>
```

Inheritance diagram for JawsMako::IOptionalContentVisibilityExpression:



Public Types

- enum eVisibilityExpressionOperation { eVEOGroupState , eVEONot , eVEOAnd , eVEOOr }

  The operation of this expression.

Public Member Functions

- virtual IOptionalContentVisibilityExpressionPtr **clone** ()=0

  Clone this expression.
- virtual eVisibilityExpressionOperation **getOperation** () const =0

  Get the type of operation.
- virtual IOptionalContentGroupReferencePtr **getGroup** () const =0

  Get the group for eVEOGroupState operations. If the operation is something else, an exception will be thrown.
- virtual COptionalContentVisibilityExpressionVect **getSubExpressions** () const =0

  Get the subexpressions if the operation is eVEONot, eVEOAnd or eVEOOr. If the operation is something else, an exception will be thrown.
- virtual eOptionalContentVisibility evaluate (const IOptionalContentPtr &optionalContent, eOptionalContentEvent event) const =0

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from JawsMako::IHashable

- virtual uint64 hash () const

  Obtain a 64-bit hash of the receiving object.
- virtual void updateHash (uint64 &hash) const =0

  Update the given hash to include the receiver.

Static Public Member Functions

- static JAWSMAKO_API IOptionalContentVisibilityExpressionPtr **createGroupState** (const IOptional↩ ContentGroupReferencePtr &groupRef)

  Create a "Group State" expression. That is, the result of this expression is true if the given group is visible, and false otherwise.
- static JAWSMAKO_API IOptionalContentVisibilityExpressionPtr **createNot** (const IOptionalContent↩ VisibilityExpressionPtr &subExpression)

  Create a "Not" expression. That is, the result of this expression is the inverse of the visibility of the given sub-expression.
- static JAWSMAKO_API IOptionalContentVisibilityExpressionPtr **createAnd** (const COptionalContent↩ VisibilityExpressionVect &subExpressions)

  Create an "And" expression. That is, the result of this expression is the logical "and" of the result of all the sub-expressions.
- static JAWSMAKO_API IOptionalContentVisibilityExpressionPtr **createOr** (const COptionalContent↩ VisibilityExpressionVect &subExpressions)

  Create an "Or" expression. That is, the result of this expression is the logical "or" of the result of all the sub-expressions.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.349.1 Detailed Description

An interface representing a PDF 1.6+ visibility expression. Please refer to table 4.4.9 of the PDF 1.7 specification for background and detail.

7.349.2 Member Enumeration Documentation

eVisibilityExpressionOperation

```
enum JawsMako::IOptionalContentVisibilityExpression::eVisibilityExpressionOperation
```

The operation of this expression.

Enumerator

eVEOGroupState	The result of this expression is the state of the single group.
eVEONot	The result of this expression is the inverse of the sub-expression.
eVEOAnd	The result of this expression is the logical and of the result of the sub-expressions.
eVEOOr	The result of this expression is the logical or of the result of the sub-expressions.

7.349.3 Member Function Documentation

evaluate()

```
virtual eOptionalContentVisibility JawsMako::IOptionalContentVisibilityExpression::evaluate (
            const IOptionalContentPtr & optionalContent,
            eOptionalContentEvent event ) const  [pure virtual]
```

Determine the result of the expression. The document's optional content must be provided.

The documentation for this class was generated from the following file:

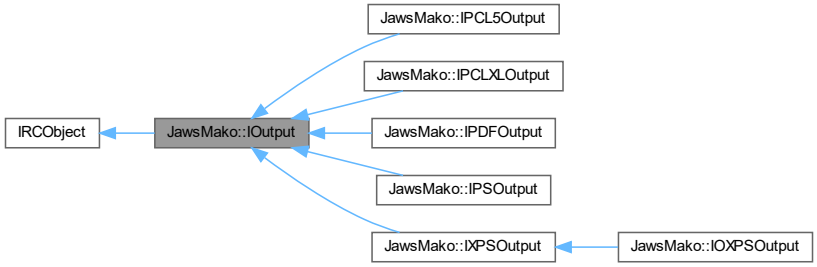
- optionalcontent.h

7.350 JawsMako::IOutput Class Reference

Abstract output sink that can output DOM to a file or stream in a given output format.

`#include <jawsmako.h>`

Inheritance diagram for JawsMako::IOutput:



Public Member Functions

- virtual void **setPreset** (const U8String &preset)=0

  Configure the output according to a general preset. Please see the supplied documentation for details of these presets. The default is "Preserve" which will attempt to produce output as close to the input as possible for the output format. A string value can be used for any parameter and will be converted as necessary.
- virtual void **setParameter** (const U8String &param, const U8String &value)=0

  Apply a key value pair output parameter with a string value. The parameter name is case insensitive. Please refer to the supplied documentation for the details of the available parameters and their ranges.
- virtual void setAllowedPermissionsFlags (uint32 allowedPermissions)=0

  Control whether or not assemblies with certain security permission flags are allowed to be written by this output.
- virtual void writeAssembly (const IDocumentAssemblyPtr &assembly, const U8String &pathToFile)=0

  Write the given document assembly to a file on disk.
- virtual void writeAssembly (const IDocumentAssemblyPtr &assembly, const String &pathToFile)=0

  Write the given document assembly to a file on disk, specified by a wide character string.
- virtual void writeAssembly (const IDocumentAssemblyPtr &assembly, const IOutputStreamPtr &stream)=0

  Write the given document assembly to a stream.
- virtual IOutputWriterPtr openWriter (const IDocumentAssemblyPtr &assembly, const U8String &pathToFile)=0

  Create an output writer for the given assembly, targeting a file on disk. This is designed to allow streaming output, or to deal with situations where an operation would require too much memory to hold an entire edited assembly in memory at once.
- virtual IOutputWriterPtr openWriter (const IDocumentAssemblyPtr &assembly, const String &pathToFile)=0

  Create an output writer for the given assembly, targeting a file on disk. As above, but with the file specified in a wide character string.
- virtual IOutputWriterPtr openWriter (const IDocumentAssemblyPtr &assembly, const IOutputStreamPtr &stream)=0

  Create an output writer for the given assembly, targeting a stream.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IOutputPtr create (const IJawsMakoPtr &jawsMako, eFileFormat format, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create an output for writing source in the given format.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.350.1 Detailed Description

Abstract output sink that can output DOM to a file or stream in a given output format.

7.350.2 Member Function Documentation

create()

```
static JAWSMAKO_API IOutputPtr JawsMako::IOutput::create (
            const IJawsMakoPtr & jawsMako,
            eFileFormat format,
            const IProgressMonitorPtr & progressMonitor = IProgressMonitorPtr() )  [static]
```

Create an output for writing source in the given format.

```
The following formats are currently supported:
- PDF
- XPS
- PostScript
- PCL/XL
- PCL5e
- PCL5c
```

Parameters

jawsMako	The JawsMako instance.
format	The file format of the output file.
progressMonitor	The progress monitor which allows aborting an operation or registering a progress callback.

Returns

IOutputPtr the new output

openWriter() [1/3]

```
virtual IOutputWriterPtr JawsMako::IOutput::openWriter (
            const IDocumentAssemblyPtr & assembly,
            const IOutputStreamPtr & stream )  [pure virtual]
```

Create an output writer for the given assembly, targeting a stream.

Parameters

assembly	The document assembly for output. The assembly need not be populated with documents.
stream	The stream for the output.

Returns

IOutputWriterPtr The new output writer.

openWriter() [2/3]

```
virtual IOutputWriterPtr JawsMako::IOutput::openWriter (
            const IDocumentAssemblyPtr & assembly,
            const String & pathToFile )  [pure virtual]
```

Create an output writer for the given assembly, targeting a file on disk. As above, but with the file specified in a wide character string.

Parameters

assembly	The document assembly for output. The assembly need not be populated with documents.
pathToFile	The path to the output file on disk.

Returns

IOutputWriterPtr The new output writer.

openWriter() [3/3]

```
virtual IOutputWriterPtr JawsMako::IOutput::openWriter (
            const IDocumentAssemblyPtr & assembly,
            const U8String & pathToFile )  [pure virtual]
```

Create an output writer for the given assembly, targeting a file on disk. This is designed to allow streaming output, or to deal with situations where an operation would require too much memory to hold an entire edited assembly in memory at once.

Parameters

assembly	The document assembly for output. The assembly need not be populated with documents.
pathToFile	The path to the output file on disk.

setAllowedPermissionsFlags()

```
virtual void JawsMako::IOutput::setAllowedPermissionsFlags (
            uint32 allowedPermissions )  [pure virtual]
```

Control whether or not assemblies with certain security permission flags are allowed to be written by this output.

If allowed permissions is set to IDOMStandardPDFSecurityInfo::eEverythingAllowed then no checking for assembly permissions will be performed.

Otherwise, the parameter is the or'd combination of IDOMStandardPDFSecurity::ePermissionsFlags that is checked against the operations allowed by the assembly's security (from IAssembly::getSecurityInfo()). The assembly allows an operation corresponding to a flag set in the parameter, then output will be allowed to continue. Otherwise, an IError exception with the error code JM_ERR_ASSEMBLY_WRITE_FORBIDDEN will be thrown.

If the source assembly has no permissions information, or shows that the owner password was supplied, the output will proceed regardless.

The default depends on the individual output. For print-centric outputs (PostScript, PCL5, PCLXL) the default is IDOMStandardPDFSecurityInfo::eHighQualityPrintAllowed. That is, permission-protected jobs that are permitted to print to high resolution will be allowed to convert. For all other outputs, the default is 0, which means that if any permissions information is present and the owner password was not supplied, then no output will be allowed in order to preserve the intentions of the input assembly.

For example, if the requirement is to generate output to be fed to a printer, then calling: `setAllowed↩Permissions(IDOMStandardPDFSecurityInfo::eHighQualityPrintAllowed)` would allow permission-protected jobs that allow high quality output to proceed.

Note: only permissions specified in the standard security handler Revisions 2 and 3 are currently checked.

Equivalent to calling setParameter() with the param name "AllowedPermissions".

writeAssembly() [1/3]

```
virtual void JawsMako::IOutput::writeAssembly (
            const IDocumentAssemblyPtr & assembly,
            const IOutputStreamPtr & stream )  [pure virtual]
```

Write the given document assembly to a stream.

Parameters

assembly	The document assembly to be written.
stream	The stream to write to.

writeAssembly() [2/3]

```
virtual void JawsMako::IOutput::writeAssembly (
            const IDocumentAssemblyPtr & assembly,
            const String & pathToFile )  [pure virtual]
```

Write the given document assembly to a file on disk, specified by a wide character string.

Parameters

assembly	The document assembly to be written.
pathToFile	The path to the output file on disk

writeAssembly() [3/3]

```
virtual void JawsMako::IOutput::writeAssembly (
            const IDocumentAssemblyPtr & assembly,
            const U8String & pathToFile )  [pure virtual]
```

Write the given document assembly to a file on disk.

Parameters

assembly	The document assembly to be written.
pathToFile	The path to the output file on disk.

The documentation for this class was generated from the following file:

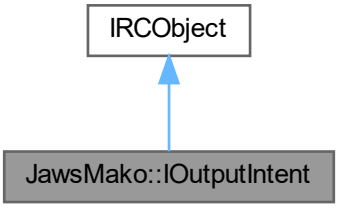
- jawsmako.h

7.351 JawsMako::IOutputIntent Class Reference

Interface class representing a PDF output intent.

`#include <outputintent.h>`

Inheritance diagram for JawsMako::IOutputIntent:



Public Member Functions

- virtual IOutputIntentPtr **clone** ()=0

  Clone the output intent.
- virtual U8String **getSubtype** () const =0

  Retrieve the subtype.
- virtual U8String **getOutputCondition** () const =0

  Retrieve the output condition, if present. An empty string is returned if there is no output condition provided.
- virtual U8String **getOutputConditionIdentifier** () const =0

  Retrieve the output condition identifier.
- virtual U8String **getRegistryName** () const =0

  Retrieve the registry name, if present. An empty string is returned if there is no registry name provided.
- virtual U8String **getInfo** () const =0

  Retrieve the output intent information string, if present. An empty string is returned if there is no info string provided.
- virtual IDOMICCProfilePtr **getProfile** () const =0

  Retrieve the output ICC profile if provided An empty string is returned if there is no output profile provided.
- virtual RawString getCheckSum () const =0

  Retrieve the 16-byte MD5 checksum of the profile, if provided. An empty string is returned if there is no checksum.
- virtual CU8StringVect **getColorantTable** () const =0

  Retrieve the colorant table for the output intent, if provided. An empty string is returned if there is no colorant table.
- virtual uint32 **getProfileVersion** () const =0

  Retrieve the ICC profile version, if provided, as a 32 bit value. 0 will be returned if the version is not provided.
- virtual RawString **getProfileColorSpaceSignature** () const =0

  Retrieve the four-byte color space signature of the IFF profile, if provided. Otherwise, an empty string will be returned.
- virtual U8String **getProfileName** () const =0

  Retrieve the profile name, if provided. Otherwise, an empty string will be returned.
- virtual CFileSpecVect getProfileFileSpecifications () const =0

  Retrieve any file specifications referring to external profiles (IFileSpecAsUrl) or embedded profiles (IFileSpecAs↩ EmbeddedData).

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IOutputIntentPtr **create** (const IJawsMakoPtr &jawsMako, const U8String &subtype, const U8String &outputCondition, const U8String &outputConditionIdentifier, const U8String &registryName, const U8String &info, const IDOMICCProfilePtr &profile=IDOMICCProfilePtr(), const RawString &check↩ Sum=RawString(), const CU8StringVect &colorantTable=CU8StringVect(), uint32 profileVersion=0, const RawString &profileCS=RawString(), const U8String &profileName=U8String(), const CFileSpecVect &file↩ Specs=CFileSpecVect())

  Create an IOutputIntent.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()
  Virtual destructor.

7.351.1 Detailed Description

Interface class representing a PDF output intent.

7.351.2 Member Function Documentation

getCheckSum()

```
virtual RawString JawsMako::IOutputIntent::getCheckSum ( ) const  [pure virtual]
```

Retrieve the 16-byte MD5 checksum of the profile, if provided. An empty string is returned if there is no checksum.

PDF 2.0 Features

getProfileFileSpecifications()

```
virtual CFileSpecVect JawsMako::IOutputIntent::getProfileFileSpecifications ( ) const  [pure virtual]
```

Retrieve any file specifications referring to external profiles (IFileSpecAsUrl) or embedded profiles (IFileSpecAs↩EmbeddedData).

For convenience, the first embedded profile in this list will be returned by getProfile() unless a top-level profile has also been provided.

The documentation for this class was generated from the following file:

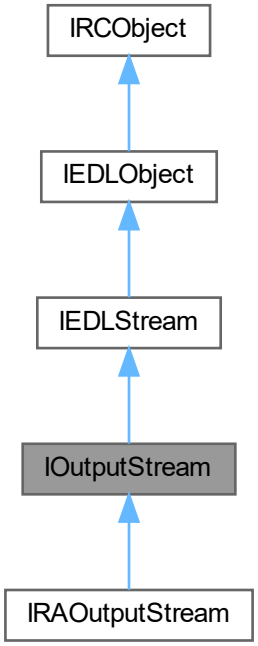
- outputintent.h

7.352 IOutputStream Class Reference

Generic output stream. Abstract base class for output streams.

`#include <edlstream.h>`

Inheritance diagram for IOutputStream:



Public Member Functions

- virtual int32 write (const char *str)
  Perform a write.
- virtual int32 writeFormatted (const char *fmt,...)
  Perform a formatted write as per fprintf().
- virtual void writeFormattedE (const char *fmt,...)
  As writeFormatted(), but throws an exception if the operation fails.
- virtual bool completeWrite (const void *buffer, int32 count)
  Perform a complete write.
- virtual bool completeWrite (const char *str)
  Perform a complete write.
- virtual void completeWriteE (const void *buffer, int32 count)
  As completeWrite(), but throws an exception if the operation fails.
- virtual void completeWriteE (const char *str)
  As completeWrite(), but throws an exception if the operation fails.

Public Member Functions inherited from IEDLStream

- virtual bool isValid () const =0

  Determine stream validity.
- virtual bool open ()

  Opens the stream.
- virtual void **openE** ()=0

  As per open(), but will throw an exception on failure (IEDLError) that for some stream types may contain additional failure information.
- virtual void **close** ()=0

  Closes the stream.
- virtual int64 getPos ()=0

  Get current stream position.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static EDL_API IRAOutputStreamPtr createToFile (IEDLClassFactory *pFactory, const EDLSysString &path, bool append=false)

  Creation function for an IOutputStream for a file on disk. Throws an IEDLError exception on failure.
- static EDL_API IRAOutputStreamPtr createToFile (IEDLClassFactory *pFactory, const EDLString &path, bool append=false)

  Creation function for an IOutputStream for a file on disk. Throws an IEDLError exception on failure.
- static EDL_API IOutputStreamPtr createFromUserWriteFunc (IEDLClassFactory *pFactory, UserStream↩ WriteFunc writeFunc, void *priv)

  Creation function for an IOutputStream from a user function that provides data. Throws an IEDLError exception on failure.
- static EDL_API IOutputStreamPtr createToFlateCompressed (IEDLClassFactory *pFactory, const IOutput↩ StreamPtr &stream, uint32 compressionLevel, bool raw=true)

  Creation routine for an output stream for compressing a flate stream. Throws an IEDLError exception on failure.
- static EDL_API IOutputStreamPtr createToLz4Compressed (IEDLClassFactory *pFactory, const IOutput↩ StreamPtr &stream, bool openSourceStream=true)

Creation routine for an output stream for compressing an lz4 stream. Throws an IEDLError exception on failure. Note: This is not intended for interoperability with other LZ4 formats, but is useful for things like temporary storage.

- static EDL_API int64 copy (const IInputStreamPtr &inStream, const IOutputStreamPtr &outStream)

  Copy a source stream to a destination stream. Opens and closes both the input and output streams. Throws an IEDLError exception on failure.
- static EDL_API int64 writeStream (const IInputStreamPtr &inStream, const IOutputStreamPtr &outStream)

  Write the contents of the given stream to an output stream. Opens and closes the input, but does not open or close the output. Throws an IEDLError exception on failure.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.352.1 Detailed Description

Generic output stream. Abstract base class for output streams.

7.352.2 Member Function Documentation

completeWrite() [1/2]

```
virtual bool IOutputStream::completeWrite (
            const char * str )  [inline], [virtual]
```

Perform a complete write.

Parameters

str	C-style string that contains the text to be written to the output.

Returns

bool True if the write operation was successful, or false if the write could not be completely fulfilled.

completeWrite() [2/2]

```
virtual bool IOutputStream::completeWrite (
            const void * buffer,
            int32 count )  [virtual]
```

Perform a complete write.

Parameters

buffer	Address of buffer.
count	Number of bytes to be written.

Returns

bool True if the write operation was successful, or false if the write could not be completely fulfilled.

completeWriteE() [1/2]

```
virtual void IOutputStream::completeWriteE (
            const char * str )  [inline], [virtual]
```

As completeWrite(), but throws an exception if the operation fails.

Parameters

str	C-style string that contains the text to be written to the output.

completeWriteE() [2/2]

```
virtual void IOutputStream::completeWriteE (
            const void * buffer,
            int32 count )  [virtual]
```

As completeWrite(), but throws an exception if the operation fails.

Parameters

buffer	Address of buffer.
count	Number of bytes to be written.

copy()

```
static EDL_API int64 IOutputStream::copy (
            const IInputStreamPtr & inStream,
            const IOutputStreamPtr & outStream )  [static]
```

Copy a source stream to a destination stream. Opens and closes both the input and output streams. Throws an IEDLError exception on failure.

Parameters

inStream	The source stream.
outStream	The destination stream.

Returns

int64 The number of bytes copied.

createFromUserWriteFunc()

```
static EDL_API IOutputStreamPtr IOutputStream::createFromUserWriteFunc (
            IEDLClassFactory * pFactory,
            UserStreamWriteFunc writeFunc,
            void * priv )  [static]
```

Creation function for an IOutputStream from a user function that provides data. Throws an IEDLError exception on failure.

Parameters

pFactory	The class factory.
writeFunc	A function that when called, will write the data for the stream.
priv	An opaque private pointer that is passed to the readFunc on each call.

Returns

IOutputStreamPtr The new input stream.

createToFile() [1/2]

```
static EDL_API IRAOutputStreamPtr IOutputStream::createToFile (
            IEDLClassFactory * pFactory,
            const EDLString & path,
            bool append = false )  [static]
```

Creation function for an IOutputStream for a file on disk. Throws an IEDLError exception on failure.

Parameters

pFactory	The EDL class factory.
path	Path to the file.
append	Specify what to do if the file exists. Optional • If false the file will be overwritten by new content. Default. • If true new content will be added to the end of the file.

Returns

IOutputStreamPtr The new output stream.

createToFile() [2/2]

```
static EDL_API IRAOutputStreamPtr IOutputStream::createToFile (
```

```
            IEDLClassFactory * pFactory,
            const EDLSysString & path,
            bool append = false )  [static]
```

Creation function for an IOutputStream for a file on disk. Throws an IEDLError exception on failure.

Parameters

pFactory	The EDL class factory.
path	Path to the file.
append	Specify what to do if the file exists. Optional • If false the file will be overwritten by new content. Default. • If true new content will be added to the end of the file.

Returns

IOutputStreamPtr The new output stream.

createToFlateCompressed()

```
static EDL_API IOutputStreamPtr IOutputStream::createToFlateCompressed (
            IEDLClassFactory * pFactory,
            const IOutputStreamPtr & stream,
            uint32 compressionLevel,
            bool raw = true )  [static]
```

Creation routine for an output stream for compressing a flate stream. Throws an IEDLError exception on failure.

Parameters

pFactory	The class factory.
stream	The compressed stream.
compressionLevel	Values are in the range 1 (fastest, least compression) through 9 (slowest, most compression).
raw	Pass true if the flate stream should have no zlib header.

Returns

IOutputStreamPtr The new output stream.

createToLz4Compressed()

```
static EDL_API IOutputStreamPtr IOutputStream::createToLz4Compressed (
            IEDLClassFactory * pFactory,
            const IOutputStreamPtr & stream,
            bool openSourceStream = true )  [static]
```

Creation routine for an output stream for compressing an lz4 stream. Throws an IEDLError exception on failure. Note: This is not intended for interoperability with other LZ4 formats, but is useful for things like temporary storage.

Parameters

pFactory	The class factory.
stream	The compressed stream.
openSourceStream	If true, the source stream will be opened and closed. If false, it is assumed that the stream is already open and the stream will not be closed on completion.

Returns

IOutputStreamPtr The new output stream.

write()

```
virtual int32 IOutputStream::write (
            const char * str )  [inline], [virtual]
```

Perform a write.

Parameters

str	C-style string that contains the text to be written to the output.

Returns

int32 On success, the total number of characters written.

writeFormatted()

```
virtual int32 IOutputStream::writeFormatted (
            const char * fmt,
             ... )  [virtual]
```

Perform a formatted write as per fprintf().

Parameters

fmt	C-style string that contains the text to be written to the output.
...	Additional arguments as specified by *fmt*.

Returns

int32 On success, the total number of characters written.

writeFormattedE()

```
virtual void IOutputStream::writeFormattedE (
            const char * fmt,
             ... )  [virtual]
```

As writeFormatted(), but throws an exception if the operation fails.

Parameters

fmt	C-style string that contains the text to be written to the output.
...	Additional arguments as specified by *fmt*.

writeStream()

```
static EDL_API int64 IOutputStream::writeStream (
            const IInputStreamPtr & inStream,
            const IOutputStreamPtr & outStream )  [static]
```

Write the contents of the given stream to an output stream. Opens and closes the input, but does not open or close the output. Throws an IEDLError exception on failure.

Parameters

inStream	The source stream.
outStream	The destination stream.

Returns

int64 The number of bytes written.

The documentation for this class was generated from the following file:

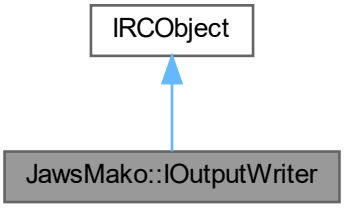
- edlstream.h

7.353 JawsMako::IOutputWriter Class Reference

A writer for writing individual pages and documents to an output in piecemeal fashion.

```
#include <jawsmako.h>
```

Inheritance diagram for JawsMako::IOutputWriter:



Public Member Functions

- virtual void beginDocument (const IDocumentPtr &document)=0

  Begin a document. This document need not be completely populated with pages.
- virtual void **endDocument** ()=0

  Finish a document.
- virtual void writePage (const IPagePtr &page)=0

  Write a page to the output.
- virtual void **finish** ()=0

  Finish output and flush the results.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.353.1 Detailed Description

A writer for writing individual pages and documents to an output in piecemeal fashion.

7.353.2 Member Function Documentation

beginDocument()

```
virtual void JawsMako::IOutputWriter::beginDocument (
            const IDocumentPtr & document )  [pure virtual]
```

Begin a document. This document need not be completely populated with pages.

```
Once this has been invoked, pages may be written using writePage().
```

writePage()

```
virtual void JawsMako::IOutputWriter::writePage (
            const IPagePtr & page )  [pure virtual]
```

Write a page to the output.

```
beginDocument() must have been called to begin the document before any pages
may be written. The page may be discarded after this is called.
```

Parameters

page	The page to write.

The documentation for this class was generated from the following file:

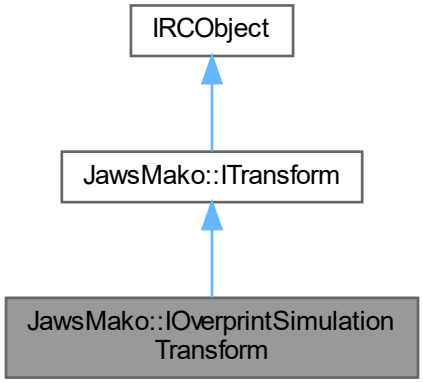
- jawsmako.h

7.354 JawsMako::IOverprintSimulationTransform Class Reference

A transform that modifies DOM such that any overprint present in the DOM will be visible when written or rendered in an environment that does not support overprint.

`#include <transforms.h>`

Inheritance diagram for JawsMako::IOverprintSimulationTransform:



Public Member Functions

- virtual void **setResolution** (uint32 resolution)=0

  Set the target resolution to use when any rendering must be performed. The default is 300dpi. The resolution must be supported by the IRendererTransform.
- virtual void setSimulationColorSpace (const IDOMColorSpacePtr &simulationSpace)=0

  Set the simulation color space.
- virtual void setSimulateBlackDeviceGrayTextOverprint (bool simulate)=0

  Set whether or not overprint simulation should be performed for 100% black DeviceGray text.

Public Member Functions inherited from JawsMako::ITransform

- virtual IDOMBrushPtr transform (const IDOMBrushPtr &brush, eBrushUsage usage=eBUGeneral, const CTransformState &state=CTransformState())=0

  Apply the transform to the given brush, if applicable. These transforms are thread safe.
- virtual IDOMImagePtr transform (const IDOMImagePtr &image, const CTransformState &state=CTransformState())=0

  Apply the transform to the given image, if applicable. These transforms are thread safe.
- virtual IDOMColorPtr transform (const IDOMColorPtr &color, const CTransformState &state=CTransformState())=0

  Apply the transform to the given color, if applicable. These transforms are thread safe.
- virtual IDOMColorSpacePtr transform (const IDOMColorSpacePtr &colorSpace, const CTransformState &state=CTransformState())=0

  Apply the transform to the given colorspace, if applicable. These transforms are thread safe.
- virtual IDOMNodePtr transform (const IDOMNodePtr &node, bool &changed, bool transformChildren=true, const CTransformState &state=CTransformState())=0

  Apply the transform to the given node, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time.
- virtual void transformPage (const IPagePtr &page, bool transformContent=true, bool transform↩ Annotations=true)=0

  Apply the transform to the given page, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time. The transform will also apply to the annotations appearances.
- virtual void **flushCaches** ()=0

  Flush the caches used by the transform. Most transforms cache recently transformed results to improve the performance of repeated transformations of equivalent results. However, it is possible that some cached results may point to entities that no longer exist, such as content inside an XPS file that no longer exists. If you are deleting or replacing files where transforms have been used, it is advisable to invoke this routine to clear the caches.
- virtual void setProgressMonitor (const IProgressMonitorPtr &progressMonitor)=0

  Set the IProgressMonitor object for this transform to allow for monitoring the progress of the transform.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IOverprintSimulationTransformPtr create (const IJawsMakoPtr &jawsMako, const IAbortPtr &abort=IAbortPtr())

  Create the transform.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.354.1 Detailed Description

A transform that modifies DOM such that any overprint present in the DOM will be visible when written or rendered in an environment that does not support overprint.

This will usually involve rendering overprinted content and replacing with an image. The result is a simulation/approximation and may not exactly match the results achieved on a real world printing device where all spot colorants are available as real inks.

Note that in order to perform this operation, all colors will be converted to a CMYK simulation color space (DeviceCMYK by default). Use an IColorConverterTransform after this transform if a non-CMYK output color space is required.

Note that if the simulation CMYK color space is not DeviceCMYK, any DeviceCMYK colors will be treated as if they are using the simulation color space.

As transparency compositing may be affected by color conversion, all transparent content is flattened.

It can only operate on entire IDOMFixedPage or IPage nodes. Any attempt to apply this transform to an individual node or subtree will result in no changes being made. Does not apply to content in annotations.

7.354.2 Member Function Documentation

create()

```
static JAWSMAKO_API IOverprintSimulationTransformPtr JawsMako::IOverprintSimulationTransform
::create (
            const IJawsMakoPtr & jawsMako,
            const IAbortPtr & abort = IAbortPtr() )  [static]
```

Create the transform.

Parameters

jawsMako	The JawsMako instance.
abort	An abort callback handler.

Returns

The new instance.

setSimulateBlackDeviceGrayTextOverprint()

```
virtual void JawsMako::IOverprintSimulationTransform::setSimulateBlackDeviceGrayTextOverprint
(
            bool simulate )  [pure virtual]
```

Set whether or not overprint simulation should be performed for 100% black DeviceGray text.

The default is true.

setSimulationColorSpace()

```
virtual void JawsMako::IOverprintSimulationTransform::setSimulationColorSpace (
            const IDOMColorSpacePtr & simulationSpace )  [pure virtual]
```

Set the simulation color space.

```
The default is DeviceCMYK. The color space that is set will also be used
for DeviceCMYK for any color conversions that take place, ignoring any
IColorManager DeviceCMYK intercept.

The color space must be either DeviceCMYK, or a four-component ICC based
color space.

The default is DeviceCMYK
```

The documentation for this class was generated from the following file:

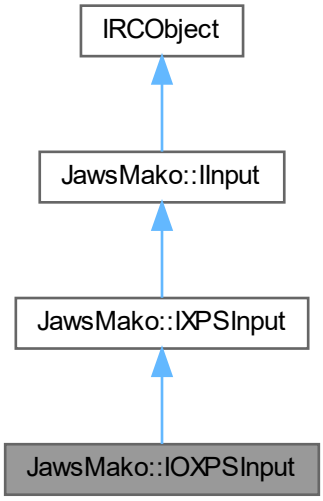
- transforms.h

7.355 JawsMako::IOXPSInput Class Reference

An instance of the JawsMako OpenXPS input class.

`#include <xpsinput.h>`

Inheritance diagram for JawsMako::IOXPSInput:



Static Public Member Functions

- static JAWSMAKO_API IXPSInputPtr create (const IJawsMakoPtr &jawsMako, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create an input for reading source documents in OpenXPS format.

Static Public Member Functions inherited from JawsMako::IXPSInput

- static JAWSMAKO_API IXPSInputPtr create (const IJawsMakoPtr &jawsMako, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create an input for reading source documents in XPS format.

Static Public Member Functions inherited from JawsMako::IInput

- static JAWSMAKO_API IInputPtr create (const IJawsMakoPtr &jawsMako, eFileFormat format, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create an input for reading source documents in the given format. The following formats are currently supported:

Additional Inherited Members

Public Member Functions inherited from JawsMako::IXPSInput

- virtual IDocumentAssemblyPtr openStreaming (const IInputStreamPtr &inputStream)=0

  Open a stream, returning the IDocumentAssembly representing the contents, in streaming mode.

Public Member Functions inherited from JawsMako::IInput

- virtual IDocumentAssemblyPtr open (const U8String &pathToFile)=0

  Open a file on disk, returning the IDocumentAssembly representing the contents.
- virtual IDocumentAssemblyPtr open (const String &pathToFile)=0

  Open a file on disk, returning the IDocumentAssembly representing the contents. Takes a wide character string.
- virtual IDocumentAssemblyPtr open (const IInputStreamPtr &inputStream)=0

  Open a stream, returning the IDocumentAssembly representing the contents.
- virtual void setSequentialMode (bool sequential)=0

  Set/unset sequential mode on this input.
- virtual void setParameter (const U8String &param, const U8String &value)=0

  Apply a key value pair output parameter with a string value. The parameter name is case insensitive. Please refer to the supplied documentation for the details of the available parameters and their ranges.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.355.1 Detailed Description

An instance of the JawsMako OpenXPS input class.

7.355.2 Member Function Documentation

create()

```
static JAWSMAKO_API IXPSInputPtr JawsMako::IOXPSInput::create (
            const IJawsMakoPtr & jawsMako,
            const IProgressMonitorPtr & progressMonitor = IProgressMonitorPtr() )  [static]
```

Create an input for reading source documents in OpenXPS format.

Parameters

jawsMako	The IJawsMako object
progressMonitor	A smart pointer to an IProgressMonitor object which can be NULL if no such object was passed in.

Returns

IXPSInputPtr the XPS input

The documentation for this class was generated from the following file:

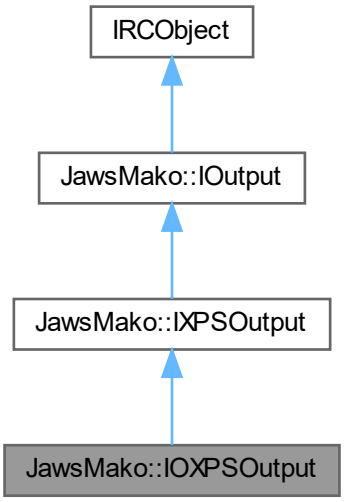
- xpsinput.h

7.356 JawsMako::IOXPSOutput Class Reference

Interface for the OpenXPS IOutput class.

```
#include <xpsoutput.h>
```

Inheritance diagram for JawsMako::IOXPSOutput:



Static Public Member Functions

- static JAWSMAKO_API IXPSOutputPtr create (const IJawsMakoPtr &jawsMako, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create an OpenXPS Output instance.

Static Public Member Functions inherited from JawsMako::IXPSOutput

- static JAWSMAKO_API IXPSOutputPtr create (const IJawsMakoPtr &jawsMako, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create an XPS Output instance.

Static Public Member Functions inherited from JawsMako::IOutput

- static JAWSMAKO_API IOutputPtr create (const IJawsMakoPtr &jawsMako, eFileFormat format, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create an output for writing source in the given format.

Additional Inherited Members

Public Member Functions inherited from JawsMako::IXPSOutput

- virtual void **setCopyExistingXPSPartsWherePossible** (bool copy)=0

  Set whether or not entire XPS parts should be copied from the original assembly source when writing that assembly if the part remains unchanged. This is almost always faster. This defaults to true, and is set to true when using the "↩ Preserve" preset. However, any change to configuration will cause this to be set to false in order to force reprocessing for the configuration to take effect. However this setting may be set to true after other configuration has been changed.

- virtual void setSubsetFonts (bool subset)=0

  Set whether fonts should be subset in the output.
- virtual void setMergeFonts (bool merge)=0

  Set whether or not an attempt will be made to merge disparate subsets of a font into a single font.
- virtual void **setMergeImages** (bool merge=true)=0

  Set if the XPS output should attempt to merge adjacent images. Equivalent to calling setParameter with "Merge↩ AdjacentImages" as the parameter name. The default is true.
- virtual void setRenameFonts (bool rename=true)=0

  Set if the XPS output should rename fonts to aid with problematic print environments.
- virtual void setColorImageMaxResolution (float resolution, float threshold=0.0f, IDOMImageDownsamplerFilter::eDownsampling method=IDOMImageDownsamplerFilter::eBicubic)=0

  Set the desired maximum resolution, threshold and downsampling method for colour images.
- virtual void setGrayImageMaxResolution (float resolution, float threshold=0.0f, IDOMImageDownsamplerFilter::eDownsamplingI method=IDOMImageDownsamplerFilter::eBicubic)=0

  Set the desired maximum resolution, threshold and downsampling method for gray images.
- virtual void setMonoImageMaxResolution (float resolution, float threshold=0.0f, IDOMImageDownsamplerFilter::eDownsampling method=IDOMImageDownsamplerFilter::eSubsample)=0

  Set the desired maximum resolution, threshold and downsampling method for monochrome images.
- virtual void setTargetColorSpace (const IDOMColorSpacePtr &targetSpace)=0

  Set the target color space for the output. The default behaviour is to, where possible, leave the color space of objects unchanged. Equivalent to calling setParameter with the param name "TargetColorSpace" with appropriate values (please refer to documentation).
- virtual void setTargetProfile (const IDOMICCProfilePtr &profile)=0

  Set the target color space for the output using an ICC profile. The default behaviour is to, where possible, leave the color space of objects unchanged. Equivalent to calling setParameter() with the param name "TargetProfile" with the value as the path to the profile.
- virtual void **applyColorConverterTransform** (const IColorConverterTransformPtr &transform)=0

  Apply the given color converter transform to the contents before writing to XPS. This supersedes the target color space parameters described above. This allows for more advanced configuration of the color spaces of the output.
- virtual void setRenderResolution (uint32 resolution)=0

  Set the resolution to use if page content requires rendering in order to be output as XPS. The default is 150dpi. This is affected also by the maximum image resolution parameters. Equivalent to calling setParameter() with param name "RenderResolution" with the value as the desired resolution as value.
- virtual void **applyRendererTransform** (const IRendererTransformPtr &transform)=0

  Apply the given renderer transform to the contents before writing to XPS. This supersedes the target color space parameters described above. This allows for more advanced configuration of rendering.
- virtual void setPreferredColorImageFormat (IImageEncoderTransform::eEncodeFormat format)=0

  Set the desired image format for color images that need to be reencoded for XPS output. The default is eEFAuto. Note: this is advisory only and may not be honoured in all cases if the image cannot be represented in XPS in the desired form. Equivalent to calling setParameter() with the parameter name "ColorImageFormat" with appropriate values (please refer to documentation).
- virtual void setPreferredGrayImageFormat (IImageEncoderTransform::eEncodeFormat format)=0

  Set the desired image format for gray images that need to be reencoded for XPS output. The default is eEFAuto. Note: this is advisory only and may not be honoured in all cases if the image cannot be represented in XPS in the desired form. Equivalent to calling setParameter() with the parameter name "GrayImageFormat" with appropriate values (please refer to documentation).
- virtual void setPreferredMonoImageFormat (IImageEncoderTransform::eEncodeFormat format)=0

  Set the desired image format for monochrome images that need to be reencoded for XPS output. The default is eEFAuto. Note: this is advisory only and may not be honoured in all cases if the image cannot be represented in XPS in the desired form. Equivalent to calling setParameter() with the parameter name "MonoImageFormat" with appropriate values (please refer to documentation).
- virtual void setJPEGQuality (uint8 quality)=0

  Set the JPEG quality to use when encoding images in JPEG format. Equivalent to calling setParameter() with the parameter name "JPEGQuality" and the value being the required quality.
- virtual void **applyEncoderTransform** (const IImageEncoderTransformPtr &transform)=0

  Apply the given image encoder transform to the contents before writing to XPS. This supersedes the image encoding parameters described above. This allows for more advanced configuration of image encoding.

Public Member Functions inherited from JawsMako::IOutput

- virtual void **setPreset** (const U8String &preset)=0

  Configure the output according to a general preset. Please see the supplied documentation for details of these presets. The default is "Preserve" which will attempt to produce output as close to the input as possible for the output format. A string value can be used for any parameter and will be converted as necessary.
- virtual void **setParameter** (const U8String &param, const U8String &value)=0

  Apply a key value pair output parameter with a string value. The parameter name is case insensitive. Please refer to the supplied documentation for the details of the available parameters and their ranges.
- virtual void setAllowedPermissionsFlags (uint32 allowedPermissions)=0

  Control whether or not assemblies with certain security permission flags are allowed to be written by this output.
- virtual void writeAssembly (const IDocumentAssemblyPtr &assembly, const U8String &pathToFile)=0

  Write the given document assembly to a file on disk.
- virtual void writeAssembly (const IDocumentAssemblyPtr &assembly, const String &pathToFile)=0

  Write the given document assembly to a file on disk, specified by a wide character string.
- virtual void writeAssembly (const IDocumentAssemblyPtr &assembly, const IOutputStreamPtr &stream)=0

  Write the given document assembly to a stream.
- virtual IOutputWriterPtr openWriter (const IDocumentAssemblyPtr &assembly, const U8String &pathToFile)=0

  Create an output writer for the given assembly, targeting a file on disk. This is designed to allow streaming output, or to deal with situations where an operation would require too much memory to hold an entire edited assembly in memory at once.
- virtual IOutputWriterPtr openWriter (const IDocumentAssemblyPtr &assembly, const String &pathToFile)=0

  Create an output writer for the given assembly, targeting a file on disk. As above, but with the file specified in a wide character string.
- virtual IOutputWriterPtr openWriter (const IDocumentAssemblyPtr &assembly, const IOutputStreamPtr &stream)=0

  Create an output writer for the given assembly, targeting a stream.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.356.1 Detailed Description

Interface for the OpenXPS IOutput class.

7.356.2 Member Function Documentation

create()

```
static JAWSMAKO_API IXPSOutputPtr JawsMako::IOXPSOutput::create (
            const IJawsMakoPtr & jawsMako,
            const IProgressMonitorPtr & progressMonitor = IProgressMonitorPtr() )  [static]
```

Create an OpenXPS Output instance.

Parameters

jawsMako	The IJawsMako object.
progressMonitor	The progress monitor which allows aborting an operation or registering a progress callback.

Returns

IXPSOutputPtr The XPS output.

The documentation for this class was generated from the following file:

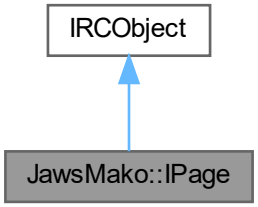
- xpsoutput.h

7.357 JawsMako::IPage Class Reference

A page from an IDocument, allowing high level page management, and providing on-demand access to page contents.

`#include <jawsmako.h>`

Inheritance diagram for JawsMako::IPage:



Public Member Functions

- virtual double **getWidth** ()=0

  Get the page (media) width of the page. This does not require the page contents to be loaded.
- virtual double **getHeight** ()=0

  Get the page (media) height of the page.
- virtual FRect **getCropBox** ()=0

  Get the CropBox for this page. The returned rect is relative to the top left of the page.
- virtual FRect **getBleedBox** ()=0

  Get the BleedBox for this page. The returned rect is relative to the top left of the page.
- virtual FRect **getTrimBox** ()=0

  Get the TrimBox for this page. The returned rect is relative to the top left of the page.
- virtual FRect **getContentBox** ()=0

  Get the ContentBox (also know as the ArtBox) for this page. The returned rect is relative to the top left of the page.

- virtual int32 **getRotate** ()=0

  Get the view rotation for the page, in degrees clockwise.
- virtual void **setRotate** (int32 rotate)=0

  Set the view rotation for the page, in degrees clockwise. Must be a multiple of 90 degrees.
- virtual IDOMFixedPagePtr getContent ()=0

  Return a smart pointer to the IDOMFixedPage, loading it from the source document if necessary.
- virtual IPageRasterPtr getPageRaster ()=0

  Return a smart pointer to the IPageRaster, loading it from the source document if necessary. You can only get either the IDOMFixedPage or the IPageRaster equivalent of the page. This is optional, not all sources support generating a page raster.
- virtual IDOMFixedPagePtr **edit** ()=0

  Mark the page as edited, returning the editable IDOMFixedPage. Do this before attempting to edit the page contents. The markup DOM will be set to incomplete and may be edited in arbitrary ways. This also prevents the fixed page from being purged in low memory conditions. Note: This member may return a different instance than returned by previous invocations of getContent(), however once this routine has been called, subsequent calls to getContent() will return the same instance as that returned here.
- virtual void revert ()=0

  Revert any edits to a page and mark the page as non-edited if possible.
- virtual bool **isLoaded** ()=0

  Has the page been loaded? That is, has the DOM been fetched?
- virtual bool **isEdited** ()=0

  Has the page been edited? That is, has the DOM been fetched?
- virtual void **setMetadataChanged** () const =0

  Let the page know that the content metadata has changed. For example if either of the MediaBox, CropBox, TrimBox or ArtBox has changed.
- virtual void **setContent** (const IDOMFixedPagePtr &content)=0

  Replace the content with the given fixed page.
- virtual void release ()=0

  Release the reference to the page content, if possible. If a page is not edited, and the original source is available, the IPage's reference to the page content will be released. This is useful to release the memory associated with the page content if a page is not to be accessed again. The content may be safely requested again using getContent().
- virtual CAnnotationVect **getAnnotations** ()=0

  Get the annotations for the page, as a vector.
- virtual void **addAnnotation** (const IAnnotationPtr &annotation)=0

  Add the given annotation.
- virtual IAnnotationPtr **findAnnotation** (const IAnnotationReferencePtr &reference)=0

  Find the annotation with the given annotation reference within the page. Throws an IError if the target could not be found.
- virtual void removeAnnotation (uint32 index)=0

  Remove the annotation at the given index.
- virtual void **removeAnnotation** (const IAnnotationPtr &annotation)=0

  Remove the given annotation from the page. If the page is not present, an exception will be thrown.
- virtual void **removeAnnotations** ()=0

  Remove all the annotations from the page.
- virtual void **getTextRuns** (CTextRunVect &runs)=0

  Get the page's text runs. Text present in Tiling brushes will not be returned.
- virtual IDOMCatalogPtr **getCatalog** () const =0

  Get the IDOMCatalog associated with this page.
- virtual DOMid **getPageId** ()=0

  Get the Id of the page.
- virtual COutputIntentVect **getOutputIntents** ()=0

  Get the output intents, if present.
- virtual IDOMJobTkPtr getJobTicket () const =0

Get the page job ticket, if present.

- virtual void **setJobTicket** (const IDOMJobTkPtr &jobTicket)=0

  Set the page job ticket.

- virtual IPagePtr clone ()=0

  Clone an IPage. For edited pages, this will clone the DOM tree.

- virtual IPDFObjectPtr lookupFarReference (const IPDFFarReferencePtr &farReference, IPDFObjectStorePtr &store, const IDocumentPtr &document=IDocumentPtr())=0

  Attempt to find and resolve an indirect far reference to a PDF object. If found, the object store that contained it will be provided. Objects which are controlled by formal Mako APIs may not be reachable, and may be overridden at output.

- virtual IPDFObjectStorePtr getObjectStore ()=0

  Obtain access to the page level object store. Objects which are controlled by formal Mako APIs may not be reachable, and may be overridden at output.

- virtual FBox getMediaBox () const =0

  Obtain the PDF media box for this page.

- virtual void setMediaBox (const FBox &mediaBox)=0

  Set the PDF media box for this page.

- virtual IPageLabelPtr getPageLabel ()=0

  Get the page label.

- virtual void setPageLabel (const IPageLabelPtr &pageLabel)=0

  Set the page label.

- virtual float getUserUnit (bool fixed=false)=0

  Get the PDF UserUnit.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.

- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.

- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IPagePtr create (const IJawsMakoPtr &jawsMako)

  Create an empty page.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.357.1 Detailed Description

A page from an IDocument, allowing high level page management, and providing on-demand access to page contents.

7.357.2 Member Function Documentation

clone()

```
virtual IPagePtr JawsMako::IPage::clone ( )  [pure virtual]
```

Clone an IPage. For edited pages, this will clone the DOM tree.

Returns

IPagePtr the fixed page.

create()

```
static JAWSMAKO_API IPagePtr JawsMako::IPage::create (
            const IJawsMakoPtr & jawsMako )  [static]
```

Create an empty page.

Returns

IPagePtr the new page.

getContent()

```
virtual IDOMFixedPagePtr JawsMako::IPage::getContent ( )  [pure virtual]
```

Return a smart pointer to the IDOMFixedPage, loading it from the source document if necessary.

Returns

IDOMFixedPagePtr the fixed page.

getJobTicket()

```
virtual IDOMJobTkPtr JawsMako::IPage::getJobTicket ( ) const  [pure virtual]
```

Get the page job ticket, if present.

Returns

IDOMJobTkPtr the job ticket, or NULL if not present

getMediaBox()

```
virtual FBox JawsMako::IPage::getMediaBox ( ) const  [pure virtual]
```

Obtain the PDF media box for this page.

The PDF Media box is reflected in the width and height of the page (see IPage::getWidth() and IPage::getHeight() as well as the IDOMFixedPage APIs IDOMFixedPage::getWidth() and IDOMFixedPage::getHeight()). However, PDF Media Boxes also allow the document to specify the origin of the page which IPage and the IDOMFixedPage do not provide.

Here, the PDF media box can be queried. Unlike the other box APIs above, the returned box uses the PDF coordinate system, where x increases to the right, y increases up, and 72 units represents one inch. Note also that Mako transparently handles UserUnit internally, and so the returned box has the UserUnit already incorporated.

Returns

FBox The MediaBox.

getObjectStore()

```
virtual IPDFObjectStorePtr JawsMako::IPage::getObjectStore ( )  [pure virtual]
```

Obtain access to the page level object store. Objects which are controlled by formal Mako APIs may not be reachable, and may be overridden at output.

Returns

IPDFObjectStorePtr The object store for the document.

getPageLabel()

```
virtual IPageLabelPtr JawsMako::IPage::getPageLabel ( )  [pure virtual]
```

Get the page label.

Returns

IPageLabelPtr The page label, or NULL.

getPageRaster()

```
virtual IPageRasterPtr JawsMako::IPage::getPageRaster ( )  [pure virtual]
```

Return a smart pointer to the IPageRaster, loading it from the source document if necessary. You can only get either the IDOMFixedPage or the IPageRaster equivalent of the page. This is optional, not all sources support generating a page raster.

Returns

IPageRasterPtr the page raster.

getUserUnit()

```
virtual float JawsMako::IPage::getUserUnit (
            bool fixed = false )  [pure virtual]
```

Get the PDF UserUnit.

Convenience function to get the UserUnit value that would be written if the page was output to PDF. The UserUnit value represents the size of default user space units, in multiples of 1/72 inch.

PDF versions beginning with 1.6 allow a UserUnit to be defined in the page dictionary which may be used to support larger page dimensions. In PDF versions earlier than PDF 1.6, the size of the default user space unit is fixed at 1/72 inch (UserUnit 1.0).

Parameters

fixed	Boolean value indicating whether the size of the default user space must be fixed (PDF versions earlier than 1.6). When set to true, an IError of code EDL_ERR_PAGE_TOO_LARGE will be thrown if the UserUnit is calculated to be greater than 1.0. The default value is false.

Returns

float The UserUnit value.

lookupFarReference()

```
virtual IPDFObjectPtr JawsMako::IPage::lookupFarReference (
            const IPDFFarReferencePtr & farReference,
            IPDFObjectStorePtr & store,
            const IDocumentPtr & document = IDocumentPtr() )  [pure virtual]
```

Attempt to find and resolve an indirect far reference to a PDF object. If found, the object store that contained it will be provided. Objects which are controlled by formal Mako APIs may not be reachable, and may be overridden at output.

Parameters

farReference	The far reference to resolve.
store	A reference to receive the store where the object was located.
document	If provided, this document will be searched for the far reference if the object is not stored alongside this page.

Returns

IPDFObject The resolved object, or NULL if the resolved object was not found.

release()

```
virtual void JawsMako::IPage::release ( )  [pure virtual]
```

Release the reference to the page content, if possible. If a page is not edited, and the original source is available, the IPage's reference to the page content will be released. This is useful to release the memory associated with the page content if a page is not to be accessed again. The content may be safely requested again using getContent().

If the page content cannot be released, nothing will happen.

removeAnnotation()

```
virtual void JawsMako::IPage::removeAnnotation (
            uint32 index )  [pure virtual]
```

Remove the annotation at the given index.

Parameters

index	The index of the annotation to be removed (0 being the first document).

revert()

```
virtual void JawsMako::IPage::revert ( )  [pure virtual]
```

Revert any edits to a page and mark the page as non-edited if possible.

This may not be used if the page was created as a new page, as there is nothing to revert to. In this case an IError exception will be thrown.

setMediaBox()

```
virtual void JawsMako::IPage::setMediaBox (
            const FBox & mediaBox )  [pure virtual]
```

Set the PDF media box for this page.

The PDF Media box is reflected in the width and height of the page (see IPage::getWidth() and IPage::getHeight() as well as the IDOMFixedPage APIs IDOMFixedPage::getWidth() and IDOMFixedPage::getHeight()). However, PDF Media Boxes also allow the document to specify the origin of the page which IPage and the IDOMFixedPage do not provide.

Here, the PDF media box can be replaced allowing the origin of the page to be changed.

The provided box must be using the PDF coordinate system where x increases to the right, y increases upward, and 72 units represents one inch. Note that Mako internally handles UserUnit automatically, and so the coordinates are in final viewing PDF units, and not the PDF default units.

When the new media box is set, any edited DOM will be modified to reflect the change by offsetting the content, and any annotation rectangles will be updated to suit.

Parameters

mediaBox	The desired MediaBox.

setPageLabel()

```
virtual void JawsMako::IPage::setPageLabel (
            const IPageLabelPtr & pageLabel )  [pure virtual]
```

Set the page label.

Parameters

pageLabel	The page label. Setting a value of NULL specifies that the page label is a continuation from the previous page in the document's labelling range, as described in the PDF reference.

The documentation for this class was generated from the following file:

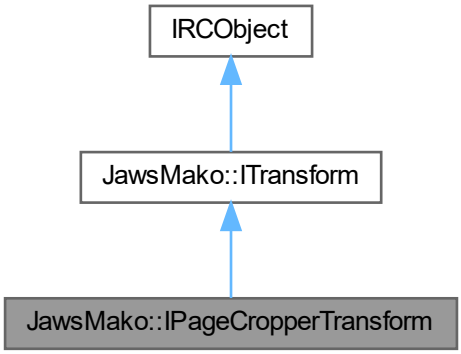
- jawsmako.h

7.358 JawsMako::IPageCropperTransform Class Reference

Very simple transform for cropping pages to one of the standard boxes.

```
#include <transforms.h>
```

Inheritance diagram for JawsMako::IPageCropperTransform:



Public Member Functions

- virtual void **setCropBox** (eBox cropBox)=0

  Sets the box to crop to. The default is the crop box.
- virtual void **setShouldClip** (bool clip)=0

  Sets whether or not the area being cropped should be clipped also. The default is true.

Public Member Functions inherited from JawsMako::ITransform

- virtual IDOMBrushPtr transform (const IDOMBrushPtr &brush, eBrushUsage usage=eBUGeneral, const CTransformState &state=CTransformState())=0

  Apply the transform to the given brush, if applicable. These transforms are thread safe.
- virtual IDOMImagePtr transform (const IDOMImagePtr &image, const CTransformState &state=CTransformState())=0

  Apply the transform to the given image, if applicable. These transforms are thread safe.
- virtual IDOMColorPtr transform (const IDOMColorPtr &color, const CTransformState &state=CTransformState())=0

  Apply the transform to the given color, if applicable. These transforms are thread safe.
- virtual IDOMColorSpacePtr transform (const IDOMColorSpacePtr &colorSpace, const CTransformState &state=CTransformState())=0

  Apply the transform to the given colorspace, if applicable. These transforms are thread safe.
- virtual IDOMNodePtr transform (const IDOMNodePtr &node, bool &changed, bool transformChildren=true, const CTransformState &state=CTransformState())=0

  Apply the transform to the given node, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time.
- virtual void transformPage (const IPagePtr &page, bool transformContent=true, bool transform↩ Annotations=true)=0

  Apply the transform to the given page, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time. The transform will also apply to the annotations appearances.
- virtual void **flushCaches** ()=0

  Flush the caches used by the transform. Most transforms cache recently transformed results to improve the performance of repeated transformations of equivalent results. However, it is possible that some cached results may point to entities that no longer exist, such as content inside an XPS file that no longer exists. If you are deleting or replacing files where transforms have been used, it is advisable to invoke this routine to clear the caches.
- virtual void setProgressMonitor (const IProgressMonitorPtr &progressMonitor)=0

  Set the IProgressMonitor object for this transform to allow for monitoring the progress of the transform.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IPageCropperTransformPtr create (const IJawsMakoPtr &jawsMako, const IAbortPtr &abort=IAbortPtr())

  Create the transform.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.358.1 Detailed Description

Very simple transform for cropping pages to one of the standard boxes.

7.358.2 Member Function Documentation

create()

```
static JAWSMAKO_API IPageCropperTransformPtr JawsMako::IPageCropperTransform::create (
            const IJawsMakoPtr & jawsMako,
            const IAbortPtr & abort = IAbortPtr() )  [static]
```

Create the transform.

Parameters

jawsMako	The JawsMako instance.
abort	An abort callback handler.

Returns

The new instance.

The documentation for this class was generated from the following file:

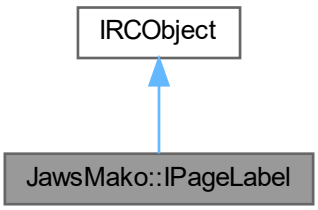
- transforms.h

7.359 JawsMako::IPageLabel Class Reference

Interface class representing a PDF page label.

`#include <pagelabels.h>`

Inheritance diagram for JawsMako::IPageLabel:



Public Types

- enum eLabelStyle {
  eNone = 0 , eDecimal , eRomanUppercase , eRomanLowercase ,
  eLetterUppercase , eLetterLowercase }

  Page label numbering styles.

Public Member Functions

- virtual eLabelStyle getStyle ()=0

  Get the page label numbering style.
- virtual void setStyle (eLabelStyle style)=0

  Set the page label numbering style.
- virtual U8String getPrefix ()=0

  Get the page label prefix.
- virtual void setPrefix (const U8String &prefix)=0

  Set the page label prefix.
- virtual uint32 getNumber ()=0

  Get the page label number.
- virtual void setNumber (uint32 number)=0

  Set the page label number.
- virtual IPageLabelPtr clone ()=0

  Clone the page label.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IPageLabelPtr create (const IJawsMakoPtr &jawsMako)

  Create an IPageLabel.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.359.1 Detailed Description

Interface class representing a PDF page label.

7.359.2 Member Enumeration Documentation

eLabelStyle

enum JawsMako::IPageLabel::eLabelStyle

Page label numbering styles.

Enumerator

eNone	No style.
eDecimal	Decimal Arabic Numerals.
eRomanUppercase	Uppercase Roman Numerals.
eRomanLowercase	Lowercase Roman Numerals.
eLetterUppercase	Uppercase letters.
eLetterLowercase	Lowercase letters.

7.359.3 Member Function Documentation

clone()

```
virtual IPageLabelPtr JawsMako::IPageLabel::clone ( )  [pure virtual]
```

Clone the page label.

Returns

IPageLabelPtr A smart pointer to the IPageLabel object.

create()

```
static JAWSMAKO_API IPageLabelPtr JawsMako::IPageLabel::create (
            const IJawsMakoPtr & jawsMako )  [static]
```

Create an IPageLabel.

Parameters

jawsMako	The IJawsMako object

Returns

IPageLabelPtr A smart pointer to the IPageLabel object.

getNumber()

```
virtual uint32 JawsMako::IPageLabel::getNumber ( )  [pure virtual]
```

Get the page label number.

Returns the value of the numeric portion to be used for the page label.

The default is 1.

Returns

uint32 The label number.

getPrefix()

```
virtual U8String JawsMako::IPageLabel::getPrefix ( )  [pure virtual]
```

Get the page label prefix.

Returns the prefix to be used for the page label.

The default is an empty string, which specifies that the label has no prefix.

Returns

U8String The label prefix.

getStyle()

```
virtual eLabelStyle JawsMako::IPageLabel::getStyle ( )  [pure virtual]
```

Get the page label numbering style.

Returns the numbering style to be used for the numeric portion of the page label.

The default is eNone, which specifies that the label has no numeric portion.

Returns

eLabelStyle The label numbering style.

setNumber()

```
virtual void JawsMako::IPageLabel::setNumber (
            uint32 number )  [pure virtual]
```

Set the page label number.

Parameters

number	The label number, which must be greater than or equal to 1.

setPrefix()

```
virtual void JawsMako::IPageLabel::setPrefix (
            const U8String & prefix )  [pure virtual]
```

Set the page label prefix.

Parameters

prefix	The label prefix.

setStyle()

```
virtual void JawsMako::IPageLabel::setStyle (
            eLabelStyle style )  [pure virtual]
```

Set the page label numbering style.

Parameters

style	The label numbering style.

The documentation for this class was generated from the following file:

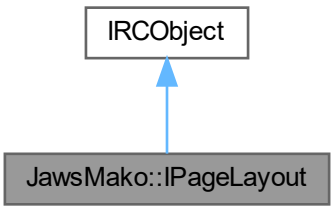
- pagelabels.h

7.360 JawsMako::IPageLayout Class Reference

Analyze the layout of a FixedPage, grouping together text deemed to be in horizontal and/or vertical blocks. Useful for text search and selection.

```
#include <text.h>
```

Inheritance diagram for JawsMako::IPageLayout:



Public Member Functions

- virtual IDOMFixedPagePtr **getFixedPage** () const =0

  Get the FixedPage being processed.
- virtual void **analyze** (ePageAnalysis analysisToPerform=ePAAll)=0

  Process the page find the blocks of text. Can optionally perform each analysis phase independently (which can be useful when debugging).

- virtual String **getLayoutInfo** () const =0

  Get a textual description of the page content, useful for debugging purposes.
- virtual IPageLayoutDataPtr **getLayoutData** () const =0

  Get a processed representation of the page content.
- virtual IPageLayoutNodeCollection **getLayoutNodeCollection** () const =0

  Get a flat collection of page content nodes.
- virtual String **getPageText** () const =0

  Return all page text.
- virtual void setVirtualSpaceThreshold (double virtualSpaceThreshold)=0

  Set the virtual space threshold.
- virtual void setMultipleSpaceMode (bool multipleSpaces)=0

  Set the multiple space mode for inserting virtual spaces.
- virtual void setLineSpacingThreshold (double threshold)=0

  Set the threshold for spacing between successive text lines when concatenating them into a text run.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IPageLayoutPtr create (IEDLClassFactory *factory, const IDOMFixedPagePtr &page)

  Creation function for an IPageLayout, a fixed page layout analyser Throws an IEDLError on failure.
- static JAWSMAKO_API IPageLayoutPtr create (IEDLClassFactory *factory, const CTextRunVect &runs)

  Creation function for an IPageLayout, a fixed page layout analyser Throws an IEDLError on failure.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.360.1 Detailed Description

Analyze the layout of a FixedPage, grouping together text deemed to be in horizontal and/or vertical blocks. Useful for text search and selection.

7.360.2 Member Function Documentation

create() [1/2]

```
static JAWSMAKO_API IPageLayoutPtr JawsMako::IPageLayout::create (
            IEDLClassFactory * factory,
            const CTextRunVect & runs )  [static]
```

Creation function for an IPageLayout, a fixed page layout analyser Throws an IEDLError on failure.

Parameters

factory	The factory to use
runs	The arrary of text runs to be analysed

Returns

IPageLayoutPtr The new page layout analyser

create() [2/2]

```
static JAWSMAKO_API IPageLayoutPtr JawsMako::IPageLayout::create (
            IEDLClassFactory * factory,
            const IDOMFixedPagePtr & page )  [static]
```

Creation function for an IPageLayout, a fixed page layout analyser Throws an IEDLError on failure.

Parameters

factory	The factory to use.
page	The fixed page to be analysed

Returns

IPageLayoutPtr The new page layout analyser

setLineSpacingThreshold()

```
virtual void JawsMako::IPageLayout::setLineSpacingThreshold (
            double threshold )  [pure virtual]
```

Set the threshold for spacing between successive text lines when concatenating them into a text run.

Parameters

threshold	The distance expressed in 1/96th inch between the bottom edge of a given line of text and the top edge of the text line immediately below it. The value may be negative to allow text lines that are very closely spaced or even overlapping vertically to be recognized as separate lines. The default value is 0.0.

setMultipleSpaceMode()

```
virtual void JawsMako::IPageLayout::setMultipleSpaceMode (
            bool multipleSpaces )  [pure virtual]
```

Set the multiple space mode for inserting virtual spaces.

Parameters

multipleSpaces	The flag determining if inserting multiple virtual spaces or a single space. If true was specified, multiple virtual spaces would be inserted, If false was specified, a single space would be inserted to be compatible with Acrobat behaviour. The default is true.

setVirtualSpaceThreshold()

```
virtual void JawsMako::IPageLayout::setVirtualSpaceThreshold (
            double virtualSpaceThreshold )  [pure virtual]
```

Set the virtual space threshold.

Parameters

virtualSpaceThreshold	The threshold determining if inserting virtual spaces. The range of this value is limited between 0.0 ∼ 1.0, and the default is 0.75.

The documentation for this class was generated from the following file:

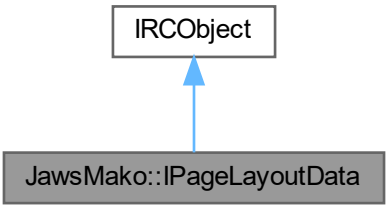
- text.h

7.361 JawsMako::IPageLayoutData Class Reference

Provides a representation of the analyzed page layout by organizing and allowing access to collections of IPage↩ LayoutNodes.

```
#include <text.h>
```

Inheritance diagram for JawsMako::IPageLayoutData:



Public Member Functions

- virtual IPageLayoutNodeCollection **getColumn** (uint32 columnNumber)=0

  Get a collection of IPageLayoutNodes, representing the content of the specified column number. A columnNumber of zero will return the root content.
- virtual uint32 **getNumberOfColumns** () const =0

  Get the number of columns (this includes the root item)

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IPageLayoutDataPtr create (IPageLayoutNodeCollection data)

  Creation function for IPageLayoutData that provides a representation of the analyzed page layout, by organizing and allowing access to collections of IPageLayoutNodes.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.361.1 Detailed Description

Provides a representation of the analyzed page layout by organizing and allowing access to collections of IPage↩ LayoutNodes.

7.361.2 Member Function Documentation

create()

```
static JAWSMAKO_API IPageLayoutDataPtr JawsMako::IPageLayoutData::create (
            IPageLayoutNodeCollection data )  [static]
```

Creation function for IPageLayoutData that provides a representation of the analyzed page layout, by organizing and allowing access to collections of IPageLayoutNodes.

Parameters

data	The collection of IPageLayotNodes

Returns

IPageLayoutDataPtr The new representation

The documentation for this class was generated from the following file:

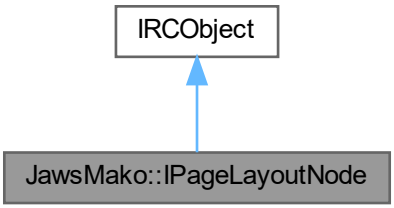
- text.h

7.362 JawsMako::IPageLayoutNode Class Reference

Simple data type representing a part of an analyzed page.

`#include <text.h>`

Inheritance diagram for JawsMako::IPageLayoutNode:



Public Member Functions

- virtual String **getContent** () const =0

  Get the unicode string content of the node, if it's of type ePLTTextRun otherwise returns an empty string.
- virtual IDOMGlyphsPtr **getContentGlyphs** () const =0

  Get the IDOMGlyphsNode that represents the content of the node, if it's of type ePLTTextRun otherwise returns a NULL.
- virtual FRect **getPageBounds** () const =0

  Gets the page bounds of the node.
- virtual ePageLayoutType **getType** () const =0

  Get the type of node.
- virtual uint32 **getColumnNumber** () const =0

  Get the column number that the node belongs to. Zero is the root element.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IPageLayoutNodePtr create (FRect rect, ePageLayoutType type, String content, const IDOMGlyphsPtr &contentGlyphs, uint32 columnNumber)

  Creation function for an IPageLayoutNode, a simple data type representing a part of an analyzed page.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()
  Virtual destructor.

7.362.1 Detailed Description

Simple data type representing a part of an analyzed page.

7.362.2 Member Function Documentation

create()

```
static JAWSMAKO_API IPageLayoutNodePtr JawsMako::IPageLayoutNode::create (
            FRect rect,
            ePageLayoutType type,
            String content,
            const IDOMGlyphsPtr & contentGlyphs,
            uint32 columnNumber )  [static]
```

Creation function for an IPageLayoutNode, a simple data type representing a part of an analyzed page.

Parameters

rect	The bounds of the text area
type	The layout node type
content	The Unicode string content
contentGlyphs	The glyphs node from which the content was obtained
columnNumber	The zero-indexed column number

Returns

IPageLayoutNodePtr The new node

The documentation for this class was generated from the following file:

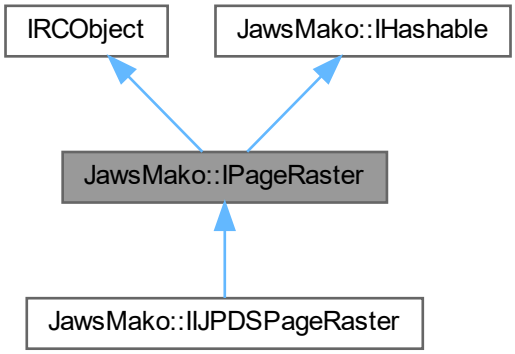
- text.h

7.363 JawsMako::IPageRaster Class Reference

A rasterized equivalent of a page from an IDocument.

```
#include <jawsmako.h>
```

Inheritance diagram for JawsMako::IPageRaster:



Public Member Functions

- virtual uint32 **getWidth** () const =0

  Get the width of the raster image.
- virtual uint32 **getHeight** () const =0

  Get the height of the raster image.
- virtual uint32 **getResX** () const =0

  Get the resolution of the raster image along the X-axis.
- virtual uint32 **getResY** () const =0

  Get the resolution of the raster image along the Y-axis.
- virtual uint32 **getBPC** () const =0

  Get the number of bits per component.
- virtual uint32 **getNumComponents** () const =0

  Get the number of component per pixel.
- virtual uint32 **getRawBytesPerRow** () const =0

  Get the number of bytes per row.
- virtual bool **isBlank** () const =0

  Return true if the page is unmarked.
- virtual uint8 * **getFrameBuffer** () const =0

  Get the address the raw frame buffer.
- virtual IDOMImagePtr getAsDOMImage () const =0

  Return an IDOMImage equivalent of the raster image.
- virtual void **releaseFrameBuffer** ()=0

  Release the frame buffer for this raster.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from JawsMako::IHashable

- virtual uint64 hash () const

  Obtain a 64-bit hash of the receiving object.
- virtual void updateHash (uint64 &hash) const =0

  Update the given hash to include the receiver.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.363.1 Detailed Description

A rasterized equivalent of a page from an IDocument.

7.363.2 Member Function Documentation

getAsDOMImage()

```
virtual IDOMImagePtr JawsMako::IPageRaster::getAsDOMImage ( ) const  [pure virtual]
```

Return an IDOMImage equivalent of the raster image.

Returns

IDOMImagePtr the dom image.

The documentation for this class was generated from the following file:

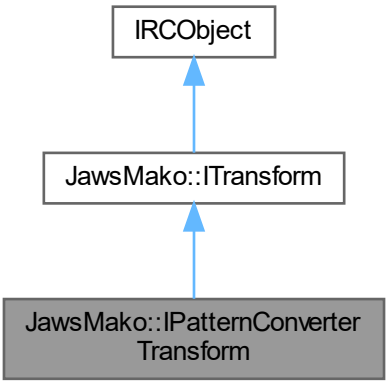
- jawsmako.h

7.364 JawsMako::IPatternConverterTransform Class Reference

Transform to convert PDF/PS tiling/shading patterns to XPS- compatible forms where possible.

`#include <transforms.h>`

Inheritance diagram for JawsMako::IPatternConverterTransform:



Public Member Functions

- virtual void **setConvertType2ShadingPatterns** (bool convert)=0

  Set whether or not Type 2 shades should be converted. The default is true.
- virtual void **setMaxType2Samples** (uint32 maxSamples)=0

  If converting Type 2 shades, what is the maximum number of samples to take from the function to generate gradient stops. The default is 255.
- virtual void **setConvertType3ShadingPatterns** (bool convert)=0

  Set whether or not Type 3 shades should be converted. The default is true.
- virtual void **setMaxType3Samples** (uint32 maxSamples)=0

  If converting Type 3 shades, what is the maximum number of samples to take from the function to generate gradient stops. The default is 255.

Public Member Functions inherited from JawsMako::ITransform

- virtual IDOMBrushPtr transform (const IDOMBrushPtr &brush, eBrushUsage usage=eBUGeneral, const CTransformState &state=CTransformState())=0

  Apply the transform to the given brush, if applicable. These transforms are thread safe.
- virtual IDOMImagePtr transform (const IDOMImagePtr &image, const CTransformState &state=CTransformState())=0

  Apply the transform to the given image, if applicable. These transforms are thread safe.
- virtual IDOMColorPtr transform (const IDOMColorPtr &color, const CTransformState &state=CTransformState())=0

  Apply the transform to the given color, if applicable. These transforms are thread safe.
- virtual IDOMColorSpacePtr transform (const IDOMColorSpacePtr &colorSpace, const CTransformState &state=CTransformState())=0

  Apply the transform to the given colorspace, if applicable. These transforms are thread safe.

- virtual IDOMNodePtr transform (const IDOMNodePtr &node, bool &changed, bool transformChildren=true, const CTransformState &state=CTransformState())=0

  Apply the transform to the given node, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time.
- virtual void transformPage (const IPagePtr &page, bool transformContent=true, bool transform↩ Annotations=true)=0

  Apply the transform to the given page, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time. The transform will also apply to the annotations appearances.
- virtual void **flushCaches** ()=0

  Flush the caches used by the transform. Most transforms cache recently transformed results to improve the performance of repeated transformations of equivalent results. However, it is possible that some cached results may point to entities that no longer exist, such as content inside an XPS file that no longer exists. If you are deleting or replacing files where transforms have been used, it is advisable to invoke this routine to clear the caches.
- virtual void setProgressMonitor (const IProgressMonitorPtr &progressMonitor)=0

  Set the IProgressMonitor object for this transform to allow for monitoring the progress of the transform.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IPatternConverterTransformPtr create (const IJawsMakoPtr &jawsMako, const IAbortPtr &abort=IAbortPtr())

  Create the transform.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.364.1 Detailed Description

Transform to convert PDF/PS tiling/shading patterns to XPS- compatible forms where possible.

7.364.2 Member Function Documentation

create()

```
static JAWSMAKO_API IPatternConverterTransformPtr JawsMako::IPatternConverterTransform::create
(
            const IJawsMakoPtr & jawsMako,
            const IAbortPtr & abort = IAbortPtr() )  [static]
```

Create the transform.

Parameters

jawsMako	The JawsMako instance.
abort	An abort callback handler.

Returns

The new instance.

The documentation for this class was generated from the following file:

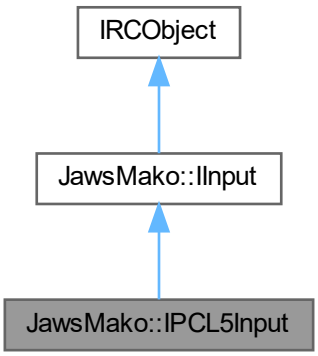
- transforms.h

7.365 JawsMako::IPCL5Input Class Reference

An instance of the JawsMako PCL5 input class.

`#include <pcl5input.h>`

Inheritance diagram for JawsMako::IPCL5Input:



Public Member Functions

- virtual void setRopResolution (uint32 resolution)=0

  Set the resolution to be used when flattening ROPs.
- virtual void setFlattenRops (bool flatten)=0

  Set whether or not ROPs should be flattened.
- virtual void setDefaultPaperSize (const U8String &paperSize)=0

  Set the default paper size.
- virtual void setDefaultLandscape (bool landscape)=0

  Set the default orientation.
- virtual void setDefaultCopies (uint32 copies)=0

Set the default number of per-page copies.

- virtual void setDefaultDuplex (bool duplex)=0

  Set whether or not duplex should be set by default.
- virtual void setDefaultDuplexBindingMode (IPJLParser::eDuplexBindingMode bindingMode)=0

  Set the default binding edge for duplexing.
- virtual void setDefaultManualFeed (bool manualFeed)=0

  Set whether or not manual feed should be set by default.
- virtual void setDefaultsFromPjl (IPJLParserPtr &pjlParser)=0

  Take initialisation data from the given PJL parser.
- virtual void enableUnencapsulatedMode (bool unencapsulated)=0

  Enable or disable unencapsulated mode.
- virtual void setMediaHandler (IMediaHandler *mediaHandler)=0

  Set the IMediaHandler instance to use to handle unsatisfied media requests.
- virtual void setIgnorePrescribe (bool ignorePrescribe)=0

  Set whether or not to ignore Kyocera PRESCRIBE commands.
- virtual void setPermanentResourceStore (IRCObjectPtr &permanentResourceStore)=0

  Set a permanent resource store.

Public Member Functions inherited from JawsMako::IInput

- virtual IDocumentAssemblyPtr open (const U8String &pathToFile)=0

  Open a file on disk, returning the IDocumentAssembly representing the contents.
- virtual IDocumentAssemblyPtr open (const String &pathToFile)=0

  Open a file on disk, returning the IDocumentAssembly representing the contents. Takes a wide character string.
- virtual IDocumentAssemblyPtr open (const IInputStreamPtr &inputStream)=0

  Open a stream, returning the IDocumentAssembly representing the contents.
- virtual void setSequentialMode (bool sequential)=0

  Set/unset sequential mode on this input.
- virtual void setParameter (const U8String &param, const U8String &value)=0

  Apply a key value pair output parameter with a string value. The parameter name is case insensitive. Please refer to the supplied documentation for the details of the available parameters and their ranges.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IPCL5InputPtr create (const IJawsMakoPtr &jawsMako, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create an input for reading source documents in PCL5 format.

Static Public Member Functions inherited from JawsMako::IInput

- static JAWSMAKO_API IInputPtr create (const IJawsMakoPtr &jawsMako, eFileFormat format, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create an input for reading source documents in the given format. The following formats are currently supported:

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.365.1 Detailed Description

An instance of the JawsMako PCL5 input class.

7.365.2 Member Function Documentation

create()

```
static JAWSMAKO_API IPCL5InputPtr JawsMako::IPCL5Input::create (
            const IJawsMakoPtr & jawsMako,
            const IProgressMonitorPtr & progressMonitor = IProgressMonitorPtr() )  [static]
```

Create an input for reading source documents in PCL5 format.

Parameters

jawsMako	The IJawsMako object
progressMonitor	A smart pointer to an IProgressMonitor object which can be NULL if no such object was passed in.

Returns

IPCL5InputPtr the PCL5 input

enableUnencapsulatedMode()

```
virtual void JawsMako::IPCL5Input::enableUnencapsulatedMode (
            bool unencapsulated )  [pure virtual]
```

Enable or disable unencapsulated mode.

```
Setting this mode to true changes the parser behaviour
as follows.
```

```
- The input stream is not opened; it is assumed to point to
  the start of a PCL5 stream.
- The PJL parser is not used. If PJL is encountered an
  error will be thrown.
- The PCL5 interpreter will not consume data after the point the
  interpreter either exhausts the stream or returns to PJL.
  After requesting a document after the last document
  in the session, or by requesting the number of documents) the
  stream will be positioned at the point that the PCL5
  interpreter exited.

Whereas the default (false) will:

- open() the input stream.
- use the built-in PJL parser when PJL is encountered.
- process all PCL5 sessions present in the stream, even
  if they are interupted by returning to PJL.
```

setDefaultCopies()

```
virtual void JawsMako::IPCL5Input::setDefaultCopies (
            uint32 copies )  [pure virtual]
```

Set the default number of per-page copies.

```
The default is 1. Must be greater than zero.
```

setDefaultDuplex()

```
virtual void JawsMako::IPCL5Input::setDefaultDuplex (
            bool duplex )  [pure virtual]
```

Set whether or not duplex should be set by default.

```
The default is false.
```

setDefaultDuplexBindingMode()

```
virtual void JawsMako::IPCL5Input::setDefaultDuplexBindingMode (
            IPJLParser::eDuplexBindingMode bindingMode )  [pure virtual]
```

Set the default binding edge for duplexing.

```
The default is long edge.
```

setDefaultLandscape()

```
virtual void JawsMako::IPCL5Input::setDefaultLandscape (
            bool landscape )  [pure virtual]
```

Set the default orientation.

```
Pass true to set landscape, or false to set portrait.
The default is portrait.
```

setDefaultManualFeed()

```
virtual void JawsMako::IPCL5Input::setDefaultManualFeed (
            bool manualFeed )  [pure virtual]
```

Set whether or not manual feed should be set by default.

```
      The default is false.
```

setDefaultPaperSize()

```
virtual void JawsMako::IPCL5Input::setDefaultPaperSize (
            const U8String & paperSize )  [pure virtual]
```

Set the default paper size.

```
      The paper size is set using a string. To take effect the paper
      size string must match one of the following standard strings:

      - letter
      - legal
      - a4
      - exec
      - executive
      - ledger
      - a3
      - monarchenvelope
      - c5
      - dl
      - jisb4
      - jisb5
      - a5
      - a6
      - tabloid
      - ofuku

      The matching is case-insensitive.

      The default page size is a4
```

setDefaultsFromPjl()

```
virtual void JawsMako::IPCL5Input::setDefaultsFromPjl (
            IPJLParserPtr & pjlParser )  [pure virtual]
```

Take initialisation data from the given PJL parser.

```
      Uses the environment in the given PJL parser to set
      defaults in the PCL5 input. Any previously set
      defaults will be ignored.

      This is especially useful with unencapsulated mode,
      where PJL parsing happens externally to the input.
```

setFlattenRops()

```
virtual void JawsMako::IPCL5Input::setFlattenRops (
            bool flatten )  [pure virtual]
```

Set whether or not ROPs should be flattened.

Please see setRopResolution() for information on ROPs.

```
If false, the input will not flatten ROPs. As a result the DOM that
is produced will not be visually correct. Instead, the ROP components
will appear in the DOM as distinct objects grouped in a special
IDOMGroup.

This mode is useful if only information about the job is required.
ROPs may be expensive especially for pathological cases which use
large amounts of overdraw. If we merely need to know what text is on
a page, the number of pages, whether the content is black and white or
color etc, then flattening ROPs may significantly slow processing for
no benefit. Setting this parameter false will avoid that processing.

The default is true; ROPs will be flattened.

If false, the ROPs will appear in the DOM inside groups with properties
denoting their purpose. The operands for a ROP will be stored inside an
IDOMGroup that is tagged with the property (see IDOMNode::getProperty())
"RopGroup". Inside that group will be up to two nodes that represent the
ROP operands. These will be tagged with the property "RopSource" for the
source operand, and "RopPaint" for the paint operand. The value of these
properties is irrelevant; only their presence is important.
```

Parameters

flatten	True to flatten ROPs, false to avoid flattening ROPs.

setIgnorePrescribe()

```
virtual void JawsMako::IPCL5Input::setIgnorePrescribe (
            bool ignorePrescribe )  [pure virtual]
```

Set whether or not to ignore Kyocera PRESCRIBE commands.

```
When enabled, an attempt will be made to ignore Kyocera PRESCRIBE
commands in PCL5 streams by skipping data from the start sequence
!R! to the end sequence EXIT;

Note that !R! may appear in PCL5 data as text and so this parameter
should only be enabled when opening PCL5 streams with Kyocera PRESCRIBE.

The default is false.
```

setMediaHandler()

```
virtual void JawsMako::IPCL5Input::setMediaHandler (
            IMediaHandler * mediaHandler )  [pure virtual]
```

Set the IMediaHandler instance to use to handle unsatisfied media requests.

Note that this routine does not take ownership of the passed pointer, which must remain allocated until after the input is complete.

setPermanentResourceStore()

```
virtual void JawsMako::IPCL5Input::setPermanentResourceStore (
            IRCObjectPtr & permanentResourceStore )  [pure virtual]
```

Set a permanent resource store.

```
Set a reference to an object used for storing PCL5 permanent resources.

PCL5 allows certain resources, such as fonts, symbol sets and macros
to be designated as permanent, to prevent the printer from deleting
them during a printer reset.  This function allows reuse of these
resources with a new IPCL5Input instance.

This is especially useful when using a PJL parser with unencapsulated
IPCL5Input instances, where a PJL stream may define separate PCL5 jobs
that share permanent resources.
```

Parameters

permanentResourceStore	Reference to a permanent resource store. Setting the reference to a NULL object sets the object to a newly created store, which will be populated with any permanent resources.

A non-null value sets the permanent resource store to an existing store. An exception will be thrown if the value is not NULL and is not a valid resource store obtained from a previous call to setPermanentResourceStore().

setRopResolution()

```
virtual void JawsMako::IPCL5Input::setRopResolution (
            uint32 resolution )  [pure virtual]
```

Set the resolution to be used when flattening ROPs.

```
PCL5, just as PCL/XL, features a method of using bitwise
operations to composite pixels from patterns and images
with content already on the page. These are known as ROPs
(Raster OPerations). These are not compatible with the DOM
and most other PDLs, and in many situations require
rendering at the input stage.

This parameter controls the rendering resolution. The default
is 600dpi.
```

Parameters

resolution	The ROP rendering resolution to use.

The documentation for this class was generated from the following file:

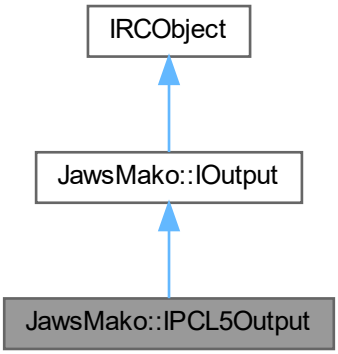
- pcl5input.h

7.366 JawsMako::IPCL5Output Class Reference

Interface for the PCL5 IOutput class.

`#include <pcl5output.h>`

Inheritance diagram for JawsMako::IPCL5Output:



Public Types

- enum ePCL5Version

  Supported versions.
- enum ePCL5ImageCompression

  Available image compression schemes.

Public Member Functions

- virtual void setVersion (ePCL5Version version)=0

  Set the PCL version. The default is ePCL5c. Equivalent to calling setParameter() with the parameter name "PCL5↩ Version" using the string value "pcl5e" or "pcl5c".
- virtual void setIncludeMarginsWhenSelectingPaper (bool include)=0

  Set whether paper selection for the PCL output should take into account the non-printable margins of the paper when selecting a paper size.
- virtual void setResolution (uint32 resolution)=0

  Set the PCL resolution The default is 600. Equivalent to calling setParameter() with the parameter name "Resolution".
- virtual void setImageCompression (ePCL5ImageCompression compression)=0

  Set the desired image compression method. The default is delta row. Equivalent to calling setParameter() with the parameter name "ImageCompression" using the string value "None", "RLE", or "DeltaRow".
- virtual void setMediaSource (uint32 mediaSource)=0

  Set the desired media source The default is 7. Equivalent to calling setParameter() with the parameter name name "MediaSource".
- virtual void setOpenStream (bool open)=0

  Set whether the output stream should be opened or not.
- virtual void setEmitPjl (bool emitPjl)=0

  Set whether or not a PJL header or end of job should be emitted.
- virtual void setEnableTrueTypeNotDef (bool enableTrueTypeNotDef)=0

  Set whether or not a TrueType font's .notdef glyph is displayed.

Public Member Functions inherited from JawsMako::IOutput

- virtual void **setPreset** (const U8String &preset)=0

  Configure the output according to a general preset. Please see the supplied documentation for details of these presets. The default is "Preserve" which will attempt to produce output as close to the input as possible for the output format. A string value can be used for any parameter and will be converted as necessary.
- virtual void **setParameter** (const U8String &param, const U8String &value)=0

  Apply a key value pair output parameter with a string value. The parameter name is case insensitive. Please refer to the supplied documentation for the details of the available parameters and their ranges.
- virtual void setAllowedPermissionsFlags (uint32 allowedPermissions)=0

  Control whether or not assemblies with certain security permission flags are allowed to be written by this output.
- virtual void writeAssembly (const IDocumentAssemblyPtr &assembly, const U8String &pathToFile)=0

  Write the given document assembly to a file on disk.
- virtual void writeAssembly (const IDocumentAssemblyPtr &assembly, const String &pathToFile)=0

  Write the given document assembly to a file on disk, specified by a wide character string.
- virtual void writeAssembly (const IDocumentAssemblyPtr &assembly, const IOutputStreamPtr &stream)=0

  Write the given document assembly to a stream.
- virtual IOutputWriterPtr openWriter (const IDocumentAssemblyPtr &assembly, const U8String &pathToFile)=0

  Create an output writer for the given assembly, targeting a file on disk. This is designed to allow streaming output, or to deal with situations where an operation would require too much memory to hold an entire edited assembly in memory at once.
- virtual IOutputWriterPtr openWriter (const IDocumentAssemblyPtr &assembly, const String &pathToFile)=0

  Create an output writer for the given assembly, targeting a file on disk. As above, but with the file specified in a wide character string.
- virtual IOutputWriterPtr openWriter (const IDocumentAssemblyPtr &assembly, const IOutputStreamPtr &stream)=0

  Create an output writer for the given assembly, targeting a stream.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IPCL5OutputPtr create (const IJawsMakoPtr &jawsMako, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create a PCL5 Output instance.

Static Public Member Functions inherited from JawsMako::IOutput

- static JAWSMAKO_API IOutputPtr create (const IJawsMakoPtr &jawsMako, eFileFormat format, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create an output for writing source in the given format.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.366.1 Detailed Description

Interface for the PCL5 IOutput class.

7.366.2 Member Function Documentation

create()

```
static JAWSMAKO_API IPCL5OutputPtr JawsMako::IPCL5Output::create (
            const IJawsMakoPtr & jawsMako,
            const IProgressMonitorPtr & progressMonitor = IProgressMonitorPtr() )  [static]
```

Create a PCL5 Output instance.

Parameters

jawsMako	The JawsMako instance.
progressMonitor	The progress monitor which allows aborting an operation or registering a progress callback.

setEmitPjl()

```
virtual void JawsMako::IPCL5Output::setEmitPjl (
            bool emitPjl )  [pure virtual]
```

Set whether or not a PJL header or end of job should be emitted.

```
The default is true.

Equivalent to calling setParameter() with the parameter name
"EmitPJL", the value "true" or "false".
```

setEnableTrueTypeNotDef()

```
virtual void JawsMako::IPCL5Output::setEnableTrueTypeNotDef (
            bool enableTrueTypeNotDef )  [pure virtual]
```

Set whether or not a TrueType font's .notdef glyph is displayed.

```
The default is false.

Equivalent to calling setParameter() with the parameter name
"EnableTrueTypeNotDef", the value "true" or "false".
```

setImageCompression()

```
virtual void JawsMako::IPCL5Output::setImageCompression (
            ePCL5ImageCompression compression )  [pure virtual]
```

Set the desired image compression method. The default is delta row. Equivalent to calling setParameter() with the parameter name "ImageCompression" using the string value "None", "RLE", or "DeltaRow".

Parameters

compression	The compression method to use.

setIncludeMarginsWhenSelectingPaper()

```
virtual void JawsMako::IPCL5Output::setIncludeMarginsWhenSelectingPaper (
            bool include )  [pure virtual]
```

Set whether paper selection for the PCL output should take into account the non-printable margins of the paper when selecting a paper size.

Real PCL/5 devices generally cannot print to the edge of a sheet of paper. Setting this parameter it to true will cause the paper size selection algorithm to potentially choose a larger paper size such that the page being output will appear in its entirety. If false, the page size will be chosen based on the physical size of the DOM page. In both cases the output will be centered on the selected paper. Equivalent to calling setParameter() with the parameter name "IncludeMarginsWhenSelectingPaper" using the string value "true" or "false".

The default is false.

Parameters

include	True to include the margins when selecting a paper size.

setMediaSource()

```
virtual void JawsMako::IPCL5Output::setMediaSource (
            uint32 mediaSource )  [pure virtual]
```

Set the desired media source The default is 7. Equivalent to calling setParameter() with the parameter name name "MediaSource".

Parameters

mediaSource	The media source to use.

setOpenStream()

```
virtual void JawsMako::IPCL5Output::setOpenStream (
            bool open )  [pure virtual]
```

Set whether the output stream should be opened or not.

```
If true, the output stream will be opened and closed by the
PCL5 output. If false, the stream will assumed to be opened
and will not be closed.

Setting this to false allows the PCL5 stream to be written
to an existing stream or channel.

The default is true.

Equivalent to calling setParameter() with the parameter name
"OpenStream", the value "true" or "false".
```

setResolution()

```
virtual void JawsMako::IPCL5Output::setResolution (
            uint32 resolution )  [pure virtual]
```

Set the PCL resolution The default is 600. Equivalent to calling setParameter() with the parameter name "↩ Resolution".

Parameters

resolution	The resolution to use.

setVersion()

```
virtual void JawsMako::IPCL5Output::setVersion (
            ePCL5Version version )  [pure virtual]
```

Set the PCL version. The default is ePCL5c. Equivalent to calling setParameter() with the parameter name "PCL5↩ Version" using the string value "pcl5e" or "pcl5c".

Parameters

version	The version to use.

The documentation for this class was generated from the following file:

- pcl5output.h

7.367 JawsMako::IPCLXLAttributeHandler Class Reference

Interface allowing users to monitor/handle illegal PCL/XL operator attributes.

```
#include <pclxlinput.h>
```

Public Member Functions

- virtual bool ignoreAttribute (const RawString &operatorName, uint16 attributeId)=0

  Invoked by the input when the an illegal PCL/XL attribute is encountered.

7.367.1 Detailed Description

Interface allowing users to monitor/handle illegal PCL/XL operator attributes.

7.367.2 Member Function Documentation

ignoreAttribute()

```
virtual bool JawsMako::IPCLXLAttributeHandler::ignoreAttribute (
            const RawString & operatorName,
            uint16 attributeId )  [pure virtual]
```

Invoked by the input when the an illegal PCL/XL attribute is encountered.

Parameters

operatorName	The operator name as listed in the PCL/XL reference.
attributeId	The attribute ID as listed in the PCL/XL reference.

Returns

bool Return true to indicate that Mako should ignore this attribute.

The documentation for this class was generated from the following file:

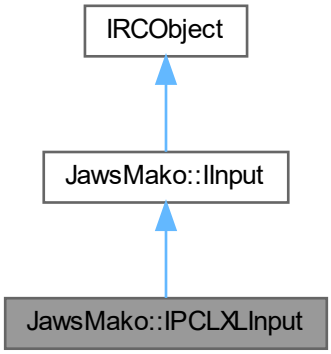
- pclxlinput.h

7.368 JawsMako::IPCLXLInput Class Reference

An instance of the JawsMako PCL/XL input class.

```
#include <pclxlinput.h>
```

Inheritance diagram for JawsMako::IPCLXLInput:



Public Member Functions

- virtual void setRopResolution (uint32 resolution)=0

  Set the resolution to be used when flattening ROPs.
- virtual void setFlattenRops (bool flatten)=0

  Set whether or not ROPs should be flattened.
- virtual void setDefaultPaperSize (const U8String &paperSize)=0

  Set the default paper size.
- virtual void setDefaultLandscape (bool landscape)=0

  Set the default orientation.
- virtual void setDefaultCopies (uint32 copies)=0

  Set the default number of per-page copies.
- virtual void setDefaultDuplex (bool duplex)=0

  Set whether or not duplex should be set by default.
- virtual void setDefaultDuplexBindingMode (IPJLParser::eDuplexBindingMode bindingMode)=0

  Set the default binding edge for duplexing.
- virtual void setDefaultManualFeed (bool manualFeed)=0

  Set whether or not manual feed should be set by default.
- virtual void setDefaultsFromPjl (IPJLParserPtr &pjlParser)=0

  Take initialisation data from the given PJL parser.
- virtual void enableUnencapsulatedMode (bool unencapsulated)=0

  Enable or disable unencapsulated mode.
- virtual void setMediaHandler (IMediaHandler *mediaHandler)=0

  Set the IMediaHandler instance to use to handle unsatisfied media requests.
- virtual void setAttributeHandler (IPCLXLAttributeHandler *attributeHandler)=0

  Set the IPCLXLAttributeHandler instance to use to handle illegal PCL/XL operator attributes.

Public Member Functions inherited from JawsMako::IInput

- virtual IDocumentAssemblyPtr open (const U8String &pathToFile)=0

  Open a file on disk, returning the IDocumentAssembly representing the contents.
- virtual IDocumentAssemblyPtr open (const String &pathToFile)=0

  Open a file on disk, returning the IDocumentAssembly representing the contents. Takes a wide character string.
- virtual IDocumentAssemblyPtr open (const IInputStreamPtr &inputStream)=0

  Open a stream, returning the IDocumentAssembly representing the contents.
- virtual void setSequentialMode (bool sequential)=0

  Set/unset sequential mode on this input.
- virtual void setParameter (const U8String &param, const U8String &value)=0

  Apply a key value pair output parameter with a string value. The parameter name is case insensitive. Please refer to the supplied documentation for the details of the available parameters and their ranges.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IPCLXLInputPtr create (const IJawsMakoPtr &jawsMako, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create an input for reading source documents in PCL/XL format.

Static Public Member Functions inherited from JawsMako::IInput

- static JAWSMAKO_API IInputPtr create (const IJawsMakoPtr &jawsMako, eFileFormat format, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create an input for reading source documents in the given format. The following formats are currently supported:

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.368.1 Detailed Description

An instance of the JawsMako PCL/XL input class.

7.368.2 Member Function Documentation

create()

```
static JAWSMAKO_API IPCLXLInputPtr JawsMako::IPCLXLInput::create (
            const IJawsMakoPtr & jawsMako,
            const IProgressMonitorPtr & progressMonitor = IProgressMonitorPtr() )  [static]
```

Create an input for reading source documents in PCL/XL format.

Parameters

jawsMako	The IJawsMako object
progressMonitor	A smart pointer to an IProgressMonitor object which can be NULL if no such object was passed in.

Returns

IPCLXLInputPtr the PCL/XL input

enableUnencapsulatedMode()

```
virtual void JawsMako::IPCLXLInput::enableUnencapsulatedMode (
            bool unencapsulated )  [pure virtual]
```

Enable or disable unencapsulated mode.

```
Setting this mode to true changes the parser behaviour
as follows.

- The input stream is not opened; it is assumed to point to
  the start of a PCL/XL session.
- The PJL parser is not used. If PJL is encountered an
  error will be thrown.
- The PCL/XL interpreter will not consume data after the point the
  PCL/XL interpreter either exhausts the stream or returns
  to PJL. After requesting a document after the last document
  in the session, or by requesting the number of documents) the
  stream will be positioned at the point that the PCL/XL
  interpreter exited.

  Note: The file pointer may move if the input stream is random
  access and earlier pages are re-requested. So, if using a
  random-access stream, care is required.

Whereas the default (false) will:

- open() the input stream.
- use the built-in PJL parser when PJL is encountered.
- process all PCL/XL sessions present in the stream, even
  if they are interupted by returning to PJL.
```

setAttributeHandler()

```
virtual void JawsMako::IPCLXLInput::setAttributeHandler (
            IPCLXLAttributeHandler * attributeHandler )  [pure virtual]
```

Set the IPCLXLAttributeHandler instance to use to handle illegal PCL/XL operator attributes.

Note that this routine does not take ownership of the passed pointer, which must remain allocated until after the input is complete.

setDefaultCopies()

```
virtual void JawsMako::IPCLXLInput::setDefaultCopies (
            uint32 copies )  [pure virtual]
```

Set the default number of per-page copies.

```
      The default is 1. Must be greater than zero.
```

setDefaultDuplex()

```
virtual void JawsMako::IPCLXLInput::setDefaultDuplex (
            bool duplex )  [pure virtual]
```

Set whether or not duplex should be set by default.

```
      The default is false.
```

setDefaultDuplexBindingMode()

```
virtual void JawsMako::IPCLXLInput::setDefaultDuplexBindingMode (
            IPJLParser::eDuplexBindingMode bindingMode )  [pure virtual]
```

Set the default binding edge for duplexing.

```
      The default is long edge.
```

setDefaultLandscape()

```
virtual void JawsMako::IPCLXLInput::setDefaultLandscape (
            bool landscape )  [pure virtual]
```

Set the default orientation.

```
      Pass true to set landscape, or false to set portrait.
      The default is portrait.
```

setDefaultManualFeed()

```
virtual void JawsMako::IPCLXLInput::setDefaultManualFeed (
            bool manualFeed )  [pure virtual]
```

Set whether or not manual feed should be set by default.

```
      The default is false.
```

setDefaultPaperSize()

```
virtual void JawsMako::IPCLXLInput::setDefaultPaperSize (
            const U8String & paperSize )  [pure virtual]
```

Set the default paper size.

The paper size is set using a string. To take effect the paper size string must match one of the standard strings in the PCL/XL specification:

- letter
- legal
- a4
- exec
- executive
- ledger
- a3
- com10envelope
- monarchenvelope
- c5envelope
- dlenvelope
- b4
- jb5
- b5envelope
- jpostcard
- jdoublepostcard
- jdoublepostcard
- a5
- a6
- jb6

The matching is case-insensitive.

The default page size is a4

setDefaultsFromPjl()

```
virtual void JawsMako::IPCLXLInput::setDefaultsFromPjl (
            IPJLParserPtr & pjlParser )  [pure virtual]
```

Take initialisation data from the given PJL parser.

Uses the environment in the given PJL parser to set defaults in the PCL/XL input. Any previously set defaults will be ignored.

This is especially useful with unencapsulated mode, where PJL parsing happens externally to the input.

setFlattenRops()

```
virtual void JawsMako::IPCLXLInput::setFlattenRops (
            bool flatten )  [pure virtual]
```

Set whether or not ROPs should be flattened.

Please see setRopResolution() for information on ROPs.

If false, the input will not flatten ROPs. As a result the DOM that is produced will not be visually correct. Instead, the ROP components will appear in the DOM as distinct objects grouped in a special IDOMGroup.

This mode is useful if only information about the job is required. ROPs may be expensive especially for pathological cases which use large amounts of overdraw. If we merely need to know what text is on a page, the number of pages, whether the content is black and white or color etc, then flattening ROPs may significantly slow processing for no benefit. Setting this parameter false will avoid that processing.

The default is true; ROPs will be flattened.

If false, the ROPs will appear in the DOM inside groups with properties denoting their purpose. The operands for a ROP will be stored inside an IDOMGroup that is tagged with the property (see IDOMNode::getProperty()) "RopGroup". Inside that group will be up to two nodes that represent the ROP operands. These will be tagged with the property "RopSource" for the source operand, and "RopPaint" for the paint operand. The value of these properties is irrelevant; only their presence is important.

Parameters

flatten	True to flatten ROPs, false to avoid flattening ROPs.

setMediaHandler()

```
virtual void JawsMako::IPCLXLInput::setMediaHandler (
            IMediaHandler * mediaHandler )  [pure virtual]
```

Set the IMediaHandler instance to use to handle unsatisfied media requests.

Note that this routine does not take ownership of the passed pointer, which must remain allocated until after the input is complete.

setRopResolution()

```
virtual void JawsMako::IPCLXLInput::setRopResolution (
            uint32 resolution )  [pure virtual]
```

Set the resolution to be used when flattening ROPs.

PCL/XL features a method of using bitwise operations to composite pixels from patterns and images with content already on the page. These are known as ROPs (Raster OPerations). These are not compatible with the DOM and most other PDLs, and in many situations require rendering at the input stage.

This parameter controls the rendering resolution. The default is 600dpi.

Parameters

resolution	The ROP rendering resolution to use.

The documentation for this class was generated from the following file:

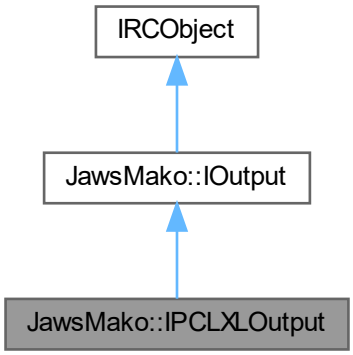
- pclxlinput.h

7.369 JawsMako::IPCLXLOutput Class Reference

Interface for the PCLXL IOutput class.

`#include <pclxloutput.h>`

Inheritance diagram for JawsMako::IPCLXLOutput:



Public Member Functions

- virtual void **setResolution** (float resolution)=0

  Set the PCLXL resolution The default is 600. Equivalent to calling setParameter() with the parameter name "Resolution".
- virtual void setOpenStream (bool open)=0

  Set whether the output stream should be opened or not.
- virtual void setEmitPjl (bool emitPjl)=0

  Set whether or not a PJL header or end of job should be emitted.

Public Member Functions inherited from JawsMako::IOutput

- virtual void **setPreset** (const U8String &preset)=0

  Configure the output according to a general preset. Please see the supplied documentation for details of these presets. The default is "Preserve" which will attempt to produce output as close to the input as possible for the output format. A string value can be used for any parameter and will be converted as necessary.
- virtual void **setParameter** (const U8String &param, const U8String &value)=0

  Apply a key value pair output parameter with a string value. The parameter name is case insensitive. Please refer to the supplied documentation for the details of the available parameters and their ranges.
- virtual void setAllowedPermissionsFlags (uint32 allowedPermissions)=0

  Control whether or not assemblies with certain security permission flags are allowed to be written by this output.

- virtual void writeAssembly (const IDocumentAssemblyPtr &assembly, const U8String &pathToFile)=0

  Write the given document assembly to a file on disk.
- virtual void writeAssembly (const IDocumentAssemblyPtr &assembly, const String &pathToFile)=0

  Write the given document assembly to a file on disk, specified by a wide character string.
- virtual void writeAssembly (const IDocumentAssemblyPtr &assembly, const IOutputStreamPtr &stream)=0

  Write the given document assembly to a stream.
- virtual IOutputWriterPtr openWriter (const IDocumentAssemblyPtr &assembly, const U8String &pathToFile)=0

  Create an output writer for the given assembly, targeting a file on disk. This is designed to allow streaming output, or to deal with situations where an operation would require too much memory to hold an entire edited assembly in memory at once.
- virtual IOutputWriterPtr openWriter (const IDocumentAssemblyPtr &assembly, const String &pathToFile)=0

  Create an output writer for the given assembly, targeting a file on disk. As above, but with the file specified in a wide character string.
- virtual IOutputWriterPtr openWriter (const IDocumentAssemblyPtr &assembly, const IOutputStreamPtr &stream)=0

  Create an output writer for the given assembly, targeting a stream.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IPCLXLOutputPtr create (const IJawsMakoPtr &jawsMako, const IProgressMonitor↩ Ptr &progressMonitor=IProgressMonitorPtr())

  Create a PCL5 Output instance.

Static Public Member Functions inherited from JawsMako::IOutput

- static JAWSMAKO_API IOutputPtr create (const IJawsMakoPtr &jawsMako, eFileFormat format, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create an output for writing source in the given format.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.369.1 Detailed Description

Interface for the PCLXL IOutput class.

7.369.2 Member Function Documentation

create()

```
static JAWSMAKO_API IPCLXLOutputPtr JawsMako::IPCLXLOutput::create (
            const IJawsMakoPtr & jawsMako,
            const IProgressMonitorPtr & progressMonitor = IProgressMonitorPtr() )  [static]
```

Create a PCL5 Output instance.

Parameters

jawsMako	The JawsMako instance.
progressMonitor	The progress monitor which allows aborting an operation or registering a progress callback.

setEmitPjl()

```
virtual void JawsMako::IPCLXLOutput::setEmitPjl (
            bool emitPjl )  [pure virtual]
```

Set whether or not a PJL header or end of job should be emitted.

```
The default is true.

Equivalent to calling setParameter() with the parameter name
"EmitPJL", the value "true" or "false".
```

setOpenStream()

```
virtual void JawsMako::IPCLXLOutput::setOpenStream (
            bool open )  [pure virtual]
```

Set whether the output stream should be opened or not.

```
If true, the output stream will be opened and closed by the
PCLXL output. If false, the stream will assumed to be opened
and will not be closed.

Setting this to false allows the PCLXL stream to be written
to an existing stream or channel.

The default is true.

Equivalent to calling setParameter() with the parameter name
"OpenStream", the value "true" or "false".
```

The documentation for this class was generated from the following file:

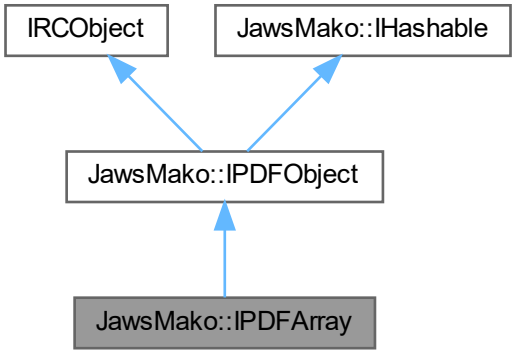
- pclxloutput.h

7.370 JawsMako::IPDFArray Class Reference

A simple class representing a mutable array of other PDF objects.

`#include <pdfobjects.h>`

Inheritance diagram for JawsMako::IPDFArray:



Public Member Functions

- virtual uint32 getSize () const =0

  Get the current size of the array.
- virtual IPDFObjectPtr get (uint32 index) const =0

  Get the object stored at the given index. An exception will be thrown if the index is out of range.
- virtual ePDFObjectType getTypeAtIndex (uint32 index) const =0

  Get the type of the object stored at the given index. An exception will be thrown if the index is out of range.
- virtual void put (uint32 index, const IPDFObjectPtr &value)=0

  Store the given object in the array at the given index, replacing the currently stored object. If the index is outside the array, it will be resized to suit. An exception will be thrown if the index is out of range.
- virtual void insert (uint32 index, const IPDFObjectPtr &value)=0

  Insert the given object in the array at the given index, moving the following objects forward to make room. An exception will be thrown if the index is out of range.
- virtual void remove (uint32 index)=0

  Remove the object at the given index from the array, decreasing the size of the array by 1. An exception will be thrown if the index is out of range.
- virtual void append (const IPDFObjectPtr &value)=0

  Convenience member to append an object to the end of the array.
- virtual void copy (uint32 destIndex, const IPDFArrayPtr &sourceArray, uint32 sourceIndex)=0

  Copy (but not clone) an entry from another array into an existing posision in this array, replacing any currently stored object. Usually more efficient for bulk operations.
- virtual bool getNumbers (CDoubleVect &numbers)=0

  If the array consists entirely of numbers, retrieve those numbers.
- virtual bool getBox (FBox &box)=0

  If the array consists of four numbers, retrieve those numbers as a box (left, bottom, right, top)
- virtual int32 getInteger (uint32 index) const =0

  Convenience member to obtain an integer from the array at the given index. An exception will be thrown if the index is out of range or the object is not an integer.
- virtual bool containsCompositeObject (const IPDFObjectPtr &object) const =0

  Recursively check to see if the array contains the given composite object (That is, a dictionary, stream, or array).

Public Member Functions inherited from JawsMako::IPDFObject

- virtual ePDFObjectType getType () const =0

  Get the type of this PDF object.
- virtual IPDFObjectPtr clone () const =0

  Create a clone of this PDF object if appropriate. Note that simple immutable objects are not actually cloned and instead the same object is returned.
- virtual IPDFObjectPtr deepClone () const =0

  Create a deep clone of this PDF object if appropriate. Note that simple immutable objects are not actually cloned and instead the same object is returned. If the object is composite (an IPDFArray, IPDFDictionary or IPDFStream) then the object will be cloned as well as the consituent objects in a recursive fashion.
- virtual bool getIsExecutable () const =0

  Get whether or not the object is considered executable, such as an executable name, array, or operator.
- virtual bool containsReferences () const =0

  Get whether or not the object contains indirect references to other objects (that is, does the object or any constituent object contain IPDFReference or IPDFFarReference objects).
- virtual void **emitPostScriptCode** (const IOutputStreamPtr &dest) const =0

  Convert the object to PostScript code and write to the given stream. Only allowed for objects that are representable in PostScript.
- virtual bool getNumber (double &number) const =0

  If the object is a number (a real or integer) obtain that number.
- virtual bool getNumber (float &number) const =0

  If the object is a number (a real or integer) within the range of a 32-bit float, obtain that number. If the object is an integer, thiw will fail if the integer cannot be represented exactly in a single precision float.
- virtual bool getNumber (int32 &number) const =0

  If the object is an integer (a integral real or integer) within the range of a 32-bit integer, obtain that number.
- virtual bool getString (RawString &string) const =0

  If the object is a string or a name, retrieve the raw string data.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from JawsMako::IHashable

- virtual uint64 hash () const

  Obtain a 64-bit hash of the receiving object.
- virtual void updateHash (uint64 &hash) const =0

  Update the given hash to include the receiver.

Static Public Member Functions

- static JAWSMAKO_API IPDFArrayPtr create (uint32 size=0, bool executable=false)

  Create a new PDF array of the given initial size.
- static JAWSMAKO_API IPDFArrayPtr create (const CPDFObjectVect &objects, bool executable=false)

  Create a new PDF array from a vector of IPDFObjects.
- static JAWSMAKO_API IPDFArrayPtr createFromNumbers (const CDoubleVect &numbers)

  Create from a vector of numbers.
- static JAWSMAKO_API IPDFArrayPtr createFromNumbersArray (const double *numbers, uint32 numNumbers)

  Create from an array of numbers.
- static JAWSMAKO_API IPDFArrayPtr createFromFBox (const FBox &box)

  Create from an FBox.

Static Public Member Functions inherited from JawsMako::IPDFObject

- static JAWSMAKO_API IPDFObjectPtr createNumber (double number)

  If the given number is an integer, create an IPDFInteger, otherwise create an IPDFReal.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.370.1 Detailed Description

A simple class representing a mutable array of other PDF objects.

7.370.2 Member Function Documentation

append()

```
virtual void JawsMako::IPDFArray::append (
            const IPDFObjectPtr & value )  [pure virtual]
```

Convenience member to append an object to the end of the array.

Parameters

value	The object to append.

containsCompositeObject()

```
virtual bool JawsMako::IPDFArray::containsCompositeObject (
```

```
            const IPDFObjectPtr & object ) const  [pure virtual]
```

Recursively check to see if the array contains the given composite object (That is, a dictionary, stream, or array).

Parameters

object	The object to search for.

Returns

bool True if the given object was found.

copy()

```
virtual void JawsMako::IPDFArray::copy (
            uint32 destIndex,
            const IPDFArrayPtr & sourceArray,
            uint32 sourceIndex )  [pure virtual]
```

Copy (but not clone) an entry from another array into an existing posision in this array, replacing any currently stored object. Usually more efficient for bulk operations.

Parameters

destIndex	The destination index to store the object.
sourceArray	The array object from which to copy.
sourceIndex	The index of the entry in sourceArray to copy.

create() [1/2]

```
static JAWSMAKO_API IPDFArrayPtr JawsMako::IPDFArray::create (
            const CPDFObjectVect & objects,
            bool executable = false )  [static]
```

Create a new PDF array from a vector of IPDFObjects.

Parameters

objects	The objects to assign to the array.
executable	If this array should be marked executable (that is, a PDF/PS procedure).

Returns

IPDFArrayPtr The created object.

create() [2/2]

```
static JAWSMAKO_API IPDFArrayPtr JawsMako::IPDFArray::create (
```

```
            uint32 size = 0,
            bool executable = false )  [static]
```

Create a new PDF array of the given initial size.

Parameters

size	The initial size of the array. It will be populated with IPDFNull objects.
executable	If this array should be marked executable (that is, a PDF/PS procedure).

Returns

IPDFArrayPtr The created object.

createFromFBox()

```
static JAWSMAKO_API IPDFArrayPtr JawsMako::IPDFArray::createFromFBox (
            const FBox & box )  [static]
```

Create from an FBox.

Parameters

box	The box to use.

Returns

IPDFArray The created object.

createFromNumbers()

```
static JAWSMAKO_API IPDFArrayPtr JawsMako::IPDFArray::createFromNumbers (
            const CDoubleVect & numbers )  [static]
```

Create from a vector of numbers.

Parameters

numbers	The numbers to assign to the array.

Returns

IPDFArray The created object.

createFromNumbersArray()

```
static JAWSMAKO_API IPDFArrayPtr JawsMako::IPDFArray::createFromNumbersArray (
            const double * numbers,
            uint32 numNumbers )  [static]
```

Create from an array of numbers.

Parameters

numbers	The numbers to assign to the array.
numNumbers	The number of numbers in the array. Integers will be created where possible.

Returns

IPDFArray The created object.

get()

```
virtual IPDFObjectPtr JawsMako::IPDFArray::get (
            uint32 index ) const  [pure virtual]
```

Get the object stored at the given index. An exception will be thrown if the index is out of range.

Parameters

index	The index of the desired object.

Returns

IPDFObjectPtr The retrieved object.

getBox()

```
virtual bool JawsMako::IPDFArray::getBox (
            FBox & box )  [pure virtual]
```

If the array consists of four numbers, retrieve those numbers as a box (left, bottom, right, top)

Parameters

box	The FBox to receive the numbers.

Returns

bool True if the array was four numbers, fails otherwise.

getInteger()

```
virtual int32 JawsMako::IPDFArray::getInteger (
            uint32 index ) const  [pure virtual]
```

Convenience member to obtain an integer from the array at the given index. An exception will be thrown if the index is out of range or the object is not an integer.

Parameters

index	The index of the object to query.

Returns

int32 The integer value of the object.

getNumbers()

```
virtual bool JawsMako::IPDFArray::getNumbers (
            CDoubleVect & numbers )  [pure virtual]
```

If the array consists entirely of numbers, retrieve those numbers.

Parameters

numbers	A reference to a vector to receive the numbers.

Returns

bool True if the array was entirely numbers, fails otherwise (in which case numbers will be empty).

getSize()

```
virtual uint32 JawsMako::IPDFArray::getSize ( ) const  [pure virtual]
```

Get the current size of the array.

Returns

uint32 The size.

getTypeAtIndex()

```
virtual ePDFObjectType JawsMako::IPDFArray::getTypeAtIndex (
            uint32 index ) const  [pure virtual]
```

Get the type of the object stored at the given index. An exception will be thrown if the index is out of range.

Parameters

index	The index of the desired object.

Returns

ePDFObjectType The type of the stored object.

insert()

```
virtual void JawsMako::IPDFArray::insert (
            uint32 index,
            const IPDFObjectPtr & value )  [pure virtual]
```

Insert the given object in the array at the given index, moving the following objects forward to make room. An exception will be thrown if the index is out of range.

Parameters

index	The index at which to store the object.
value	The object to insert.

put()

```
virtual void JawsMako::IPDFArray::put (
            uint32 index,
            const IPDFObjectPtr & value )  [pure virtual]
```

Store the given object in the array at the given index, replacing the currently stored object. If the index is outside the array, it will be resized to suit. An exception will be thrown if the index is out of range.

Parameters

index	The index at which to store the object.
value	The object to store.

remove()

```
virtual void JawsMako::IPDFArray::remove (
            uint32 index )  [pure virtual]
```

Remove the object at the given index from the array, decreasing the size of the array by 1. An exception will be thrown if the index is out of range.

Parameters

index	The index of the object to remove.

The documentation for this class was generated from the following file:

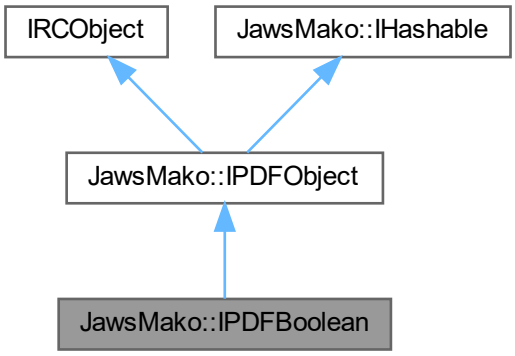
- pdfobjects.h

7.371 JawsMako::IPDFBoolean Class Reference

A simple immutable boolean PDF object type.

`#include <pdfobjects.h>`

Inheritance diagram for JawsMako::IPDFBoolean:



Public Member Functions

- virtual bool getValue () const =0

  Get the value of the boolean.

Public Member Functions inherited from JawsMako::IPDFObject

- virtual ePDFObjectType getType () const =0

  Get the type of this PDF object.
- virtual IPDFObjectPtr clone () const =0

  Create a clone of this PDF object if appropriate. Note that simple immutable objects are not actually cloned and instead the same object is returned.
- virtual IPDFObjectPtr deepClone () const =0

  Create a deep clone of this PDF object if appropriate. Note that simple immutable objects are not actually cloned and instead the same object is returned. If the object is composite (an IPDFArray, IPDFDictionary or IPDFStream) then the object will be cloned as well as the consituent objects in a recursive fashion.
- virtual bool getIsExecutable () const =0

  Get whether or not the object is considered executable, such as an executable name, array, or operator.
- virtual bool containsReferences () const =0

  Get whether or not the object contains indirect references to other objects (that is, does the object or any constituent object contain IPDFReference or IPDFFarReference objects).
- virtual void **emitPostScriptCode** (const IOutputStreamPtr &dest) const =0

  Convert the object to PostScript code and write to the given stream. Only allowed for objects that are representable in PostScript.
- virtual bool getNumber (double &number) const =0

  If the object is a number (a real or integer) obtain that number.

- virtual bool getNumber (float &number) const =0

  If the object is a number (a real or integer) within the range of a 32-bit float, obtain that number. If the object is an integer, thiw will fail if the integer cannot be represented exactly in a single precision float.
- virtual bool getNumber (int32 &number) const =0

  If the object is an integer (a integral real or integer) within the range of a 32-bit integer, obtain that number.
- virtual bool getString (RawString &string) const =0

  If the object is a string or a name, retrieve the raw string data.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from JawsMako::IHashable

- virtual uint64 hash () const

  Obtain a 64-bit hash of the receiving object.
- virtual void updateHash (uint64 &hash) const =0

  Update the given hash to include the receiver.

Static Public Member Functions

- static JAWSMAKO_API IPDFBooleanPtr create (bool boolean)

  Create a boolean object.

Static Public Member Functions inherited from JawsMako::IPDFObject

- static JAWSMAKO_API IPDFObjectPtr createNumber (double number)

  If the given number is an integer, create an IPDFInteger, otherwise create an IPDFReal.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.371.1 Detailed Description

A simple immutable boolean PDF object type.

7.371.2 Member Function Documentation

create()

```
static JAWSMAKO_API IPDFBooleanPtr JawsMako::IPDFBoolean::create (
            bool boolean )  [static]
```

Create a boolean object.

Parameters

boolean	The boolean value.

Returns

IPDFBooleanPtr The created object.

getValue()

```
virtual bool JawsMako::IPDFBoolean::getValue ( ) const  [pure virtual]
```

Get the value of the boolean.

Returns

bool The value of the boolean.

The documentation for this class was generated from the following file:

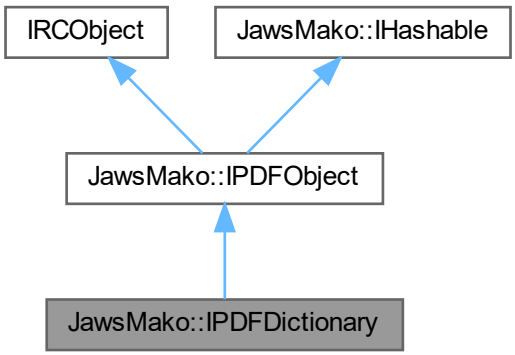
- pdfobjects.h

7.372 JawsMako::IPDFDictionary Class Reference

A simple class representing a mutable dictionary of key-value pairs where the keys are PDF names and the values are PDF objects.

```
#include <pdfobjects.h>
```

Inheritance diagram for JawsMako::IPDFDictionary:



Public Member Functions

- virtual uint32 getSize () const =0

  Get the current size of the dictionary, including currently empty entries.
- virtual IPDFNamePtr getKeyAtIndex (uint32 index) const =0

  Get the key of the dictionary entry at the given index.
- virtual IPDFObjectPtr getValueAtIndex (uint32 index) const =0

  Get the value stored in the dictionary entry at the given index.
- virtual ePDFObjectType getValueTypeAtIndex (uint32 index) const =0

  Get the type of the value stored in the dictionary entry at the given index. An exception is thrown if the dictionary entry is empty.
- virtual IPDFObjectPtr get (const char *key) const =0

  Get the object stored in the dictionary with the given null-terminated string as key.
- virtual IPDFObjectPtr get (const RawString &key) const =0

  Get the object stored in the dictionary with the given raw string as key.
- virtual IPDFObjectPtr get (const IPDFNamePtr &key) const =0

  Get the object stored in the dictionary with the given DF name as key.
- virtual void put (const char *key, const IPDFObjectPtr &value)=0

  Store an object in the dictionary under the given key.
- virtual void put (const RawString &key, const IPDFObjectPtr &value)=0

  Store an object in the dictionary under the given key.
- virtual void put (const IPDFNamePtr &key, const IPDFObjectPtr &value)=0

  Store an object in the dictionary under the given key.
- virtual void undefine (const char *key)=0

  Remove the object stored under the given key, if it exists.
- virtual void undefine (const RawString &key)=0

  Remove the object stored under the given key, if it exists.
- virtual void undefine (const IPDFNamePtr &key)=0

  Remove the object stored under the given key, if it exists.
- virtual void copy (const IPDFDictionaryPtr &sourceDict, uint32 sourceIndex)=0

  Copy (but not clone) an entry from another dictionary into this dictionary, replacing any currently stored object. Usually more efficient for bulk operations.
- virtual void putInteger (const IPDFNamePtr &key, int32 integer)=0

  Convenience member to add an Integer to a dictionary.
- virtual void putReal (const IPDFNamePtr &key, double real)=0

  Convenience member to add a real to a dictionary.
- virtual bool containsCompositeObject (const IPDFObjectPtr &object) const =0

  Recursively check to see if the dictionary contains the given composite object (That is, a dictionary, stream, or array).
- virtual Iterator begin ()=0

  Create an iterator to iterate through all the key-value pairs in the dictionary, beginning at the first entry.
- virtual Iterator end ()=0

  Obtain an interator representing the end of the dictionary.

Public Member Functions inherited from JawsMako::IPDFObject

- virtual ePDFObjectType getType () const =0

  Get the type of this PDF object.
- virtual IPDFObjectPtr clone () const =0

  Create a clone of this PDF object if appropriate. Note that simple immutable objects are not actually cloned and instead the same object is returned.
- virtual IPDFObjectPtr deepClone () const =0

Create a deep clone of this PDF object if appropriate. Note that simple immutable objects are not actually cloned and instead the same object is returned. If the object is composite (an IPDFArray, IPDFDictionary or IPDFStream) then the object will be cloned as well as the consituent objects in a recursive fashion.

- virtual bool getIsExecutable () const =0

  Get whether or not the object is considered executable, such as an executable name, array, or operator.
- virtual bool containsReferences () const =0

  Get whether or not the object contains indirect references to other objects (that is, does the object or any constituent object contain IPDFReference or IPDFFarReference objects).
- virtual void **emitPostScriptCode** (const IOutputStreamPtr &dest) const =0

  Convert the object to PostScript code and write to the given stream. Only allowed for objects that are representable in PostScript.
- virtual bool getNumber (double &number) const =0

  If the object is a number (a real or integer) obtain that number.
- virtual bool getNumber (float &number) const =0

  If the object is a number (a real or integer) within the range of a 32-bit float, obtain that number. If the object is an integer, thiw will fail if the integer cannot be represented exactly in a single precision float.
- virtual bool getNumber (int32 &number) const =0

  If the object is an integer (a integral real or integer) within the range of a 32-bit integer, obtain that number.
- virtual bool getString (RawString &string) const =0

  If the object is a string or a name, retrieve the raw string data.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from JawsMako::IHashable

- virtual uint64 hash () const

  Obtain a 64-bit hash of the receiving object.
- virtual void updateHash (uint64 &hash) const =0

  Update the given hash to include the receiver.

Static Public Member Functions

- static JAWSMAKO_API IPDFDictionaryPtr create (IEDLClassFactory *pFactory, uint32 size=0)

  Create a new PDF dictionary of the given initial capacity.

Static Public Member Functions inherited from JawsMako::IPDFObject

- static JAWSMAKO_API IPDFObjectPtr createNumber (double number)

  If the given number is an integer, create an IPDFInteger, otherwise create an IPDFReal.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()
  Virtual destructor.

7.372.1 Detailed Description

A simple class representing a mutable dictionary of key-value pairs where the keys are PDF names and the values are PDF objects.

A simple class representing a mutable PDF stream. A stream consists of a dictionary paired with an IRAInputStream containing accompanying data.

7.372.2 Member Function Documentation

begin()

```
virtual Iterator JawsMako::IPDFDictionary::begin ( )  [pure virtual]
```

Create an iterator to iterate through all the key-value pairs in the dictionary, beginning at the first entry.

Returns

IPDFDictionary::Iterator an iterator positioned at the start of the dictionary

containsCompositeObject()

```
virtual bool JawsMako::IPDFDictionary::containsCompositeObject (
            const IPDFObjectPtr & object ) const  [pure virtual]
```

Recursively check to see if the dictionary contains the given composite object (That is, a dictionary, stream, or array).

Parameters

object	The object to search for.

Returns

bool True if the given object was found.

copy()

```
virtual void JawsMako::IPDFDictionary::copy (
            const IPDFDictionaryPtr & sourceDict,
            uint32 sourceIndex )  [pure virtual]
```

Copy (but not clone) an entry from another dictionary into this dictionary, replacing any currently stored object. Usually more efficient for bulk operations.

Parameters

sourceDict	The dictionary from which to copy.
sourceIndex	The index of the entry in sourceDict to copy.

create()

```
static JAWSMAKO_API IPDFDictionaryPtr JawsMako::IPDFDictionary::create (
            IEDLClassFactory * pFactory,
            uint32 size = 0 )  [static]
```

Create a new PDF dictionary of the given initial capacity.

Parameters

pFactory	The JawsMako instance or factory to use to create the dictionary.
size	The initial capacity of the dictionary. It may be beneficial to preallocate for performance.

Returns

IPDFDictionaryPtr The created object.

end()

```
virtual Iterator JawsMako::IPDFDictionary::end ( )  [pure virtual]
```

Obtain an interator representing the end of the dictionary.

Returns

IPDFDictionary::Iterator the iterator at the end of the dictionary

get() [1/3]

```
virtual IPDFObjectPtr JawsMako::IPDFDictionary::get (
            const char * key ) const  [pure virtual]
```

Get the object stored in the dictionary with the given null-terminated string as key.

Returns

IPDFObjectPtr The object stored under the given key, or NULL if there is no such object.

get() [2/3]

```
virtual IPDFObjectPtr JawsMako::IPDFDictionary::get (
            const IPDFNamePtr & key ) const  [pure virtual]
```

Get the object stored in the dictionary with the given DF name as key.

Returns

IPDFObjectPtr The object stored under the given key, or NULL if there is no such object.

get() [3/3]

```
virtual IPDFObjectPtr JawsMako::IPDFDictionary::get (
            const RawString & key ) const  [pure virtual]
```

Get the object stored in the dictionary with the given raw string as key.

Returns

IPDFObjectPtr The object stored under the given key, or NULL if there is no such object.

getKeyAtIndex()

```
virtual IPDFNamePtr JawsMako::IPDFDictionary::getKeyAtIndex (
            uint32 index ) const  [pure virtual]
```

Get the key of the dictionary entry at the given index.

Returns

IPDFNamePtr The key at the given index, or NULL if the dictionary entry is empty.

getSize()

```
virtual uint32 JawsMako::IPDFDictionary::getSize ( ) const  [pure virtual]
```

Get the current size of the dictionary, including currently empty entries.

Returns

uint32 The size.

getValueAtIndex()

```
virtual IPDFObjectPtr JawsMako::IPDFDictionary::getValueAtIndex (
            uint32 index ) const  [pure virtual]
```

Get the value stored in the dictionary entry at the given index.

Returns

IPDFObjectPtr The object at the given index, or NULL if the dictionary entry is empty.

getValueTypeAtIndex()

```
virtual ePDFObjectType JawsMako::IPDFDictionary::getValueTypeAtIndex (
            uint32 index ) const  [pure virtual]
```

Get the type of the value stored in the dictionary entry at the given index. An exception is thrown if the dictionary entry is empty.

Returns

ePDFObjectType The type of the object at the given index, or NULL if the dictionary slot is empty.

put() [1/3]

```
virtual void JawsMako::IPDFDictionary::put (
            const char * key,
            const IPDFObjectPtr & value )  [pure virtual]
```

Store an object in the dictionary under the given key.

Parameters

key	The null-terminated string to use as key.
value	The PDF object to store.

put() [2/3]

```
virtual void JawsMako::IPDFDictionary::put (
            const IPDFNamePtr & key,
            const IPDFObjectPtr & value )  [pure virtual]
```

Store an object in the dictionary under the given key.

Parameters

key	The name object to use as key.
value	The PDF object to store.

put() [3/3]

```
virtual void JawsMako::IPDFDictionary::put (
            const RawString & key,
            const IPDFObjectPtr & value )  [pure virtual]
```

Store an object in the dictionary under the given key.

Parameters

key	The raw string to use as key.
value	The PDF object to store.

putInteger()

```
virtual void JawsMako::IPDFDictionary::putInteger (
            const IPDFNamePtr & key,
            int32 integer )  [pure virtual]
```

Convenience member to add an Integer to a dictionary.

Parameters

key	The name object to use as key.
integer	The integer value to store.

putReal()

```
virtual void JawsMako::IPDFDictionary::putReal (
            const IPDFNamePtr & key,
            double real )  [pure virtual]
```

Convenience member to add a real to a dictionary.

Parameters

key	The name object to use as key.
real	The floating point value to store.

undefine() [1/3]

```
virtual void JawsMako::IPDFDictionary::undefine (
            const char * key )  [pure virtual]
```

Remove the object stored under the given key, if it exists.

Parameters

key	The null-terminated string to use as key.

undefine() [2/3]

```
virtual void JawsMako::IPDFDictionary::undefine (
            const IPDFNamePtr & key )  [pure virtual]
```

Remove the object stored under the given key, if it exists.

Parameters

key	The name object to use as key.

undefine() [3/3]

```
virtual void JawsMako::IPDFDictionary::undefine (
            const RawString & key )  [pure virtual]
```

Remove the object stored under the given key, if it exists.

Parameters

key	The raw string to use as key.

The documentation for this class was generated from the following file:

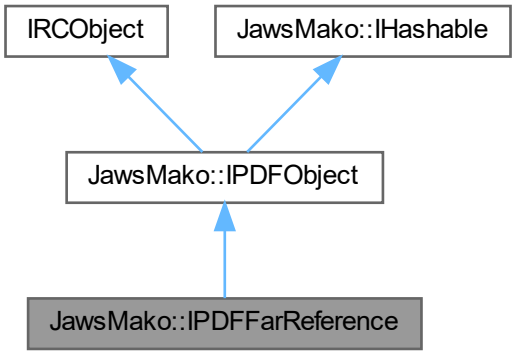
- pdfobjects.h

7.373 JawsMako::IPDFFarReference Class Reference

A simple class representing an immutable PDF indirect reference from a remote context such as (for example) from a different PDF document.

```
#include <pdfobjects.h>
```

Inheritance diagram for JawsMako::IPDFFarReference:



Public Member Functions

- virtual const CPDFFarReference & getValue () const =0

  Obtain the CPDFFarReference data for this reference.

Public Member Functions inherited from JawsMako::IPDFObject

- virtual ePDFObjectType getType () const =0

  Get the type of this PDF object.
- virtual IPDFObjectPtr clone () const =0

  Create a clone of this PDF object if appropriate. Note that simple immutable objects are not actually cloned and instead the same object is returned.
- virtual IPDFObjectPtr deepClone () const =0

  Create a deep clone of this PDF object if appropriate. Note that simple immutable objects are not actually cloned and instead the same object is returned. If the object is composite (an IPDFArray, IPDFDictionary or IPDFStream) then the object will be cloned as well as the consituent objects in a recursive fashion.
- virtual bool getIsExecutable () const =0

  Get whether or not the object is considered executable, such as an executable name, array, or operator.
- virtual bool containsReferences () const =0

  Get whether or not the object contains indirect references to other objects (that is, does the object or any constituent object contain IPDFReference or IPDFFarReference objects).
- virtual void **emitPostScriptCode** (const IOutputStreamPtr &dest) const =0

  Convert the object to PostScript code and write to the given stream. Only allowed for objects that are representable in PostScript.
- virtual bool getNumber (double &number) const =0

  If the object is a number (a real or integer) obtain that number.
- virtual bool getNumber (float &number) const =0

  If the object is a number (a real or integer) within the range of a 32-bit float, obtain that number. If the object is an integer, thiw will fail if the integer cannot be represented exactly in a single precision float.
- virtual bool getNumber (int32 &number) const =0

  If the object is an integer (a integral real or integer) within the range of a 32-bit integer, obtain that number.
- virtual bool getString (RawString &string) const =0

  If the object is a string or a name, retrieve the raw string data.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from JawsMako::IHashable

- virtual uint64 hash () const

  Obtain a 64-bit hash of the receiving object.
- virtual void updateHash (uint64 &hash) const =0

  Update the given hash to include the receiver.

Static Public Member Functions

- static JAWSMAKO_API IPDFFarReferencePtr create (const CPDFFarReference &reference)

  Create a remote indirect reference object from CPDFFarReference data.

Static Public Member Functions inherited from JawsMako::IPDFObject

- static JAWSMAKO_API IPDFObjectPtr createNumber (double number)

  If the given number is an integer, create an IPDFInteger, otherwise create an IPDFReal.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.373.1 Detailed Description

A simple class representing an immutable PDF indirect reference from a remote context such as (for example) from a different PDF document.

7.373.2 Member Function Documentation

create()

```
static JAWSMAKO_API IPDFFarReferencePtr JawsMako::IPDFFarReference::create (
            const CPDFFarReference & reference )  [static]
```

Create a remote indirect reference object from CPDFFarReference data.

Parameters

reference	The CPDFFarReference data to use.

Returns

IPDFFarReferencePtr The created object.

getValue()

```
virtual const CPDFFarReference & JawsMako::IPDFFarReference::getValue ( ) const  [pure virtual]
```

Obtain the CPDFFarReference data for this reference.

Returns

CPDFFarReference The reference data.

The documentation for this class was generated from the following file:

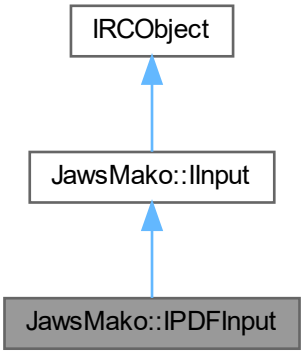
- pdfobjects.h

7.374 JawsMako::IPDFInput Class Reference

An instance of the JawsMako PDF input class.

```
#include <pdfinput.h>
```

Inheritance diagram for JawsMako::IPDFInput:



Classes

- class CPdfFontInfo

  Information about a font in a PDF file, obtained by scanning the PDF font structures.
- class CPdfScannedInk

  Basic information about an ink used in a PDF file, obtained by scanning the PDF page tree.

Public Member Functions

- virtual void setPassword (const U8String &password)=0

  Set the password that should be used when attempting to open PDF files. For password encrypted PDF, this is the user or owner password. For PDF files encrypted with public key cryptography, this is the password to be used to extract the private key from the PKCS12/PFX container. The default is no password.
- virtual void setPkcs12 (const IRAInputStreamPtr &pkcs12)=0

  Set the stream containing the PKCS12/PFX data containing the private key used to decrypt the PDF when public key encryption is used.
- virtual void setFailOnFontFallback (bool failOnFontFallback)=0

  Set whether or not to fail when a font cannot be found and PDF input needs to use a fallback font or create and emulated font.
- virtual void setDefaultRenderingIntent (eRenderingIntent intent)=0

  Override the default rendering intent in the PDF.
- virtual CPdfFontInfoVect scanPdfForFonts (const U8String &pathToFile, const IProgressMonitorPtr &progressMonitor, bool domFont=false)=0

  Quickly scan the PDF file at the given UTF-8 path for fonts. Does not attempt to validate the fonts, nor use them in any way. Instead, the PDF page tree is scanned and the PDF font objects interrogated. Annotations are scanned also.
- virtual CPdfFontInfoVect scanPdfForFonts (const String &pathToFile, const IProgressMonitorPtr &progress↩ Monitor, bool domFont=false)=0

  Quickly scan the PDF file at the given wide-character Unicode path for fonts. Does not attempt to validate the fonts, nor use them in any way. Instead, the PDF page tree is scanned and the PDF font objects interrogated. Annotations are scanned also.
- virtual CPdfFontInfoVect scanPdfForFonts (const IInputStreamPtr &pdfStream, const IProgressMonitorPtr &progressMonitor, bool domFont=false)=0

  Quickly scan the PDF file in the given stream for fonts. Does not attempt to validate the fonts, nor use them in any way. Instead, the PDF page tree is scanned and the PDF font objects interrogated. Annotations are scanned also.
- virtual CPdfScannedInkVect scanPdfForInks (const U8String &pathToFile, const IProgressMonitorPtr &progressMonitor, uint32 startPageIndex=0, uint32 endPageIndex=0)=0

  Quickly scan the PDF file at the given UTF-8 path and find any mentions of inks in color spaces.
- virtual CPdfScannedInkVect scanPdfForInks (const String &pathToFile, const IProgressMonitorPtr &progressMonitor, uint32 startPageIndex=0, uint32 endPageIndex=0)=0

  Quickly scan the PDF file at the given wide character path and find any mentions of inks in color spaces.
- virtual CPdfScannedInkVect scanPdfForInks (const IInputStreamPtr &pdfStream, const IProgressMonitorPtr &progressMonitor, uint32 startPageIndex=0, uint32 endPageIndex=0)=0

  Quickly scan the given PDF stream and find any mentions of inks in color spaces.
- virtual uint32 getNumIncrementalSaves (const U8String &pathToFile)=0

  Find the number of incremental saves in the given PDF stream.
- virtual uint32 getNumIncrementalSaves (const String &pathToFile)=0

  Find the number of incremental saves in the given PDF stream.
- virtual uint32 getNumIncrementalSaves (const IInputStreamPtr &pdfStream)=0

  Find the number of incremental saves in the given PDF stream.
- virtual IDocumentAssemblyPtr openIncremental (const U8String &pathToFile, uint32 incrementalIndex)=0

  Open an incremental PDF stream, returning the IDocumentAssembly representing the contents.
- virtual IDocumentAssemblyPtr openIncremental (const String &pathToFile, uint32 incrementalIndex)=0

Open an incremental PDF stream, returning the IDocumentAssembly representing the contents.

- virtual IDocumentAssemblyPtr openIncremental (const IInputStreamPtr &pdfStream, uint32 incrementalIndex)=0

  Open an incremental PDF stream, returning the IDocumentAssembly representing the contents.
- virtual CIRAInputStreamVect getIncrementalSaves (const U8String &pathToFile)=0

  Find the number of incremental saves in the given PDF stream, returning a vector of streams.
- virtual CIRAInputStreamVect getIncrementalSaves (const String &pathToFile)=0

  Find the number of incremental saves in the given PDF stream, returning a vector of streams.
- virtual CIRAInputStreamVect getIncrementalSaves (const IInputStreamPtr &pdfStream)=0

  Find the number of incremental saves in the given PDF stream, returning a vector of streams.

Public Member Functions inherited from JawsMako::IInput

- virtual IDocumentAssemblyPtr open (const U8String &pathToFile)=0

  Open a file on disk, returning the IDocumentAssembly representing the contents.
- virtual IDocumentAssemblyPtr open (const String &pathToFile)=0

  Open a file on disk, returning the IDocumentAssembly representing the contents. Takes a wide character string.
- virtual IDocumentAssemblyPtr open (const IInputStreamPtr &inputStream)=0

  Open a stream, returning the IDocumentAssembly representing the contents.
- virtual void setSequentialMode (bool sequential)=0

  Set/unset sequential mode on this input.
- virtual void setParameter (const U8String &param, const U8String &value)=0

  Apply a key value pair output parameter with a string value. The parameter name is case insensitive. Please refer to the supplied documentation for the details of the available parameters and their ranges.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IPDFInputPtr create (const IJawsMakoPtr &jawsMako, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create an input for reading source documents in PDF format.

Static Public Member Functions inherited from JawsMako::IInput

- static JAWSMAKO_API IInputPtr create (const IJawsMakoPtr &jawsMako, eFileFormat format, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create an input for reading source documents in the given format. The following formats are currently supported:

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()
  Virtual destructor.

7.374.1 Detailed Description

An instance of the JawsMako PDF input class.

7.374.2 Member Function Documentation

create()

```
static JAWSMAKO_API IPDFInputPtr JawsMako::IPDFInput::create (
            const IJawsMakoPtr & jawsMako,
            const IProgressMonitorPtr & progressMonitor = IProgressMonitorPtr() )  [static]
```

Create an input for reading source documents in PDF format.

Parameters

jawsMako	The IJawsMako object
progressMonitor	A smart pointer to an IProgressMonitor object which can be NULL if no such object was passed in.

Returns

IPDFInputPtr the PDF input

getIncrementalSaves() [1/3]

```
virtual CIRAInputStreamVect JawsMako::IPDFInput::getIncrementalSaves (
            const IInputStreamPtr & pdfStream )  [pure virtual]
```

Find the number of incremental saves in the given PDF stream, returning a vector of streams.

Parameters

pdfStream	The input PDF stream.

Returns

CIRAInputStreamVect A vector of incremental PDF streams, which will be zero length for PDF files saved without incremental update. Returned streams are not copied from the input, but are created from the input with createSubFile() and offset to the end of each incremental save.

getIncrementalSaves() [2/3]

```
virtual CIRAInputStreamVect JawsMako::IPDFInput::getIncrementalSaves (
            const String & pathToFile )  [pure virtual]
```

Find the number of incremental saves in the given PDF stream, returning a vector of streams.

Parameters

pathToFile	The path to the input file.

Returns

CIRAInputStreamVect A vector of incremental PDF streams, which will be zero length for PDF files saved without incremental update. Returned streams are not copied from the input, but are created from the input with createSubFile() and offset to the end of each incremental save.

getIncrementalSaves() [3/3]

```
virtual CIRAInputStreamVect JawsMako::IPDFInput::getIncrementalSaves (
            const U8String & pathToFile )  [pure virtual]
```

Find the number of incremental saves in the given PDF stream, returning a vector of streams.

Parameters

pathToFile	The path to the input file.

Returns

CIRAInputStreamVect A vector of incremental PDF streams, which will be zero length for PDF files saved without incremental update. Returned streams are not copied from the input, but are created from the input with createSubFile() and offset to the end of each incremental save.

getNumIncrementalSaves() [1/3]

```
virtual uint32 JawsMako::IPDFInput::getNumIncrementalSaves (
            const IInputStreamPtr & pdfStream )  [pure virtual]
```

Find the number of incremental saves in the given PDF stream.

Parameters

pdfStream	The input PDF stream.

Returns

uint32 The number of incremental saves in the PDF stream. Returns a value of zero for PDF files saved without incremental update.

getNumIncrementalSaves() [2/3]

```
virtual uint32 JawsMako::IPDFInput::getNumIncrementalSaves (
            const String & pathToFile )  [pure virtual]
```

Find the number of incremental saves in the given PDF stream.

Parameters

pathToFile	The path to the input file.

Returns

uint32 The number of incremental saves in the PDF stream. Returns a value of zero for PDF files saved without incremental update.

getNumIncrementalSaves() [3/3]

```
virtual uint32 JawsMako::IPDFInput::getNumIncrementalSaves (
            const U8String & pathToFile )  [pure virtual]
```

Find the number of incremental saves in the given PDF stream.

Parameters

pathToFile	The path to the input file.

Returns

uint32 The number of incremental saves in the PDF stream. Returns a value of zero for PDF files saved without incremental update.

openIncremental() [1/3]

```
virtual IDocumentAssemblyPtr JawsMako::IPDFInput::openIncremental (
            const IInputStreamPtr & pdfStream,
            uint32 incrementalIndex )  [pure virtual]
```

Open an incremental PDF stream, returning the IDocumentAssembly representing the contents.

Parameters

pdfStream	The input PDF stream.
incrementalUpdateIndex	The incremental update to load, with valid values being in the range 0 to getNumIncrementalSaves().

Returns

IDocumentAssemblyPtr The document assembly.

openIncremental() [2/3]

```
virtual IDocumentAssemblyPtr JawsMako::IPDFInput::openIncremental (
            const String & pathToFile,
            uint32 incrementalIndex )  [pure virtual]
```

Open an incremental PDF stream, returning the IDocumentAssembly representing the contents.

Parameters

pathToFile	The path to the input file.
incrementalUpdateIndex	The incremental update to load, with valid values being in the range 0 to getNumIncrementalSaves().

Returns

IDocumentAssemblyPtr The document assembly.

openIncremental() [3/3]

```
virtual IDocumentAssemblyPtr JawsMako::IPDFInput::openIncremental (
            const U8String & pathToFile,
            uint32 incrementalIndex )  [pure virtual]
```

Open an incremental PDF stream, returning the IDocumentAssembly representing the contents.

Parameters

pathToFile	The path to the input file.
incrementalUpdateIndex	The incremental update to load, with valid values being in the range 0 to getNumIncrementalSaves().

Returns

IDocumentAssemblyPtr The document assembly.

scanPdfForFonts() [1/3]

```
virtual CPdfFontInfoVect JawsMako::IPDFInput::scanPdfForFonts (
            const IInputStreamPtr & pdfStream,
            const IProgressMonitorPtr & progressMonitor,
            bool domFont = false )  [pure virtual]
```

Quickly scan the PDF file in the given stream for fonts. Does not attempt to validate the fonts, nor use them in any way. Instead, the PDF page tree is scanned and the PDF font objects interrogated. Annotations are scanned also.

You can pass in an optional callback function to receive progress information. From this function, execution may be aborted by throwing an abort exception using throwEDLError(EDL_ERR_ABORTED) which will propagate back to the caller.

When the optional domFont parameter is set to true, scanPdfFonts() will generate a IDOMFont from any found PDF font object.

Note: Even if a font is embedded, a given PDF consumer may decide not to use it and instead use a substitute, for example if the font is invalid or broken.

scanPdfForFonts() [2/3]

```
virtual CPdfFontInfoVect JawsMako::IPDFInput::scanPdfForFonts (
            const String & pathToFile,
            const IProgressMonitorPtr & progressMonitor,
            bool domFont = false )  [pure virtual]
```

Quickly scan the PDF file at the given wide-character Unicode path for fonts. Does not attempt to validate the fonts, nor use them in any way. Instead, the PDF page tree is scanned and the PDF font objects interrogated. Annotations are scanned also.

You can pass in an optional callback function to receive progress information. From this function, execution may be aborted by throwing an abort exception using throwEDLError(EDL_ERR_ABORTED) which will propagate back to the caller.

When the optional domFont parameter is set to true, scanPdfFonts() will generate a IDOMFont from any found PDF font object.

Note: Even if a font is embedded, a given PDF consumer may decide not to use it and instead use a substitute, for example if the font is invalid or broken.

scanPdfForFonts() [3/3]

```
virtual CPdfFontInfoVect JawsMako::IPDFInput::scanPdfForFonts (
            const U8String & pathToFile,
            const IProgressMonitorPtr & progressMonitor,
            bool domFont = false )  [pure virtual]
```

Quickly scan the PDF file at the given UTF-8 path for fonts. Does not attempt to validate the fonts, nor use them in any way. Instead, the PDF page tree is scanned and the PDF font objects interrogated. Annotations are scanned also.

You can pass in an optional callback function to receive progress information. From this function, execution may be aborted by throwing an abort exception using throwEDLError(EDL_ERR_ABORTED) which will propagate back to the caller.

When the optional domFont parameter is set to true, scanPdfFonts() will generate a IDOMFont from any found PDF font object.

Note: Even if a font is embedded, a given PDF consumer may decide not to use it and instead use a substitute, for example if the font is invalid or broken.

scanPdfForInks() [1/3]

```
virtual CPdfScannedInkVect JawsMako::IPDFInput::scanPdfForInks (
            const IInputStreamPtr & pdfStream,
            const IProgressMonitorPtr & progressMonitor,
            uint32 startPageIndex = 0,
            uint32 endPageIndex = 0 )  [pure virtual]
```

Quickly scan the given PDF stream and find any mentions of inks in color spaces.

Parameters

pdfStream	The input PDF stream to scan.
progressMonitor	A smart pointer to an IProgressMonitor object which can be NULL if no such object was passed in.
startPageIndex	Optional index of the first page to scan. Pages are numbered starting with page 0, which is the default value.
endPageIndex	Optional index of the end page. This is the index after the last page, where the number of pages to scan is equal to (endPageIndex - startPageIndex). Setting this parameter to 0 indicates that the function should scan up to and including the final page.

scanPdfForInks() [2/3]

```
virtual CPdfScannedInkVect JawsMako::IPDFInput::scanPdfForInks (
            const String & pathToFile,
            const IProgressMonitorPtr & progressMonitor,
            uint32 startPageIndex = 0,
            uint32 endPageIndex = 0 )  [pure virtual]
```

Quickly scan the PDF file at the given wide character path and find any mentions of inks in color spaces.

Parameters

pathToFile	The path to the input file to scan.
progressMonitor	A smart pointer to an IProgressMonitor object which can be NULL if no such object was passed in.
startPageIndex	Optional index of the first page to scan. Pages are numbered starting with page 0, which is the default value.
endPageIndex	Optional index of the end page. This is the index after the last page, where the number of pages to scan is equal to (endPageIndex - startPageIndex). Setting this parameter to 0 indicates that the function should scan up to and including the final page.

scanPdfForInks() [3/3]

```
virtual CPdfScannedInkVect JawsMako::IPDFInput::scanPdfForInks (
            const U8String & pathToFile,
            const IProgressMonitorPtr & progressMonitor,
            uint32 startPageIndex = 0,
            uint32 endPageIndex = 0 )  [pure virtual]
```

Quickly scan the PDF file at the given UTF-8 path and find any mentions of inks in color spaces.

Parameters

pathToFile	The path to the input file to scan.
progressMonitor	A smart pointer to an IProgressMonitor object which can be NULL if no such object was passed in.
startPageIndex	Optional index of the first page to scan. Pages are numbered starting with page 0, which is the default value.
endPageIndex	Optional index of the end page. This is the index after the last page, where the number of pages to scan is equal to (endPageIndex - startPageIndex). Setting this parameter to 0 indicates that the function should scan up to and including the final page.

setDefaultRenderingIntent()

```
virtual void JawsMako::IPDFInput::setDefaultRenderingIntent (
            eRenderingIntent intent )  [pure virtual]
```

Override the default rendering intent in the PDF.

```
The default is RelativeColorimetric, as per the PDF specification.
```

setFailOnFontFallback()

```
virtual void JawsMako::IPDFInput::setFailOnFontFallback (
            bool failOnFontFallback )  [pure virtual]
```

Set whether or not to fail when a font cannot be found and PDF input needs to use a fallback font or create and emulated font.

If false, font substitution and/or font emulation for missing fonts is allowed, which is sensible behaviour for general use cases. If a font is found to be invalid, a font may be sought on the users system or substituted.

If true, PDF input will fail and an exception thrown if a font requires the use of a fallback font or an emulated font, or if a damaged font is found and needs to be substituted.

The default is false.

setPassword()

```
virtual void JawsMako::IPDFInput::setPassword (
            const U8String & password )  [pure virtual]
```

Set the password that should be used when attempting to open PDF files. For password encrypted PDF, this is the user or owner password. For PDF files encrypted with public key cryptography, this is the password to be used to extract the private key from the PKCS12/PFX container. The default is no password.

Parameters

password	The password to use. Assumed to be UTF8.

setPkcs12()

```
virtual void JawsMako::IPDFInput::setPkcs12 (
            const IRAInputStreamPtr & pkcs12 )  [pure virtual]
```

Set the stream containing the PKCS12/PFX data containing the private key used to decrypt the PDF when public key encryption is used.

Note: Public Key Encryption is only supported on Linux and Windows (non-UWP) platforms at this time.

Parameters

pkcs12	A random access stream pointing to a pkcs12/pfx file for the user that is authorised as a recipient for this PDF.

The documentation for this class was generated from the following file:

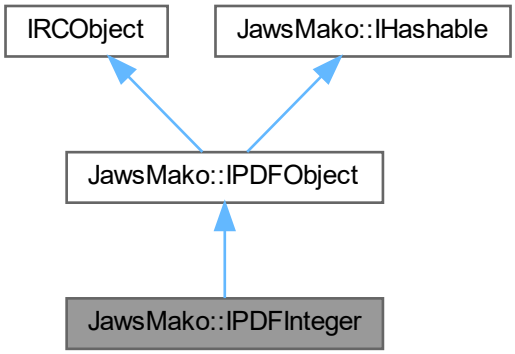
- pdfinput.h

7.375 JawsMako::IPDFInteger Class Reference

A simple immutable integer PDF object type.

`#include <pdfobjects.h>`

Inheritance diagram for JawsMako::IPDFInteger:



Public Member Functions

- virtual int32 getValue () const =0

  Get the value of the integer.

Public Member Functions inherited from JawsMako::IPDFObject

- virtual ePDFObjectType getType () const =0

  Get the type of this PDF object.
- virtual IPDFObjectPtr clone () const =0

  Create a clone of this PDF object if appropriate. Note that simple immutable objects are not actually cloned and instead the same object is returned.
- virtual IPDFObjectPtr deepClone () const =0

Create a deep clone of this PDF object if appropriate. Note that simple immutable objects are not actually cloned and instead the same object is returned. If the object is composite (an IPDFArray, IPDFDictionary or IPDFStream) then the object will be cloned as well as the consituent objects in a recursive fashion.

- virtual bool getIsExecutable () const =0

  Get whether or not the object is considered executable, such as an executable name, array, or operator.
- virtual bool containsReferences () const =0

  Get whether or not the object contains indirect references to other objects (that is, does the object or any constituent object contain IPDFReference or IPDFFarReference objects).
- virtual void **emitPostScriptCode** (const IOutputStreamPtr &dest) const =0

  Convert the object to PostScript code and write to the given stream. Only allowed for objects that are representable in PostScript.
- virtual bool getNumber (double &number) const =0

  If the object is a number (a real or integer) obtain that number.
- virtual bool getNumber (float &number) const =0

  If the object is a number (a real or integer) within the range of a 32-bit float, obtain that number. If the object is an integer, thiw will fail if the integer cannot be represented exactly in a single precision float.
- virtual bool getNumber (int32 &number) const =0

  If the object is an integer (a integral real or integer) within the range of a 32-bit integer, obtain that number.
- virtual bool getString (RawString &string) const =0

  If the object is a string or a name, retrieve the raw string data.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from JawsMako::IHashable

- virtual uint64 hash () const

  Obtain a 64-bit hash of the receiving object.
- virtual void updateHash (uint64 &hash) const =0

  Update the given hash to include the receiver.

Static Public Member Functions

- static JAWSMAKO_API IPDFIntegerPtr create (int32 integer)

  Create an integer object.

Static Public Member Functions inherited from JawsMako::IPDFObject

- static JAWSMAKO_API IPDFObjectPtr createNumber (double number)

  If the given number is an integer, create an IPDFInteger, otherwise create an IPDFReal.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()
  Virtual destructor.

7.375.1 Detailed Description

A simple immutable integer PDF object type.

7.375.2 Member Function Documentation

create()

```
static JAWSMAKO_API IPDFIntegerPtr JawsMako::IPDFInteger::create (
            int32 integer )  [static]
```

Create an integer object.

Parameters

integer	The integer value.

Returns

IPDFIntegerPtr The created object.

getValue()

```
virtual int32 JawsMako::IPDFInteger::getValue ( ) const  [pure virtual]
```

Get the value of the integer.

Returns

int32 The value of the integer.

The documentation for this class was generated from the following file:

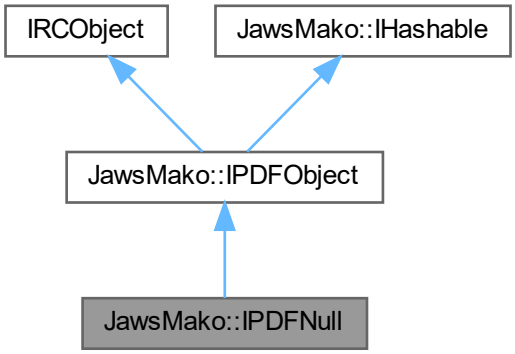
- pdfobjects.h

7.376 JawsMako::IPDFNull Class Reference

A simple immutable "null" pdf object.

`#include <pdfobjects.h>`

Inheritance diagram for JawsMako::IPDFNull:



Static Public Member Functions

- static JAWSMAKO_API IPDFNullPtr create ()

  Create a "null" object.

Static Public Member Functions inherited from JawsMako::IPDFObject

- static JAWSMAKO_API IPDFObjectPtr createNumber (double number)

  If the given number is an integer, create an IPDFInteger, otherwise create an IPDFReal.

Additional Inherited Members

Public Member Functions inherited from JawsMako::IPDFObject

- virtual ePDFObjectType getType () const =0

  Get the type of this PDF object.
- virtual IPDFObjectPtr clone () const =0

  Create a clone of this PDF object if appropriate. Note that simple immutable objects are not actually cloned and instead the same object is returned.
- virtual IPDFObjectPtr deepClone () const =0

  Create a deep clone of this PDF object if appropriate. Note that simple immutable objects are not actually cloned and instead the same object is returned. If the object is composite (an IPDFArray, IPDFDictionary or IPDFStream) then the object will be cloned as well as the consituent objects in a recursive fashion.
- virtual bool getIsExecutable () const =0

  Get whether or not the object is considered executable, such as an executable name, array, or operator.

- virtual bool containsReferences () const =0

  Get whether or not the object contains indirect references to other objects (that is, does the object or any constituent object contain IPDFReference or IPDFFarReference objects).
- virtual void **emitPostScriptCode** (const IOutputStreamPtr &dest) const =0

  Convert the object to PostScript code and write to the given stream. Only allowed for objects that are representable in PostScript.
- virtual bool getNumber (double &number) const =0

  If the object is a number (a real or integer) obtain that number.
- virtual bool getNumber (float &number) const =0

  If the object is a number (a real or integer) within the range of a 32-bit float, obtain that number. If the object is an integer, thiw will fail if the integer cannot be represented exactly in a single precision float.
- virtual bool getNumber (int32 &number) const =0

  If the object is an integer (a integral real or integer) within the range of a 32-bit integer, obtain that number.
- virtual bool getString (RawString &string) const =0

  If the object is a string or a name, retrieve the raw string data.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from JawsMako::IHashable

- virtual uint64 hash () const

  Obtain a 64-bit hash of the receiving object.
- virtual void updateHash (uint64 &hash) const =0

  Update the given hash to include the receiver.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.376.1 Detailed Description

A simple immutable "null" pdf object.

7.376.2 Member Function Documentation

create()

```
static JAWSMAKO_API IPDFNullPtr JawsMako::IPDFNull::create ( )  [static]
```

Create a "null" object.

Returns

IPDFNullPtr The created object.

The documentation for this class was generated from the following file:

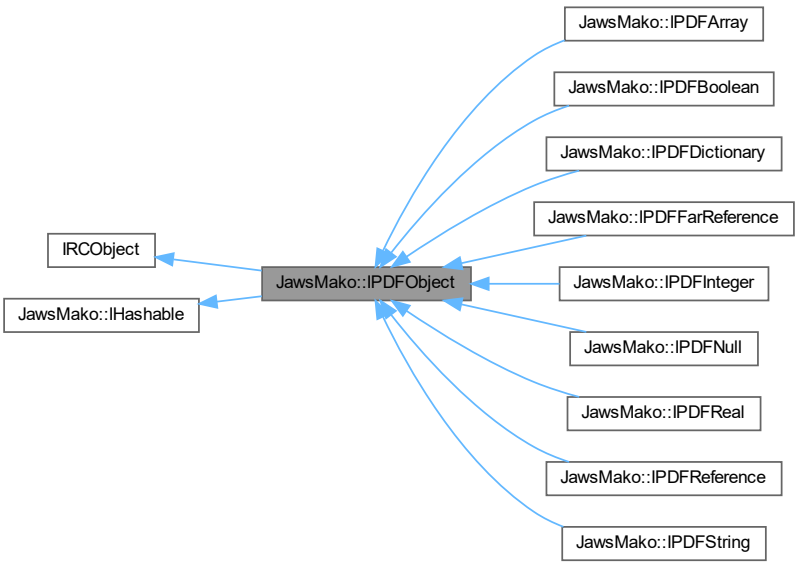
- pdfobjects.h

7.377 JawsMako::IPDFObject Class Reference

Abstract interface for a PDF internal object and common interfaces. All non-composite objects are immutable. All PDF objects are hashable.

```
#include <pdfobjects.h>
```

Inheritance diagram for JawsMako::IPDFObject:



Public Member Functions

- virtual ePDFObjectType getType () const =0

  Get the type of this PDF object.
- virtual IPDFObjectPtr clone () const =0

  Create a clone of this PDF object if appropriate. Note that simple immutable objects are not actually cloned and instead the same object is returned.
- virtual IPDFObjectPtr deepClone () const =0

  Create a deep clone of this PDF object if appropriate. Note that simple immutable objects are not actually cloned and instead the same object is returned. If the object is composite (an IPDFArray, IPDFDictionary or IPDFStream) then the object will be cloned as well as the consituent objects in a recursive fashion.
- virtual bool getIsExecutable () const =0

  Get whether or not the object is considered executable, such as an executable name, array, or operator.
- virtual bool containsReferences () const =0

  Get whether or not the object contains indirect references to other objects (that is, does the object or any constituent object contain IPDFReference or IPDFFarReference objects).
- virtual void **emitPostScriptCode** (const IOutputStreamPtr &dest) const =0

  Convert the object to PostScript code and write to the given stream. Only allowed for objects that are representable in PostScript.
- virtual bool getNumber (double &number) const =0

  If the object is a number (a real or integer) obtain that number.
- virtual bool getNumber (float &number) const =0

  If the object is a number (a real or integer) within the range of a 32-bit float, obtain that number. If the object is an integer, thiw will fail if the integer cannot be represented exactly in a single precision float.
- virtual bool getNumber (int32 &number) const =0

  If the object is an integer (a integral real or integer) within the range of a 32-bit integer, obtain that number.
- virtual bool getString (RawString &string) const =0

  If the object is a string or a name, retrieve the raw string data.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from JawsMako::IHashable

- virtual uint64 hash () const

  Obtain a 64-bit hash of the receiving object.
- virtual void updateHash (uint64 &hash) const =0

  Update the given hash to include the receiver.

Static Public Member Functions

- static JAWSMAKO_API IPDFObjectPtr createNumber (double number)

  If the given number is an integer, create an IPDFInteger, otherwise create an IPDFReal.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.377.1 Detailed Description

Abstract interface for a PDF internal object and common interfaces. All non-composite objects are immutable. All PDF objects are hashable.

7.377.2 Member Function Documentation

clone()

```
virtual IPDFObjectPtr JawsMako::IPDFObject::clone ( ) const  [pure virtual]
```

Create a clone of this PDF object if appropriate. Note that simple immutable objects are not actually cloned and instead the same object is returned.

Returns

IPDFObjectPtr The cloned object, or simply the object if immutable.

containsReferences()

```
virtual bool JawsMako::IPDFObject::containsReferences ( ) const  [pure virtual]
```

Get whether or not the object contains indirect references to other objects (that is, does the object or any constituent object contain IPDFReference or IPDFFarReference objects).

Returns

bool True if one or more reference objects are found, false otherwise.

createNumber()

```
static JAWSMAKO_API IPDFObjectPtr JawsMako::IPDFObject::createNumber (
            double number )  [static]
```

If the given number is an integer, create an IPDFInteger, otherwise create an IPDFReal.

Parameters

number	The number to create the object for.

Returns

IPDFObjectPtr Either an IPDFReal or an IPDFInteger as appropriate.

deepClone()

```
virtual IPDFObjectPtr JawsMako::IPDFObject::deepClone ( ) const  [pure virtual]
```

Create a deep clone of this PDF object if appropriate. Note that simple immutable objects are not actually cloned and instead the same object is returned. If the object is composite (an IPDFArray, IPDFDictionary or IPDFStream) then the object will be cloned as well as the consituent objects in a recursive fashion.

Returns

IPDFObjectPtr The cloned object, or simply the object if immutable.

getIsExecutable()

```
virtual bool JawsMako::IPDFObject::getIsExecutable ( ) const  [pure virtual]
```

Get whether or not the object is considered executable, such as an executable name, array, or operator.

Returns

bool True if the object is executable, false otherwise

getNumber() [1/3]

```
virtual bool JawsMako::IPDFObject::getNumber (
            double & number ) const  [pure virtual]
```

If the object is a number (a real or integer) obtain that number.

Parameters

number	A reference to receive the number

Returns

bool Returns true if the object is a number, false otherwise (in which case the value of number is undefined)

getNumber() [2/3]

```
virtual bool JawsMako::IPDFObject::getNumber (
            float & number ) const  [pure virtual]
```

If the object is a number (a real or integer) within the range of a 32-bit float, obtain that number. If the object is an integer, thiw will fail if the integer cannot be represented exactly in a single precision float.

Parameters

number	A reference to receive the number

Returns

bool Returns true if the object is such a number, false otherwise (in which case the value of number is undefined)

getNumber() [3/3]

```
virtual bool JawsMako::IPDFObject::getNumber (
            int32 & number ) const  [pure virtual]
```

If the object is an integer (a integral real or integer) within the range of a 32-bit integer, obtain that number.

Parameters

number	A reference to receive the number

Returns

bool Returns true if the object is such a number, false otherwise (in which case the value of number is undefined)

getString()

```
virtual bool JawsMako::IPDFObject::getString (
            RawString & string ) const  [pure virtual]
```

If the object is a string or a name, retrieve the raw string data.

Parameters

string	A reference to receive the string.

Returns

bool Returns true if the object is a name or string, false otherwise (in which case string will be undefined).

getType()

```
virtual ePDFObjectType JawsMako::IPDFObject::getType ( ) const  [pure virtual]
```

Get the type of this PDF object.

Returns

ePDFObjectType The type of this object

The documentation for this class was generated from the following file:

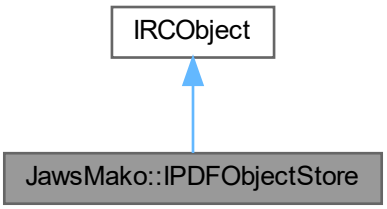
- pdfobjects.h

7.378 JawsMako::IPDFObjectStore Class Reference

A store of IPDFObjects holding a subset/subtree of an entire PDF object database, allowing storage, resolution and editing thereof. Every object store will have a "root" object from which the subtree or subset stems.

`#include <pdfobjects.h>`

Inheritance diagram for JawsMako::IPDFObjectStore:



Public Member Functions

- virtual IPDFObjectStorePtr clone () const =0

  Return a clone of this object store.
- virtual IPDFFarReferencePtr getRootObjectReference () const =0

  Get the far reference for the root object in this store.
- virtual IPDFObjectPtr getRootObject () const =0

  Get the root object from the store.
- virtual bool hasReferencedObject (const IPDFObjectPtr &reference) const =0

  Is the object referred to by the given IPDFReference or IPDFFarReference present in this store?
- virtual IPDFObjectPtr **resolve** (const IPDFObjectPtr &object) const =0

  Resolve the given object, if an IPDFReference or IPDFFarReference to the object it points to. This can take any object, and for objects that are not a reference, the original object will be returned. An exception will be thrown if the input is a reference and the store does not have the target object. hasReferencedObject() can be used to check this.
- virtual void dirtyObject (const IPDFObjectPtr &object)=0

  Mark the given object as dirty so that PDF output will know that it has changed. Only mutable objects (arrays, dictionarys, streams) and references can be marked dirty, as only they can be edited. Use this function whenever you edit an item present inside the store.
- virtual void **dirtyRootObject** ()=0

  Convenience to mark the root object as dirty. See dirtyObject() above.
- virtual IPDFReferencePtr newReference ()=0

Allocate a new indirect reference to use for a stored object.

- virtual void store (const IPDFObjectPtr &object, const IPDFReferencePtr &reference, bool markDirty=true)=0

  Store an object in the store under the given reference.
- virtual IPDFReferencePtr storeUsingNewReference (const IPDFObjectPtr &object)

  Store an object and return a new indirect reference to that object. A convenience that performs newReference() and store() in a single member.
- virtual IPDFFarReferencePtr getFarReferenceForReference (const IPDFReferencePtr &reference) const =0

  For a given resource present in this store, produce a far reference that can be used to point to it from another store.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.378.1 Detailed Description

A store of IPDFObjects holding a subset/subtree of an entire PDF object database, allowing storage, resolution and editing thereof. Every object store will have a "root" object from which the subtree or subset stems.

7.378.2 Member Function Documentation

clone()

```
virtual IPDFObjectStorePtr JawsMako::IPDFObjectStore::clone ( ) const  [pure virtual]
```

Return a clone of this object store.

Returns

IPDFObjectStorePtr The clone

dirtyObject()

```
virtual void JawsMako::IPDFObjectStore::dirtyObject (
            const IPDFObjectPtr & object )  [pure virtual]
```

Mark the given object as dirty so that PDF output will know that it has changed. Only mutable objects (arrays, dictionarys, streams) and references can be marked dirty, as only they can be edited. Use this function whenever you edit an item present inside the store.

Parameters

object	The edited object.

getFarReferenceForReference()

```
virtual IPDFFarReferencePtr JawsMako::IPDFObjectStore::getFarReferenceForReference (
            const IPDFReferencePtr & reference ) const  [pure virtual]
```

For a given resource present in this store, produce a far reference that can be used to point to it from another store.

Parameters

reference	The reference to use.

Returns

IPDFFarReferencePtr The resulting far reference.

getRootObject()

```
virtual IPDFObjectPtr JawsMako::IPDFObjectStore::getRootObject ( ) const  [pure virtual]
```

Get the root object from the store.

Returns

IPDFObjectPtr The root object

getRootObjectReference()

```
virtual IPDFFarReferencePtr JawsMako::IPDFObjectStore::getRootObjectReference ( ) const  [pure
virtual]
```

Get the far reference for the root object in this store.

Returns

IPDFFarReferencePtr The far reference to the root object

hasReferencedObject()

```
virtual bool JawsMako::IPDFObjectStore::hasReferencedObject (
            const IPDFObjectPtr & reference ) const  [pure virtual]
```

Is the object referred to by the given IPDFReference or IPDFFarReference present in this store?

Parameters

reference	Either an IPDFFarReference or IPDFReference to check

Returns

bool True if the referenced object is present, false otherwise.

newReference()

```
virtual IPDFReferencePtr JawsMako::IPDFObjectStore::newReference ( )  [pure virtual]
```

Allocate a new indirect reference to use for a stored object.

Returns

IPDFReferencePtr The new reference.

store()

```
virtual void JawsMako::IPDFObjectStore::store (
            const IPDFObjectPtr & object,
            const IPDFReferencePtr & reference,
            bool markDirty = true )  [pure virtual]
```

Store an object in the store under the given reference.

Parameters

object	The object to store
reference	The reference under which the object is stored. Must have previously been allocated using newReference() above.
markDirty	If true, mark the object as dirty.

storeUsingNewReference()

```
virtual IPDFReferencePtr JawsMako::IPDFObjectStore::storeUsingNewReference (
            const IPDFObjectPtr & object )  [inline], [virtual]
```

Store an object and return a new indirect reference to that object. A convenience that performs newReference() and store() in a single member.

Parameters

object	The object to store.

Returns

IPDFReferencePtr The new reference under which the object was stored.

The documentation for this class was generated from the following file:

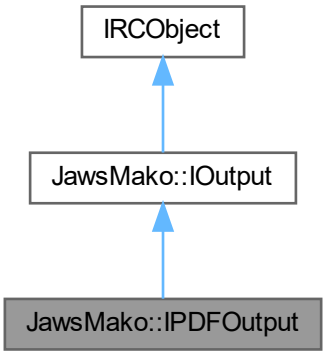
- pdfobjects.h

7.379 JawsMako::IPDFOutput Class Reference

Interface for the PDF IOutput class.

`#include <pdfoutput.h>`

Inheritance diagram for JawsMako::IPDFOutput:



Public Types

- enum ePDFVersion {
  ePDF1_3 = MAKE_PDF_VERSION(1,3) , ePDF1_4 = MAKE_PDF_VERSION(1,4) , ePDF1_5 = MAKE_↩
  PDF_VERSION(1,5) , ePDF1_6 = MAKE_PDF_VERSION(1,6) ,
  ePDF1_7 = MAKE_PDF_VERSION(1,7) , ePDF2_0 = MAKE_PDF_VERSION(2,0) , ePDFA1b = MAKE↩
  _PDF_VERSION(1,4) | (MAKE_PDF_VARIANT (65, 1, 98)) , ePDFA2b = MAKE_PDF_VERSION(1,7) |
  (MAKE_PDF_VARIANT (65, 2, 98)) ,
  ePDFA3b = MAKE_PDF_VERSION(1,7) | (MAKE_PDF_VARIANT (65, 3, 98)) , ePDFA2u = MAKE_PDF_↩
  VERSION(1,7) | (MAKE_PDF_VARIANT (65, 2, 117)) , ePDFUA = MAKE_PDF_VERSION(1,7) | (MAKE_↩
  PDF_VARIANT (65, 2, 85)) , ePDFX1a = MAKE_PDF_VERSION(1,4) | (MAKE_PDF_VARIANT (88, 1, 97)) ,
  ePDFX4 = MAKE_PDF_VERSION(1,6) | (MAKE_PDF_VARIANT (88, 4, 0)) }

  Supported versions.
- enum eImageCompression { }

  Enumeration for image compression formats.
- enum ePdfDeviceNHandling {
  ePDNHDefault = 0 , ePDNHFailOnMissingColorantInformation = 0x1 , ePDNHFailOnInconsistentColorantInformation
  = 0x2 , ePDNHFailIfBetterColorantInformationFoundAfterFirstUse = 0x4 ,
  ePDNHFailIfDotGainFound = 0x8 , ePDNFailOnMismatchedDeviceNProcessSpace = 0x10 , ePDNFailOnNonUtf8ColorantName
  = 0x20 }

Flags controlling PDF/X-4 and PDF/A-2 DeviceN handling.

- enum ePdfOptionalContentHandling { ePOCHDefault = 0 , ePOCHFailOnInconsistentConfigurationName = 0x1 , ePOCHFailOnInvalidOrder = 0x2 , ePOCHFailIfAutoStateFound = 0x4 }

  Flags controlling PDF/X-4 and PDF/A-2 Optional content.
  Allows some control on what the PDF writer is allowed to invent, or what inconsistencies are considered safe to ignore.
  Should an exception need to be raised, an IError with code EDL_ERR_INCOMPATIBLE_PDFX or EDL_ERR_↩ INCOMPATIBLE_PDFA (depending on output version) will be thrown, where the error description will contain a short message outlining the problem that was encountered.
- enum ePdfExtendedGraphicsStateHandling { ePEGSHDefault = 0 , ePEGSHFailOnTransferFunction = 0x1 }

  Flags controlling PDF/X-4 and PDF/A-2 Extended Graphics State information Allows some control on what the PDF writer is allowed to invent, or what inconsistencies are considered safe to ignore.
- enum ePdfAlternateImageHandling { ePAIHDefault = 0 , ePAIHFailIfAlternatesFound = 0x1 , ePAIHFailIfDefaultForPrintingFound = 0x2 }

  Flags controlling PDF/X-4 and PDF/A-2 alternate image handling.
  Allows control on what the PDF/X writer is allowed to invent, or what inconsistencies are considered safe to ignore.
  For PDF/X-4, Alternate images are allowed, but no alternate image is allowed to specify that it is the default for printing (via the DefaultForPrinting key). By default, if an image with DefaultForPrinting set to true is present, then this image will be selected and the other alternate images shall be dropped.
  For PDF/A-2b/u, Alternate images are not allowed at all. By default, we will apply the optional content rules for alternate images to select a single image as if the PDF was being printed.
  Instead of this default, the following flags allow processing to stop in these situations.
  Should an exception need to be raised, an IError with code EDL_ERR_INCOMPATIBLE_PDFX or EDL_ERR_↩ INCOMPATIBLE_PDFA (depending on output version) will be thrown, where the error description will contain a short message outlining the problem that was encountered.
- enum eFormCJKCharacterSet

  A CJK character set.

Public Member Functions

- virtual void setVersion (ePDFVersion version)=0

  Set the PDF version to generate.
- virtual void setProducer (const U8String &producer)=0

  Set the Producer name for the output PDF.
- virtual void setEnableIncrementalOutput (bool enable=true)=0

  Set whether or not incremental output should be enabled.
- virtual void setLinearize (bool linearize=true)=0

  Set whether or not the output should be linearized.
- virtual void setEncryption (uint32 keyLength, uint32 permissions, bool encryptMetadata=false, const U8String &ownerPassword="", const U8String &userPassword="")=0

  Set the encryption for the output PDF. The default is no encryption.
- virtual void setPublicKeyEncryption (uint32 keyLength, const CEDLVector< CPDFRecipientsInfo > &recipients, bool encryptMetadata=false)=0

  Use certificate/public-key encryption for the output PDF. The default is no encryption.
- virtual void setTargetColorSpace (const IDOMColorSpacePtr &targetSpace)=0

  Set the target color space for the output.
- virtual void setTargetProfile (const IDOMICCProfilePtr &profile)=0

  Set the target color space for the output using an ICC profile.
- virtual void setOutputIntent (const IOutputIntentPtr &outputIntent)=0

  Set an explicit output intent, or NULL to clear it.
- virtual void setOutputIntents (const COutputIntentVect &outputIntents)=0

  Set a vector of output intents, or an empty vector to clear them.
- virtual void setConvertAllColors (bool convert)=0

  Set whether or not all content should be color converted to the target space set by setTargetColorSpace().

- virtual void setConvertGray (bool convert)=0

  Set whether or not gray colors should be subject to color conversion. That is, colors using DeviceGray, sGray, or single-component ICC colorspaces.
- virtual void setColorImageMaxResolution (float resolution, float threshold=0.0f, IDOMImageDownsamplerFilter::eDownsampling method=IDOMImageDownsamplerFilter::eBicubic)=0

  Set the desired maximum resolution, threshold and downsampling method for colour images.
- virtual void setGrayImageMaxResolution (float resolution, float threshold=0.0f, IDOMImageDownsamplerFilter::eDownsampling method=IDOMImageDownsamplerFilter::eBicubic)=0

  Set the desired maximum resolution, threshold and downsampling method for gray images.
- virtual void setMonoImageMaxResolution (float resolution, float threshold=0.0f, IDOMImageDownsamplerFilter::eDownsampling method=IDOMImageDownsamplerFilter::eSubsample)=0

  Set the desired maximum resolution, threshold and downsampling method for monochrome images.
- virtual void setDownsampleMaskedImages (bool downsampleMaskedImages)=0

  Set whether or not to downsample masked images.
- virtual void setUseMaskResolutionForMaskedImages (bool useMaskResolutionSettingForMaskedImages)=0

  Set whether or not the mask resolution setting should be applied to the image portion of a masked image.
- virtual void setRenderResolution (uint32 resolution)=0

  Set the resolution to use if page content requires rendering in order to be output. The default is 300dpi. This is affected also by the maximum image resolution parameters. Equivalent to calling setParameter() with param name "RenderResolution" with the value as the desired resolution as value.
- virtual void setPreferredColorImageCompression (eImageCompression compression)=0

  Set the desired image compression for color images that need to be reencoded. The default is eICAuto.
- virtual void setPreferredGrayImageCompression (eImageCompression compression)=0

  Set the desired image compression for gray images that need to be reencoded. The default is eICAuto.
- virtual void setPreferredMonoImageCompression (eImageCompression compression)=0

  Set the desired image compression for monochrome images that need to be reencoded. The default is eICCCITT. JPEG/DCT is not allowed for mono images, and auto will be set to CCITT.
- virtual void setPreferredRenderedImageCompression (eImageCompression compression)=0

  Set the desired image compression for images that are the result of rendering. The default is eICFlate. If set to auto, flate will be used.
- virtual void setJPEGQuality (uint8 quality)=0

  Set the JPEG quality to use when compressing images in DCT format. Equivalent to calling setParameter() with the parameter name "JPEGQuality" and the value being the required quality.
- virtual void setJPEGChromaSubsampling (bool subsample)=0

  Set whether or not to use chroma subsampling when compressing color images in DCT/JPEG format. Chroma subsampling improves compression by reducing the resolution of color information in the image, leaving luminance information at high resolution.
- virtual void setReencodeImages (bool reencode)=0

  Set whether images should be reencoded.
- virtual void setCompressPages (bool compressPages)=0

  Set whether or not compression should be applied to page content. The default is true.
- virtual void setCompressObjects (bool compressObjects)=0

  Set whether or not to compress individual objects.
- virtual void setAutoRotatePages (bool autoRotate)=0

  Set whether or not pages should be automatically rotated. The default is false.
- virtual void setSubsetFonts (bool subset)=0

  Set whether embedded fonts should be subset or not. The default is true.
- virtual void setEmbedFonts (bool embed)=0

  Set whether fonts should be forcibly embedded. If true, this overrides the embedding settings in each IDOMFontOpenType. The standard 14 fonts are not affected by this; use setEmbedBaseFonts() instead.
- virtual void setEmbedBase14Fonts (bool embed)=0

  Set whether the base 14 PDF fonts should be forcibly embedded. If true, this overrides the embedding settings in IDOMFontOpenType. Only the standard 14 fonts are not affected by this; use setEmbedFonts() to affect other fonts.

- virtual void setAlwaysEmbedFonts (CU8StringVect &fontNames)=0

  Provide a list of name of fonts that should be always embedded, regardless of the settings for setEmbedFonts() or setEmbedBase14Fonts(). This setting is ignored for PDF/A.
- virtual void setNeverEmbedFonts (CU8StringVect &fontNames)=0

  Provide a list of name of fonts that should never be embedded, regardless of the settings for setEmbedFonts() or setEmbedBase14Fonts(). This setting is ignored for PDF/A.
- virtual void setAllowRestrictedFonts (bool allowRestrictedFonts)=0

  Allow restricted fonts to be embedded in the output.
- virtual void setEnableTrueTypeNotDef (bool enableTrueTypeNotDef)=0

  Enable the use of a True Type font's /.notdef glyph in the output.
- virtual void setBlockNotdefGlyphs (bool blockNotdefGlyphs)=0

  Set whether to block notdef glyphs from PDF output.
- virtual void setEmit30CmapSubtableForSymbolicTrueTypeFonts (bool emit)=0

  Set whether to add a 3,0 ("Windows") 'cmap' subtable to symbolic TrueType fonts in PDF output.
- virtual void setForceEmitPageGroup (bool force)=0

  Set whether or not the page group should be forcibly written.
- virtual void setForceMediaBoxOriginZero (bool force)=0

  Set whether or not the media box origin should be set to 0,0 in the output.
- virtual void overrideMaximumICCVersion (uint32 majorVersion, uint32 minorVersion)=0

  Override the maximum allowed ICC profile version for PDF output.
- virtual void setDeviceNErrorHandling (uint32 flags)=0

  Set flags controlling PDF/X-4 and PDF/A-2 DeviceN handling. See ePdfDeviceNHandling# for details.
- virtual void setOptionalContentErrorHandling (uint32 flags)=0

  Set flags controlling PDF/X-4 and PDF/A-2 Optional Content handling. See ePdfOptionalContentHandling# for details.
- virtual void setExtendedGraphicsStateErrorHandling (uint32 flags)=0

  Set flags controlling PDF/X-4 and PDF/A-2 Extended Graphics State handling. See ePdfExtendedGraphicsStateHandling# for details.
- virtual void setAlternateImageErrorHandling (uint32 flags)=0

  Set flags controlling PDF/X-4 and PDF/A-2b alternate image handling See ePdfAlternateImageHandling# for details.
- virtual void setNonUtf8InkNameFallbackEncoding (eInkFallbackEncoding fallbackEncoding)=0

  Set the fallback encoding to use when attempting to decode non-UTF8 compatible ink names for PDF/A-2 and PDF/X-4 output. Only consulted when ePDNFailOnNonUtf8ColorantName is unset in setDeviceNErrorHandling. See the two-parameter method of inkNameToString for how this is done.
- virtual void setDefaultFormCJKLanguage (eFormCJKCharacterSet characterSet)=0

  Set the default CJK character set for forms. Used when generating form appearance streams, and is only used when the character set cannot be determined from the content of the form. The default is Japanese.
- virtual void setRetainEmbeddedFiles (bool retainEmbeddedFiles)=0

  Allow embedded files to be retained in PDF/A and PDF/X-4 output.
- virtual void setRetainClippedContent (bool retainClippedContent)=0

  Allow clipped out content to be retained in the output.
- virtual void setAllowOptionalContentUpdateDuringWrite (bool allow)=0

  Allow updates to optional content to be made while the output is being written using an IOutputWriter.
- virtual void setValidateXmp (bool validate)=0

  Validate XMP metadata.

Public Member Functions inherited from JawsMako::IOutput

- virtual void **setPreset** (const U8String &preset)=0

  Configure the output according to a general preset. Please see the supplied documentation for details of these presets. The default is "Preserve" which will attempt to produce output as close to the input as possible for the output format. A string value can be used for any parameter and will be converted as necessary.
- virtual void **setParameter** (const U8String &param, const U8String &value)=0

Apply a key value pair output parameter with a string value. The parameter name is case insensitive. Please refer to the supplied documentation for the details of the available parameters and their ranges.

- virtual void setAllowedPermissionsFlags (uint32 allowedPermissions)=0

  Control whether or not assemblies with certain security permission flags are allowed to be written by this output.
- virtual void writeAssembly (const IDocumentAssemblyPtr &assembly, const U8String &pathToFile)=0

  Write the given document assembly to a file on disk.
- virtual void writeAssembly (const IDocumentAssemblyPtr &assembly, const String &pathToFile)=0

  Write the given document assembly to a file on disk, specified by a wide character string.
- virtual void writeAssembly (const IDocumentAssemblyPtr &assembly, const IOutputStreamPtr &stream)=0

  Write the given document assembly to a stream.
- virtual IOutputWriterPtr openWriter (const IDocumentAssemblyPtr &assembly, const U8String &pathToFile)=0

  Create an output writer for the given assembly, targeting a file on disk. This is designed to allow streaming output, or to deal with situations where an operation would require too much memory to hold an entire edited assembly in memory at once.
- virtual IOutputWriterPtr openWriter (const IDocumentAssemblyPtr &assembly, const String &pathToFile)=0

  Create an output writer for the given assembly, targeting a file on disk. As above, but with the file specified in a wide character string.
- virtual IOutputWriterPtr openWriter (const IDocumentAssemblyPtr &assembly, const IOutputStreamPtr &stream)=0

  Create an output writer for the given assembly, targeting a stream.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IPDFOutputPtr create (const IJawsMakoPtr &jawsMako, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create a PDF Output instance.

Static Public Member Functions inherited from JawsMako::IOutput

- static JAWSMAKO_API IOutputPtr create (const IJawsMakoPtr &jawsMako, eFileFormat format, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create an output for writing source in the given format.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.379.1 Detailed Description

Interface for the PDF IOutput class.

Supported presets (use IOutput::setPreset) are:

- "Preserve" which will attempt to produce output as close to the input as possible for the output format.
- "PDF1.3" which will configure the output for PDF 1.3. In addition to setting the output version, this will also cause all objects to be colour converted to DeviceRGB, but this may be overridden with setTargetSpace().
- "General" - a general purpose PDF output for general document exchange and viewing. PDF 1.7, RGB.
- "Print" - a higher quality/larger file size for print purposes. Uses PDF 1.4 for wide support.
- "Web" - a general purpose PDF output for distributing documents online. PDF 1.7, RGB, Linearized.
- "PDF/A-1b" - A preset for generating PDF/A-1b, RGB, with sensible defaults for resolution and downsampling.
- "PDF/A-2b" - A preset for generating PDF/A-2b, RGB output intent, with sensible defaults for resolution and downsampling.
- "PDF/A-2u" - A preset for generating PDF/A-2u, RGB output intent, with sensible defaults for resolution and downsampling.
- "PDF/A-3b" - A preset for generating PDF/A-3b, RGB output intent, with sensible defaults for resolution and downsampling.
- "PDF/UA" - A preset for generating PDF/UA, RGB output intent, with sensible defaults for resolution and downsampling. Note that this does not guarantee a valid PDF/UA document as a document is valid if and only if the content is tagged correctly (see IPDFOutput::setVersion() below).
- "PDF/X-1a" - A preset for generating PDF/X-1a, CMYK (SWOP) with everything converted to CMYK.
- "PDF/X-4" - A preset for generating PDF/X-4, CMYK (SWOP) with sensible defaults.

Note that when writing PDF output with multiple documents, only the forms and other top level metadata from the first document is used. Multiple documents should be combined before writing an assembly if required.

Note that there is a limit on the number of PDF and PS output operations depending on the amount of available memory. These are:

- 32 bit platforms - 8 PDF or PS output operations at a time, subject to available memory
- 64 bit platforms - 96 PDF or PS output operations at a time for most tool chains, subject to available memory
  - VS2015 Static builds are currently limited to 48

7.379.2 Member Enumeration Documentation

eImageCompression

`enum JawsMako::IPDFOutput::eImageCompression`

Enumeration for image compression formats.

Enumerator

eICDCT	Jpeg.
eICRLE	Run length encoding.

ePdfAlternateImageHandling

enum JawsMako::IPDFOutput::ePdfAlternateImageHandling

Flags controlling PDF/X-4 and PDF/A-2 alternate image handling.
Allows control on what the PDF/X writer is allowed to invent, or what inconsistencies are considered safe to ignore.
For PDF/X-4, Alternate images are allowed, but no alternate image is allowed to specify that it is the default for printing (via the DefaultForPrinting key). By default, if an image with DefaultForPrinting set to true is present, then this image will be selected and the other alternate images shall be dropped.
For PDF/A-2b/u, Alternate images are not allowed at all. By default, we will apply the optional content rules for alternate images to select a single image as if the PDF was being printed.
Instead of this default, the following flags allow processing to stop in these situations.
Should an exception need to be raised, an IError with code EDL_ERR_INCOMPATIBLE_PDFX or EDL_ERR↩_INCOMPATIBLE_PDFA (depending on output version) will be thrown, where the error description will contain a short message outlining the problem that was encountered.

Enumerator

ePAIHDefault	Default; see the comment above.
ePAIHFailIfAlternatesFound	Throw an exception if an alternate image is encountered.
ePAIHFailIfDefaultForPrintingFound	Throw an exception if an alternate image marked as default for printing is encountered.

ePdfDeviceNHandling

enum JawsMako::IPDFOutput::ePdfDeviceNHandling

Flags controlling PDF/X-4 and PDF/A-2 DeviceN handling.

Allows some control on what the PDF writer is allowed to invent, or what inconsistencies are considered safe to ignore.

Should an exception need to be raised, an IError with code EDL_ERR_INCOMPATIBLE_PDFX or EDL_ERR↩_INCOMPATIBLE_PDFA (depending on output version) will be thrown, where the error description will contain a short message outlining the problem that was encountered.

Enumerator

ePDNHDefault	Default; make a best effort attempt to produce.
ePDNHFailOnMissingColorantInformation	Throw an exception if a spot colorant has.
ePDNHFailOnInconsistentColorantInformation	Throw an exception if inconsistent spot colorant.
ePDNHFailIfBetterColorantInformationFoundAfter↩FirstUse	Throw an exception if better spot colorant.
ePDNHFailIfDotGainFound	Throw an exception if a dotgain is encountered.
ePDNFailOnMismatchedDeviceNProcessSpace	Throw an exception if a DeviceN Process Property does.
ePDNFailOnNonUtf8ColorantName	Throw an exception if a DeviceN/Separation Colorant

name is not valid UTF-8. Normal behaviour is to attempt to re-encode the names as UTF-8.

ePdfExtendedGraphicsStateHandling

enum JawsMako::IPDFOutput::ePdfExtendedGraphicsStateHandling

Flags controlling PDF/X-4 and PDF/A-2 Extended Graphics State information Allows some control on what the PDF writer is allowed to invent, or what inconsistencies are considered safe to ignore.

Should an exception need to be raised, an IError with code EDL_ERR_INCOMPATIBLE_PDFX or EDL_ERR↩ _INCOMPATIBLE_PDFA (depending on output version) will be thrown, where the error description will contain a short message outlining the problem that was encountered.

Enumerator

ePEGSHDefault	Default; make a best effort attempt to produce.
ePEGSHFailOnTransferFunction	Throw an exception if a non-default transfer function is encountered.

ePdfOptionalContentHandling

enum JawsMako::IPDFOutput::ePdfOptionalContentHandling

Flags controlling PDF/X-4 and PDF/A-2 Optional content.
Allows some control on what the PDF writer is allowed to invent, or what inconsistencies are considered safe to ignore.
Should an exception need to be raised, an IError with code EDL_ERR_INCOMPATIBLE_PDFX or EDL_ERR↩ _INCOMPATIBLE_PDFA (depending on output version) will be thrown, where the error description will contain a short message outlining the problem that was encountered.

Enumerator

ePOCHDefault	Default; make a best effort attempt to produce.
ePOCHFailOnInconsistentConfigurationName	Throw an exception if an optional content configuration.
ePOCHFailOnInvalidOrder	Throw if the Optional Content order information, if present.
ePOCHFailIfAutoStateFound	Throw an exception if an optional content configuration contains.

ePDFVersion

enum JawsMako::IPDFOutput::ePDFVersion

Supported versions.

Enumerator

ePDF1↩ _3	PDF 1.3.
ePDF1↩ _4	PDF 1.4.
ePDF1↩ _5	PDF 1.5.
ePDF1↩ _6	PDF 1.6.

Enumerator

ePDF1↩_7	PDF 1.7.
ePDF2↩_0	PDF 2.0 (EXPERIMENTAL)
ePDFA1b	PDF/A-1b, based on PDF 1.4.
ePDFA2b	PDF/A-2b, based on PDF 1.7.
ePDFA3b	PDF/A-3b, based on PDF 1.7.
ePDFA2u	PDF/A-2u, based on PDF 1.7.
ePDFUA	PDF/UA, based on PDF/A-2u.
ePDFX1a	PDF/X-1a:2003, based on PDF 1.4.
ePDFX4	PDF/X-4, based on PDF 1.6.

7.379.3 Member Function Documentation

create()

```
static JAWSMAKO_API IPDFOutputPtr JawsMako::IPDFOutput::create (
            const IJawsMakoPtr & jawsMako,
            const IProgressMonitorPtr & progressMonitor = IProgressMonitorPtr() )  [static]
```

Create a PDF Output instance.

Parameters

jawsMako	The JawsMako instance.
progressMonitor	The progress monitor which allows aborting an operation or registering a progress callback.

Returns

IPDFOutputPtr The PDF Output instance

overrideMaximumICCVersion()

```
virtual void JawsMako::IPDFOutput::overrideMaximumICCVersion (
            uint32 majorVersion,
            uint32 minorVersion )  [pure virtual]
```

Override the maximum allowed ICC profile version for PDF output.

Different PDF versions allow different ICC profile versions to be embedded. PDF/X-4 for example, by a strict reading of the specification, allows ICC Profiles of version 4.2 or earlier. However, some consumers will allow newer ICC Profiles to be used. This API allows the maximum ICC version to be overridden. Only do this if it is certain that the downstream consumers will be able to handle the setting.

The default value is 0,0 indicating that the maximum version for the current PDF output version will be used.

Equivalent to calling setParameter() with the parameter name OverrideMaximumICCVersionMajor and Override↩MaximumICCVersionMinor with the maximum major and minor version respectively.

```
    @param majorVersion An integer indicating the major version, e.g. 4 (for 4.x)
    @param minorVersion An integer indicating the minor version, e.g. 2 (for x.2)
```

setAllowOptionalContentUpdateDuringWrite()

```
virtual void JawsMako::IPDFOutput::setAllowOptionalContentUpdateDuringWrite (
            bool allow )  [pure virtual]
```

Allow updates to optional content to be made while the output is being written using an IOutputWriter.

If true, IOutputWriter::beginDocument() will *not* take a copy of the IOptionalContent associated with that document; instead it references the IOptionalContent. Additional groups may be added, and other edits may be made to the optional content object during the output operation. For example, it is possible to add optional content groups before a page that may use them. However for correct operation:

- Groups must not be deleted
- Any groups used by a page must be added before writePage() is called
- The IOptionalContent object must have been created before IOutputWriter::beginDocument() is invoked.

If false, normal behaviour occurs; the PDF output takes a copy of the IOptionalContent and any edits made during the output operation are not reflected in the output.

The default is false.

Equivalent to calling setParameter() with the parameter name "AllowOptionalContentUpdateDuringWrite" and the value "true" or "false".

If in any doubt, leave this set to false.

Parameters

allow	The required setting

setAllowRestrictedFonts()

```
virtual void JawsMako::IPDFOutput::setAllowRestrictedFonts (
            bool allowRestrictedFonts )  [pure virtual]
```

Allow restricted fonts to be embedded in the output.

If true, fonts that flag that they should not be embedded will be allowed to be embedded. If false, then such fonts will not be embedded.

Set this to true if you have sufficient permissions to do so.

Note that for PDF/X-4 and PDF/A-2b output, the output will instead fail if this setting is false and a restricted font is encountered.

The default is false.

Equivalent to calling setParameter() with the parameter name "AllowRestrictedFonts" and the value "true" or "false".

setAlternateImageErrorHandling()

```
virtual void JawsMako::IPDFOutput::setAlternateImageErrorHandling (
            uint32 flags )  [pure virtual]
```

Set flags controlling PDF/X-4 and PDF/A-2b alternate image handling See ePdfAlternateImageHandling# for details.

Equivalent to calling setParameter() with the parameter name "AlternateImageErrorHandling" and an unsigned integer string containing the flags value.

setAlwaysEmbedFonts()

```
virtual void JawsMako::IPDFOutput::setAlwaysEmbedFonts (
            CU8StringVect & fontNames )  [pure virtual]
```

Provide a list of name of fonts that should be always embedded, regardless of the settings for setEmbedFonts() or setEmbedBase14Fonts(). This setting is ignored for PDF/A.

Note that if incremental output is used, the content of unedited pages is not affected.

Equivalent to calling setParameter() with the parameter name "AlwaysEmbed" and a string with the fonts delimited with semicolons (backslashes are used as an escape character).

setAutoRotatePages()

```
virtual void JawsMako::IPDFOutput::setAutoRotatePages (
            bool autoRotate )  [pure virtual]
```

Set whether or not pages should be automatically rotated. The default is false.

Note that if incremental output is used, the content of unedited pages is not affected.

If true, pages are automatically rotated based on the prevalent text direction on the page. If incremental output is used, non-modified pages will not be subject to this setting. Equivalent to calling setParameter() with the parameter name "AutoRotatePages" with a boolean string ("true" or "false").

The rotation is not a physical rotation of content, merely the PDF Page's view rotate is set appropriately. So, in PDF viewers all annotations and page content will continue to appear correctly.

setBlockNotdefGlyphs()

```
virtual void JawsMako::IPDFOutput::setBlockNotdefGlyphs (
            bool blockNotdefGlyphs )  [pure virtual]
```

Set whether to block notdef glyphs from PDF output.

If true, notdef glyphs that are not-marking are dropped, and if a marking notdef character is encountered, an error is thrown. Also if true, the setting setEnableTrueTypeNotDef() is ignored.

The default is false.

Equivalent to calling setParameter() with the parameter name "BlockNotdefGlyphs" and the value "true" or "false".

setColorImageMaxResolution()

```
virtual void JawsMako::IPDFOutput::setColorImageMaxResolution (
            float resolution,
            float threshold = 0.0f,
            IDOMImageDownsamplerFilter::eDownsamplingMethod method = IDOMImageDownsamplerFilter↩
::eBicubic )  [pure virtual]
```

Set the desired maximum resolution, threshold and downsampling method for colour images.

Note that if incremental output is used, the content of unedited pages is not affected.

The default behaviour is leave the image resolution unchanged. Equivalent to calling setParameter() with the param names "ColorImageDownsamplingResolution", "ColorImageDownsamplingThreshold" and "ColorImage↩ DownsamplingMethod" with the respective values.

Parameters

resolution	The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default.
threshold	The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution.
method	The method to use when downsampling. The default is bicubic for colour images.

setCompressObjects()

```
virtual void JawsMako::IPDFOutput::setCompressObjects (
            bool compressObjects )  [pure virtual]
```

Set whether or not to compress individual objects.

In addition to the compression of individual objects, setting this to true allows the creation of cross reference streams, which allow the generation of PDF files larger than 9999999999 (∼9.3GB).

This setting is advisory only. In particular:

- If performing incremental update, the cross reference format of the original PDF will be retained.
- If writing to PDF 1.4 or earlier, object compression will not be used at all.

The default is true.

setCompressPages()

```
virtual void JawsMako::IPDFOutput::setCompressPages (
            bool compressPages )  [pure virtual]
```

Set whether or not compression should be applied to page content. The default is true.

Note that if incremental output is used, the content of unedited pages is not affected.

Equivalent to calling setParameter() with the parameter name "CompressPages" with a boolean string ("true" or "false").

setConvertAllColors()

```
virtual void JawsMako::IPDFOutput::setConvertAllColors (
            bool convert )  [pure virtual]
```

Set whether or not all content should be color converted to the target space set by setTargetColorSpace().

The default is false.

This parameter is ignored and assumed true when writing to PDF/A-1b or PDF/X-1a format output.

Note that if incremental output is used, the content of unedited pages is not colour converted.

Equivalent to calling setParameter() with the param name "ConvertAllColors" with a boolean string ("true" or "false").

setConvertGray()

```
virtual void JawsMako::IPDFOutput::setConvertGray (
            bool convert )  [pure virtual]
```

Set whether or not gray colors should be subject to color conversion. That is, colors using DeviceGray, sGray, or single-component ICC colorspaces.

The default is true.

This parameter is ignored unless setConvertAllColors() is true.

For PDF/X output, this parameter only applies to objects using the DeviceGray colour space. All other gray spaces are subject to the setting of setConvertAllColors().

Note that if incremental output is used, the content of unedited pages is not colour converted.

Equivalent to calling setParameter() with the param name "ConvertGray" with a boolean string ("true" or "false").

setDefaultFormCJKLanguage()

```
virtual void JawsMako::IPDFOutput::setDefaultFormCJKLanguage (
            eFormCJKCharacterSet characterSet )  [pure virtual]
```

Set the default CJK character set for forms. Used when generating form appearance streams, and is only used when the character set cannot be determined from the content of the form. The default is Japanese.

Parameters

characterSet	The default CJK character set for forms

setDeviceNErrorHandling()

```
virtual void JawsMako::IPDFOutput::setDeviceNErrorHandling (
            uint32 flags )  [pure virtual]
```

Set flags controlling PDF/X-4 and PDF/A-2 DeviceN handling. See ePdfDeviceNHandling# for details.

Equivalent to calling setParameter() with the parameter name "DeviceNErrorHandling" and an unsigned integer string containing the flags value.

Ignored if the output version is anything other than PDF/A-2b or PDF/X-4.

setDownsampleMaskedImages()

```
virtual void JawsMako::IPDFOutput::setDownsampleMaskedImages (
            bool downsampleMaskedImages )  [pure virtual]
```

Set whether or not to downsample masked images.

This applies to cases where an image is masked by a separate masked image, such as types of PDF masked or soft-masked images. These are represented in the DOM using IDOMMaskedBrush, where the sub-brush is an image.

Has no effect if downsampling is not enabled

The default is true.

Equivalent to calling setParameter() with the param names "DownsampleMaskedImages" with the values "true" or "false".

setEmbedBase14Fonts()

```
virtual void JawsMako::IPDFOutput::setEmbedBase14Fonts (
            bool embed )  [pure virtual]
```

Set whether the base 14 PDF fonts should be forcibly embedded. If true, this overrides the embedding settings in IDOMFontOpenType. Only the standard 14 fonts are not affected by this; use setEmbedFonts() to affect other fonts.

Note that if incremental output is used, the content of unedited pages is not affected.

The default is false. This setting is ignored for PDF/A and PDF/X where it is forced to true.

Equivalent to calling setParameter() with the parameter name "EmbedBase14Fonts" with a boolean string ("true" or "false").

setEmbedFonts()

```
virtual void JawsMako::IPDFOutput::setEmbedFonts (
            bool embed )  [pure virtual]
```

Set whether fonts should be forcibly embedded. If true, this overrides the embedding settings in each IDOMFontOpenType. The standard 14 fonts are not affected by this; use setEmbedBaseFonts() instead.

Note that if incremental output is used, the content of unedited pages is not affected.

The default is false. This setting is ignored for PDF/A and PDF/X where it is forced to true.

Equivalent to calling setParameter() with the parameter name "EmbedFonts" with a boolean string ("true" or "false").

setEmit30CmapSubtableForSymbolicTrueTypeFonts()

```
virtual void JawsMako::IPDFOutput::setEmit30CmapSubtableForSymbolicTrueTypeFonts (
            bool emit )  [pure virtual]
```

Set whether to add a 3,0 ("Windows") 'cmap' subtable to symbolic TrueType fonts in PDF output.

If true, a 3,0 cmap subtable will be emitted alongside a 1,0 format subtable for symbolic TrueType fonts. If false, only the 1,0 subtable will be emitted.

This setting is ignored for PDF/A and PDF/X.

The default is true.

Such subtables are required for compatibility with some PDF consumers, and are well tolerated by most PDF consumers. However some preflight tools may emit errors. This can be set false to afford compatibility with such tools.

If unsure, please contact Mako support.

Equivalent to calling setParameter() with the parameter name "Emit30CmapSubtableForSymbolicTrueTypeFonts" and the value "true" or "false".

setEnableIncrementalOutput()

```
virtual void JawsMako::IPDFOutput::setEnableIncrementalOutput (
            bool enable = true )  [pure virtual]
```

Set whether or not incremental output should be enabled.

The default is false (that is, a full save is performed). Incremental output is only possible when all of the following are true:

- writeAssembly() is used (rather than the IOutputWriter interface,
- The assembly has only one document,
- The assembly was opened from a PDF, and
- The PDF was either a file or opened from a random-access stream.
- The PDF version specified is the same version or higher than the input PDF.
- The output is not to be linearized.

If all are true, only modified content is written and appended to the output (actually the source PDF is copied, and then the new content is appended to the copied PDF). This is usually much faster than a full save when the assembly has only been lightly modified. Note that the original PDF will be recoverable and so this should not be used when redacting for example.

Otherwise, a regular full save is performed.

Equivalent to calling setParameter with the parameter name "IncrementalOutput".

For PDF/X and PDF/A, this setting is ignored.

setEnableTrueTypeNotDef()

```
virtual void JawsMako::IPDFOutput::setEnableTrueTypeNotDef (
            bool enableTrueTypeNotDef )  [pure virtual]
```

Enable the use of a True Type font's /.notdef glyph in the output.

If true, a character that does not have a glyph or that has been linked to the /.notdef glyph will be replaced by the /.notdef glyph.

If false, a character that does not have a glyph or that has been linked to the /.notdef glyph will be ignored.

Note that for PDF/X-4 and PDF/A-2b output, the flag is set to true anyway. This flag is also ignored if setBlockNotdefGlyphs() is set true.

The default is true.

Equivalent to calling setParameter() with the parameter name "EnableTrueTypeNotDef" and the value "true" or "false".

setEncryption()

```
virtual void JawsMako::IPDFOutput::setEncryption (
            uint32 keyLength,
            uint32 permissions,
            bool encryptMetadata = false,
            const U8String & ownerPassword = "",
            const U8String & userPassword = "" )  [pure virtual]
```

Set the encryption for the output PDF. The default is no encryption.

Please note:

- Setting a key length of zero disables encryption.
- If encryption is enabled, the encryption algorithm will be chosen based on the output PDF version. Not all key lengths are available for all versions, and an exception will be thrown if an unavailable key length is chosen.
- If encryption is enabled, at least one password must be supplied. If only a user password is supplied, it will also be used for the owner password.
- Permissions are only used if encryption is enabled. That is, permissions information is only stored if the PDF is to be encrypted.
- If 40 bit encryption is used, then encryptMetadata must be true, or an exception will result.
- Some permissions flags are ignored for 40 bit key lengths. For 40 bit output, eFillFormAllowed, eExtraction↩ Allowed, eAssemblyAllowed and eHighQualityPrintAllowed are ignored.
- If encryption is enabled, incremental output is disabled.

Equivalent to calling setParameter with the parameter names "KeyLength", "Permissions", "EncryptMetadata", "↩ OwnerPassword" and "UserPassword".

For PDF/A and PDF/X, encryption is not allowed and an exception will be thrown if output is attempted with Encryption enabled.

Parameters

keyLength	The key length to use. Must be 0, 40, 128 or 256. 0 indicates that encryption will not be performed.
permissions	The permissions bit mask for the output PDF. (see IDOMStandardPDFSecurityInfo::ePermissionsFlags).
encryptMetadata	Whether or not to encrypt metadata. The default is false. Must be set to true if a 40 bit key length is specified.
ownerPassword	The owner password to use.
userPassword	The user password to use.

setExtendedGraphicsStateErrorHandling()

```
virtual void JawsMako::IPDFOutput::setExtendedGraphicsStateErrorHandling (
            uint32 flags )  [pure virtual]
```

Set flags controlling PDF/X-4 and PDF/A-2 Extended Graphics State handling. See ePdfExtendedGraphicsStateHandling# for details.

Equivalent to calling setParameter() with the parameter name "ExtendedGraphicsStateErrorHandling" and an unsigned integer string containing the flags value.

setForceEmitPageGroup()

```
virtual void JawsMako::IPDFOutput::setForceEmitPageGroup (
            bool force )  [pure virtual]
```

Set whether or not the page group should be forcibly written.

If false, the page group will only be written if the PDF output module detects transparent content on the page. Set this to true to emit the page group (if present) regardless of whether the page contains transparency or not.

The default, is true.

This setting is ignored when writing PDF versions that do not support transparency, where the page group will never be written.

Equivalent to calling setParameter() with the parameter name "ForceEmitPageGroup" and the value "true" or "false".

setForceMediaBoxOriginZero()

```
virtual void JawsMako::IPDFOutput::setForceMediaBoxOriginZero (
            bool force )  [pure virtual]
```

Set whether or not the media box origin should be set to 0,0 in the output.

Only applies to pages whose content needs to be written. That is, if incremental output is used, this setting only applies to pages that are edited, or new pages.

The default is false.

Equivalent to calling setParameter() with the parameter name "ForceMediaBoxOriginZero" and the value "true" or "false".

setGrayImageMaxResolution()

```
virtual void JawsMako::IPDFOutput::setGrayImageMaxResolution (
            float resolution,
            float threshold = 0.0f,
            IDOMImageDownsamplerFilter::eDownsamplingMethod method = IDOMImageDownsamplerFilter↩
::eBicubic )  [pure virtual]
```

Set the desired maximum resolution, threshold and downsampling method for gray images.

Note that if incremental output is used, the content of unedited pages is not affected.

The default behaviour is leave the image resolution unchanged. Equivalent to calling setParameter() with the param names "GrayImageDownsamplingResolution", "GrayImageDownsamplingThreshold" and "GrayImage↩ DownsamplingMethod" with the respective values.

Parameters

resolution	The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default.
threshold	The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution.
method	The method to use when downsampling. The default is bicubic for gray images.

setJPEGChromaSubsampling()

```
virtual void JawsMako::IPDFOutput::setJPEGChromaSubsampling (
            bool subsample )  [pure virtual]
```

Set whether or not to use chroma subsampling when compressing color images in DCT/JPEG format. Chroma subsampling improves compression by reducing the resolution of color information in the image, leaving luminance information at high resolution.

For natural images, this results in smaller output file size, but usually with very little perceived quality difference.

However, for non-natural images, which usually compress poorly using DCT/JPEG, chroma subsampling can result in worse perceived quality in areas where color changes rapidly, such as around text.

Note that using setJPEGQuality() will override this setting. Be sure to set the JPEG quality first before setting this parameter.

Equivalent to calling setParameter() with the parameter name "JPEGChromaSubsampling" and the value being "true" or "false".

As the default JPEG quality is 5, the default chroma subsampling is false.

Parameters

subsample	Whether or not to apply chroma subsampling when compressing a color image using DCT.

setJPEGQuality()

```
virtual void JawsMako::IPDFOutput::setJPEGQuality (
```

```
            uint8 quality )  [pure virtual]
```

Set the JPEG quality to use when compressing images in DCT format. Equivalent to calling setParameter() with the parameter name "JPEGQuality" and the value being the required quality.

The default is 5 - highest quality.

For quality levels 4 and 5, chroma subsampling is turned off. For quality levels 1 through 3, chroma subsampling is turned on. However this can be overridden by subsequently using setJPEGChromaSubsampling() to explicitly control chroma subsampling.

Parameters

quality	The desired quality level, with 1 being lowest quality and 5 being highest quality.

setLinearize()

```
virtual void JawsMako::IPDFOutput::setLinearize (
            bool linearize = true )  [pure virtual]
```

Set whether or not the output should be linearized.

The default is false. When true a PDF optimised for byte serving (for example from a web server) will be produced. This will generally be a slow operation. If linearized, a PDF cannot be written incrementally and the setting of setEnableIncrementalOutput() will be ignored.

Equivalent to calling setParameter with the parameter name "Linearize".

setMonoImageMaxResolution()

```
virtual void JawsMako::IPDFOutput::setMonoImageMaxResolution (
            float resolution,
            float threshold = 0.0f,
            IDOMImageDownsamplerFilter::eDownsamplingMethod method = IDOMImageDownsamplerFilter↩
::eSubsample )  [pure virtual]
```

Set the desired maximum resolution, threshold and downsampling method for monochrome images.

Note that if incremental output is used, the content of unedited pages is not affected.

The default behaviour is leave the image resolution unchanged. Equivalent to calling setParameter() with the param names "MonoImageDownsamplingResolution", "MonoImageDownsamplingThreshold" and "MonoImage↩ DownsamplingMethod" with the respective values.

Parameters

resolution	The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default.
threshold	The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution.
method	The method to use when downsampling. The default is subsample for monochrome images; using any other method will result in grayscale output.

setNeverEmbedFonts()

```
virtual void JawsMako::IPDFOutput::setNeverEmbedFonts (
            CU8StringVect & fontNames )  [pure virtual]
```

Provide a list of name of fonts that should never be embedded, regardless of the settings for setEmbedFonts() or setEmbedBase14Fonts(). This setting is ignored for PDF/A.

Equivalent to calling setParameter() with the parameter name "NeverEmbed" and a string with the fonts delimited with semicolons (backslashes are used as an escape character).

setNonUtf8InkNameFallbackEncoding()

```
virtual void JawsMako::IPDFOutput::setNonUtf8InkNameFallbackEncoding (
            eInkFallbackEncoding fallbackEncoding )  [pure virtual]
```

Set the fallback encoding to use when attempting to decode non-UTF8 compatible ink names for PDF/A-2 and PDF/X-4 output. Only consulted when ePDNFailOnNonUtf8ColorantName is unset in setDeviceNErrorHandling. See the two-parameter method of inkNameToString for how this is done.

Equivalent to calling setParameter() with the parameter name "NonUtf8InkNameFallbackEncoding" and the eInk↩ FallbackEncoding enumeration name, with the "eIFE" prefix removed. For example: setParameter("NonUtf8Ink↩ NameFallbackEncoding", "ShiftJIS");

setOptionalContentErrorHandling()

```
virtual void JawsMako::IPDFOutput::setOptionalContentErrorHandling (
            uint32 flags )  [pure virtual]
```

Set flags controlling PDF/X-4 and PDF/A-2 Optional Content handling. See ePdfOptionalContentHandling# for details.

Equivalent to calling setParameter() with the parameter name "OptionalContentErrorHandling" and an unsigned integer string containing the flags value.

setOutputIntent()

```
virtual void JawsMako::IPDFOutput::setOutputIntent (
            const IOutputIntentPtr & outputIntent )  [pure virtual]
```

Set an explicit output intent, or NULL to clear it.

Note: This is only currently honoured for PDF/X-4 and PDF/A-2b output. Please note that as a result of this, PDF 2.0 features are ignored.

Note: This is only currently honoured for the following output versions:

- PDF/X-4
- PDF/A-2b
- PDF/A-2u
- PDF/A-3b
- PDF/UA Please note that as a result of this, PDF 2.0 features are ignored.

For PDF/A-2b and up, setting a NULL output intent causes no output intent to be emitted.

setOutputIntents()

```
virtual void JawsMako::IPDFOutput::setOutputIntents (
            const COutputIntentVect & outputIntents )  [pure virtual]
```

Set a vector of output intents, or an empty vector to clear them.

Note: This is only currently honoured for the following output versions:

- PDF/X-4
- PDF/A-2b
- PDF/A-2u
- PDF/A-3b
- PDF/UA Please note that as a result of this, PDF 2.0 features are ignored.

There are restrictions placed on the content of these output intents:

- If writing to PDF/X-4, there must be one (and only one) output intent with the Subtype of GTS_PDFX. For Mako, this output intent must specify a profile.
- If writing to PDF/A (or variants), there must be one (and only one) output intent with the Subtype GTS_↩ PDFA1. For Mako, this output intent must specify a profile. Further, any output intent that provides a profile must point to the same profile.

The intent associated with the output format will be used to direct the process of conforming the content to the required PDF standard.

For PDF/A-2 and up, setting an empty vector causes no output intent to be emitted.

For best compatibility with downstream consumers, set the first intent to be the primary intent for the output format being targeted. For example, for PDF/X, the intent with GTS_PDFX should be the first intent.

setPreferredColorImageCompression()

```
virtual void JawsMako::IPDFOutput::setPreferredColorImageCompression (
            eImageCompression compression )  [pure virtual]
```

Set the desired image compression for color images that need to be reencoded. The default is eICAuto.

CCITT is not allowed for colour images.

Note: this is advisory only and may not be honoured in all cases.

Note also that if incremental output is used, the content of unedited pages is not affected. For example, PDF/A does not allow LZW.

Equivalent to calling setParameter() with the parameter name "ColorImageCompression" with appropriate values.

Parameters

compression	The desired compression.

setPreferredGrayImageCompression()

```
virtual void JawsMako::IPDFOutput::setPreferredGrayImageCompression (
            eImageCompression compression )  [pure virtual]
```

Set the desired image compression for gray images that need to be reencoded. The default is eICAuto.

CCITT is not allowed for gray images.

Note: this is advisory only and may not be honoured in all cases. For example, PDF/A does not allow LZW.

Note also that if incremental output is used, the content of unedited pages is not affected.

Equivalent to calling setParameter() with the parameter name "GrayImageCompression" with appropriate values (please refer to documentation).

Parameters

compression	The desired compression.

setPreferredMonoImageCompression()

```
virtual void JawsMako::IPDFOutput::setPreferredMonoImageCompression (
            eImageCompression compression )  [pure virtual]
```

Set the desired image compression for monochrome images that need to be reencoded. The default is eICCCITT. JPEG/DCT is not allowed for mono images, and auto will be set to CCITT.

Note: this is advisory only and may not be honoured in all cases. For example, PDF/A does not allow LZW.

Note also that if incremental output is used, the content of unedited pages is not affected.

Equivalent to calling setParameter() with the parameter name "MonoImageCompression" with appropriate values (please refer to documentation).

Parameters

compression	The desired compression.

setPreferredRenderedImageCompression()

```
virtual void JawsMako::IPDFOutput::setPreferredRenderedImageCompression (
            eImageCompression compression )  [pure virtual]
```

Set the desired image compression for images that are the result of rendering. The default is eICFlate. If set to auto, flate will be used.

Note: this is advisory only and may not be honoured in all cases. For example, PDF/A does not allow LZW.

Note also that if incremental output is used, the content of unedited pages is not affected.

Equivalent to calling setParameter() with the parameter name "RenderedImageCompression" with appropriate values (please refer to documentation).

Parameters

compression	The desired compression.

setProducer()

```
virtual void JawsMako::IPDFOutput::setProducer (
            const U8String & producer )  [pure virtual]
```

Set the Producer name for the output PDF.

Equivalent to calling setParameter with the parameter name "Producer" with the value as a string.

The default is "Mako VX.X.X"

Note that this will also be used for the PDF's Creator if the metadata for the document being written does not specify a Creator.

If an empty string is provided, the default will be used.

setPublicKeyEncryption()

```
virtual void JawsMako::IPDFOutput::setPublicKeyEncryption (
            uint32 keyLength,
            const CEDLVector< CPDFRecipientsInfo > & recipients,
            bool encryptMetadata = false )  [pure virtual]
```

Use certificate/public-key encryption for the output PDF. The default is no encryption.

Please note:

- Only AES based encryption is supported.
- If encryption is enabled, incremental output is disabled.

Note: Public key encryption is only supported on Linux and Windows (non-UWP) platforms at this time. Linux platforms will make use of AES encryption for the CMS objects embedded in the PDF, while on Windows Triple DES (3DES) is used. The body of the PDF document however is always encrypted using AES.

For PDF/A and PDF/X, encryption is not allowed and an exception will be thrown if output is attempted with Encryption enabled.

Public key encryption requires at least PDF 1.5. Take care if producing PDF prior to PDF 1.7 using 256 bit encryption.

Parameters

keyLength	The key length to use. Must be 0, 128 or 256. 0 indicates that encryption will not be performed.
recipients	A vector of CPDFRecipients objects for all users who will be allowed to access the resulting document. Pass an empty array to disable encryption.
encryptMetadata	Whether or not to encrypt metadata. The default is false.

setReencodeImages()

```
virtual void JawsMako::IPDFOutput::setReencodeImages (
            bool reencode )  [pure virtual]
```

Set whether images should be reencoded.

PDF Output will attempt to embed images in the output PDF as is, if possible. To force recompression, set to true.

The default is false.

Equivalent to calling setParameter() with the parameter name "ReencodeImages" with "true" or "false".

Note that if incremental output is used, the content of unedited pages is not affected.

setRenderResolution()

```
virtual void JawsMako::IPDFOutput::setRenderResolution (
            uint32 resolution )  [pure virtual]
```

Set the resolution to use if page content requires rendering in order to be output. The default is 300dpi. This is affected also by the maximum image resolution parameters. Equivalent to calling setParameter() with param name "RenderResolution" with the value as the desired resolution as value.

Parameters

resolution	The desired resolution in dpi.

setRetainClippedContent()

```
virtual void JawsMako::IPDFOutput::setRetainClippedContent (
            bool retainClippedContent )  [pure virtual]
```

Allow clipped out content to be retained in the output.

If false, gross clipping will be performed on any object whose bounds falls completely out of the clipping path, and the object will be dropped from the output.

The default is true.

Equivalent to calling setParameter() with the parameter name "RetainClippedContent" and the value "true" or "false".

Parameters

retainClippedContent	The required setting

setRetainEmbeddedFiles()

```
virtual void JawsMako::IPDFOutput::setRetainEmbeddedFiles (
            bool retainEmbeddedFiles )  [pure virtual]
```

Allow embedded files to be retained in PDF/A and PDF/X-4 output.

Applies only to PDF/A and PDF/X-4. If true, any embedded files will be retained in the output. If false embedded files will be dropped.

The default is false.

Equivalent to calling setParameter() with the parameter name "RetainEmbeddedFiles" and the value "true" or "false".

Parameters

retainEmbeddedFiles	The required setting

setSubsetFonts()

```
virtual void JawsMako::IPDFOutput::setSubsetFonts (
            bool subset )  [pure virtual]
```

Set whether embedded fonts should be subset or not. The default is true.

This is a preference only. Some font types may require subsetting based on context.

Note that if incremental output is used, the content of unedited pages is not affected.

Equivalent to calling setParameter() with the parameter name "SubsetFonts" with a boolean string ("true" or "false").

setTargetColorSpace()

```
virtual void JawsMako::IPDFOutput::setTargetColorSpace (
            const IDOMColorSpacePtr & targetSpace )  [pure virtual]
```

Set the target color space for the output.

The default is DeviceRGB.

By default, the target color space is used if anything must be rendered or color-converted. The default behaviour is to apply this only when required. Use convertAllColors() if converting everything is required.

Equivalent to calling setParameter with the param name "TargetColorSpace" with appropriate values (please refer to documentation).

For PDF/A-1b output, an OutputIntents dictionary will be created for this color space. LAB color spaces may not be used as a destination color space for PDF/A.

For PDF/X-1a output, only DeviceCMYK is supported; this setting is ignored. If you wish to change the profile to be used, please set the DeviceCMYK intercept in the color manager to your desired profile (see IColorManager::setDeviceCMYKIntercept()). As for PDF/A-1b an OutputIntents dictionary will be created.

For PDF/X-4 and PDF/A-2b output, an OutputIntents dictionary will be created for this color space. LAB colour spaces may not be used as a destination color space for PDF/X-4 or PDF/A-2. Please note that content using a device space with the same number of components as the output intent will be assumed to be compatible with this space. For example, if a CMYK ICCBased colour space is used for the target space, then any DeviceCMYK content will remain as DeviceCMYK in the output even if the current DeviceCMYK intercept is set to some other profile. Note also that if the output intent is a CMYK color space, then DeviceGray similarly will also be considered equivalent to the K channel of the output intent.

Additionally for PDF/A-2b output, it is possible to set a NULL output intent, in which case no OutputIntent will be produced.

Parameters

targetSpace	The desired color space. Must be a simple or ICC space. scRGB is not supported for output.

setTargetProfile()

```
virtual void JawsMako::IPDFOutput::setTargetProfile (
            const IDOMICCProfilePtr & profile )  [pure virtual]
```

Set the target color space for the output using an ICC profile.

By default, the target color space is used if anything must be rendered or color-converted. The default behaviour is to apply this only when required. Use convertAllColors() if converting everything is required.

Equivalent to calling setParameter() with the param name "TargetProfile" with the value as the path to the profile.

For PDF/A-1b, PDF/X-1a and PDF/X-4 output, the restrictions described in setTargetColorSpace() above apply.

Parameters

profile	The desired profile.

setUseMaskResolutionForMaskedImages()

```
virtual void JawsMako::IPDFOutput::setUseMaskResolutionForMaskedImages (
            bool useMaskResolutionSettingForMaskedImages )  [pure virtual]
```

Set whether or not the mask resolution setting should be applied to the image portion of a masked image.

This applies to cases where an image is masked by a separate masked image, such as types of PDF masked or soft-masked images. These are represented in the DOM using IDOMMaskedBrush, where the sub-brush is an image.

If false, then the mask and the image are evaluated separately and a downsampling resolution and threshold are chosen. For the mask, this is normally either grayscale or monochrome. The image data can be anything. In this mode it is possible for the downsampled image and mask to be downsampled to different resolutions.

If true, then whatever target resolution and threshold is applied to the mask will also be applied to the image samples. If these images have the same effective resolution before downsampling, then they will also share the same effective resolution after downsampling.

Has no effect if downsampling is not enabled

The default is true.

Equivalent to calling setParameter() with the param names "UseMaskResolutionForMaskedImages" with the values "true" or "false".

setValidateXmp()

```
virtual void JawsMako::IPDFOutput::setValidateXmp (
            bool validate )  [pure virtual]
```

Validate XMP metadata.

If true, any XMP properties that do not use the predefined schemas defined in the XMP Specification will be dropped, or if false will be written to the output.

The default is true.

Note that using non-predefined schemas may produce non-conformant output when writing to PDF/X or PDF/A, and it is advised to leave this set to true when writing to PDF/X or PDF/A.

Equivalent to calling setParameter() with the parameter name "ValidateXmp" and the value "true" or "false".

```
    @param validate Validate the XMP.
```

setVersion()

```
virtual void JawsMako::IPDFOutput::setVersion (
            ePDFVersion version )  [pure virtual]
```

Set the PDF version to generate.

Valid versions supported by this release are:

- 1.3
- 1.4
- 1.5
- 1.6
- 1.7
- PDF/A-1b
- PDF/A-2b

- PDF/A-2u
- PDF/A-3b
- PDF/UA
- PDF/X-1a
- PDF/X-2b
- PDF/X-4

Equivalent to calling setParameter with the parameter name "PDFVersion" with the value of the version as a string (i.e. "1.3", "1.7", "PDF/A-1b", "PDF/X-1a"). The default is 1.7. Some features in the content being written may be dropped.

For PDF 1.3 it is recommended that the "PDF1.3" preset be used. This will ensure correct-looking results on diverse consumers with different color management schemes, and will also set the version to 1.3.

Note that for PDF/UA, setting this version only writes the necessary conformance information to the XMP metadata and applies the usual PDF/A-2U processing. This will not automatically result in a conformant or useful PDF. In order for the document to be useful and conformant, all content in the PDF must be appropriately tagged. See the pdfuacreation simple example for an example of how this may be done.

Parameters

version	Version string

The documentation for this class was generated from the following file:

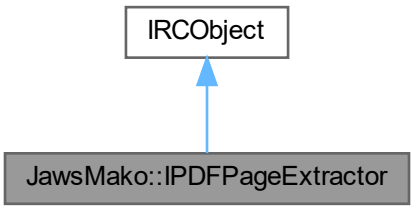
- pdfoutput.h

7.380 JawsMako::IPDFPageExtractor Class Reference

An instance of the JawsMako PDFPageExtractor class.

`#include <pdfpage.h>`

Inheritance diagram for JawsMako::IPDFPageExtractor:



Public Member Functions

- virtual uint32 getNumPages ()=0

  Return the number of pages in the document.
- virtual void extract (const IOutputStreamPtr &outputStream, uint32 pageIndex, uint32 numPages)=0

  Extract a range of pages into a given output stream.
- virtual void extractFrom (const IOutputStreamPtr &outputStream, uint32 pageIndex)=0

  Extract a range of pages from a given index into a given output stream.
- virtual void extractTo (const IOutputStreamPtr &outputStream, uint32 pageIndex)=0

  Extract a range of pages to a given index into a given output stream.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IPDFPageExtractorPtr create (const IJawsMakoPtr &jawsMako, const IInputStream↩ Ptr &inputStream, const U8String &password=U8String(), uint32 permissions=0, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create an IPDFPageExtractor interface.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.380.1 Detailed Description

An instance of the JawsMako PDFPageExtractor class.

7.380.2 Member Function Documentation

create()

```
static JAWSMAKO_API IPDFPageExtractorPtr JawsMako::IPDFPageExtractor::create (
            const IJawsMakoPtr & jawsMako,
            const IInputStreamPtr & inputStream,
            const U8String & password = U8String(),
            uint32 permissions = 0,
            const IProgressMonitorPtr & progressMonitor = IProgressMonitorPtr() )  [static]
```

Create an IPDFPageExtractor interface.

The IPDFPageExtractor interface allows extraction of pages from a PDF.

Parameters

jawsMako	The IJawsMako object.
inputStream	The input PDF stream.
password	The password to use. Assumed to be UTF-8.
permissions	The allowed permissions.

If allowed permissions is set to IDOMStandardPDFSecurityInfo::eEverythingAllowed then no checking for assembly permissions will be performed.

Otherwise, the parameter is the or'd combination of IDOMStandardPDFSecurity::ePermissionsFlags that is checked against the permissions defined in the input PDF. If an operation corresponds to a flag set in the parameter, then output will be allowed to continue. Otherwise, an IError exception with the error code JM_ERR_ASSEMBLY_↩ WRITE_FORBIDDEN will be thrown.

If the source has no permissions information the output will proceed regardless.

The default is 0.

Parameters

progressMonitor	A smart pointer to an IProgressMonitor object which can be NULL if no such object was passed in.

Returns

IPDFPageExtractorPtr A smart pointer to the IPDFPageExtractor object.

extract()

```
virtual void JawsMako::IPDFPageExtractor::extract (
            const IOutputStreamPtr & outputStream,
            uint32 pageIndex,
            uint32 numPages )  [pure virtual]
```

Extract a range of pages into a given output stream.

Pages are numbered starting at zero. An exception will be thrown if an invalid page range is specified.

Parameters

outputStream	The output stream.
pageIndex	The index of the first page to extract.
numPages	The number of pages to extract.

extractFrom()

```
virtual void JawsMako::IPDFPageExtractor::extractFrom (
            const IOutputStreamPtr & outputStream,
            uint32 pageIndex )  [pure virtual]
```

Extract a range of pages from a given index into a given output stream.

A convenience function. Equivalent to extract(pageIndex, getNumPages());

Pages are numbered starting at zero. An exception will be thrown if an invalid page range is specified.

Parameters

outputStream	The output stream.
pageIndex	The index of the first page to extract.

extractTo()

```
virtual void JawsMako::IPDFPageExtractor::extractTo (
            const IOutputStreamPtr & outputStream,
            uint32 pageIndex )  [pure virtual]
```

Extract a range of pages to a given index into a given output stream.

A convenience function. Equivalent to extract(0, pageIndex + 1);

Pages are numbered starting at zero. An exception will be thrown if an invalid page range is specified.

Parameters

outputStream	The output stream.
pageIndex	The index of the page to extract to.

getNumPages()

```
virtual uint32 JawsMako::IPDFPageExtractor::getNumPages ( )  [pure virtual]
```

Return the number of pages in the document.

Returns

uint32 The number of pages in the document.

The documentation for this class was generated from the following file:

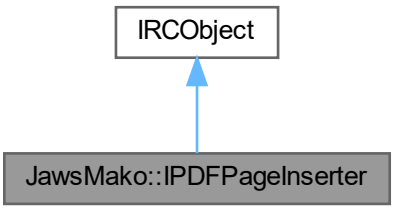
- pdfpage.h

7.381 JawsMako::IPDFPageInserter Class Reference

An instance of the JawsMako PDFPageInserter class.

`#include <pdfpage.h>`

Inheritance diagram for JawsMako::IPDFPageInserter:



Public Member Functions

- virtual uint32 getNumPages ()=0

  Return the number of pages in the document.
- virtual void insert (const IInputStreamPtr &inputStream, uint32 destPageIndex, uint32 sourcePageIndex, uint32 numPages, const U8String &password=U8String(), uint32 permissions=0)=0

  Insert a range of pages from a given input stream.
- virtual void insertFrom (const IInputStreamPtr &inputStream, uint32 destPageIndex, uint32 sourcePageIndex, const U8String &password=U8String(), uint32 permissions=0)=0

  Insert a range of pages from a given input stream.
- virtual void insertTo (const IInputStreamPtr &inputStream, uint32 destPageIndex, uint32 sourcePageIndex, const U8String &password=U8String(), uint32 permissions=0)=0

  Insert a range of pages from a given input stream.
- virtual void save (const IOutputStreamPtr &outputStream)=0

  Save into a given output stream.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IPDFPageInserterPtr create (const IJawsMakoPtr &jawsMako, const IInputStream↩ Ptr &inputStream, const U8String &password=U8String(), uint32 permissions=0, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create an IPDFPageInserter interface.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()
  Virtual destructor.

7.381.1 Detailed Description

An instance of the JawsMako PDFPageInserter class.

7.381.2 Member Function Documentation

create()

```
static JAWSMAKO_API IPDFPageInserterPtr JawsMako::IPDFPageInserter::create (
            const IJawsMakoPtr & jawsMako,
            const IInputStreamPtr & inputStream,
            const U8String & password = U8String(),
            uint32 permissions = 0,
            const IProgressMonitorPtr & progressMonitor = IProgressMonitorPtr() )  [static]
```

Create an IPDFPageInserter interface.

The IPDFPageInserter interface allows insertion of pages into a PDF.

Parameters

jawsMako	The IJawsMako object.
inputStream	The input PDF stream.
password	The password to use. Assumed to be UTF-8.
permissions	The allowed permissions.

If allowed permissions is set to IDOMStandardPDFSecurityInfo::eEverythingAllowed then no checking for assembly permissions will be performed.

Otherwise, the parameter is the or'd combination of IDOMStandardPDFSecurity::ePermissionsFlags that is checked against the permissions defined in the input PDF. If an operation corresponds to a flag set in the parameter, then output will be allowed to continue. Otherwise, an IError exception with the error code JM_ERR_ASSEMBLY_↩ WRITE_FORBIDDEN will be thrown.

If the source has no permissions information the output will proceed regardless.

The default is 0.

Parameters

progressMonitor	A smart pointer to an IProgressMonitor object which can be NULL if no such object was passed in.

Returns

IPDFPageInserterPtr A smart pointer to the IPDFPageInserter object.

getNumPages()

```
virtual uint32 JawsMako::IPDFPageInserter::getNumPages ( )  [pure virtual]
```

Return the number of pages in the document.

Returns

uint32 The number of pages in the document.

insert()

```
virtual void JawsMako::IPDFPageInserter::insert (
            const IInputStreamPtr & inputStream,
            uint32 destPageIndex,
            uint32 sourcePageIndex,
            uint32 numPages,
            const U8String & password = U8String(),
            uint32 permissions = 0 )  [pure virtual]
```

Insert a range of pages from a given input stream.

Pages are numbered starting at zero. An exception will be thrown if an invalid index or page range is specified.

Parameters

inputStream	The input stream.
destPageIndex	The page index of the destination to insert the pages into.
sourcePageIndex	The page index of the source to insert the pages from.
numPages	The number of pages to insert.
password	The password to use. Assumed to be UTF-8.
permissions	The allowed permissions.

insertFrom()

```
virtual void JawsMako::IPDFPageInserter::insertFrom (
            const IInputStreamPtr & inputStream,
            uint32 destPageIndex,
            uint32 sourcePageIndex,
            const U8String & password = U8String(),
            uint32 permissions = 0 )  [pure virtual]
```

Insert a range of pages from a given input stream.

A convenience function. Equivalent to insert(destPageIndex, sourcePageIndex, sourceNumPages);

Pages are numbered starting at zero. An exception will be thrown if an invalid index is specified.

Parameters

inputStream	The input stream.
destPageIndex	The page index of the destination to insert the pages into.
sourcePageIndex	The page index of the source to insert the pages from.
password	The password to use. Assumed to be UTF-8.
permissions	The allowed permissions.

insertTo()

```
virtual void JawsMako::IPDFPageInserter::insertTo (
            const IInputStreamPtr & inputStream,
            uint32 destPageIndex,
            uint32 sourcePageIndex,
            const U8String & password = U8String(),
            uint32 permissions = 0 )  [pure virtual]
```

Insert a range of pages from a given input stream.

A convenience function. Equivalent to insert(destPageIndex, 0, sourcePageIndex + 1);

Pages are numbered starting at zero. An exception will be thrown if an invalid index is specified.

Parameters

inputStream	The input stream.
destPageIndex	The page index of the destination to insert the pages into.
sourcePageIndex	The final page index of the source to insert the pages from.
password	The password to use. Assumed to be UTF-8.
permissions	The allowed permissions.

save()

```
virtual void JawsMako::IPDFPageInserter::save (
            const IOutputStreamPtr & outputStream )  [pure virtual]
```

Save into a given output stream.

Parameters

outputStream	The output stream.

The documentation for this class was generated from the following file:

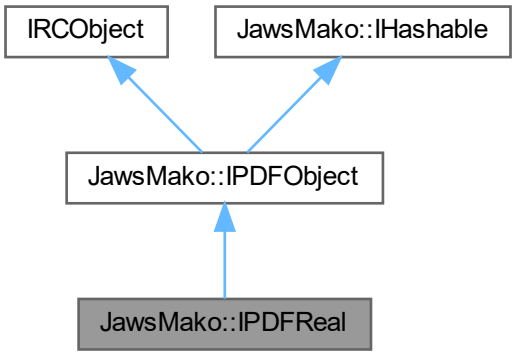
- pdfpage.h

7.382 JawsMako::IPDFReal Class Reference

A simple immutable floating-point PDF object type.

`#include <pdfobjects.h>`

Inheritance diagram for JawsMako::IPDFReal:



Public Member Functions

- virtual double getValue () const =0

  Get the value of the real.

Public Member Functions inherited from JawsMako::IPDFObject

- virtual ePDFObjectType getType () const =0

  Get the type of this PDF object.
- virtual IPDFObjectPtr clone () const =0

  Create a clone of this PDF object if appropriate. Note that simple immutable objects are not actually cloned and instead the same object is returned.
- virtual IPDFObjectPtr deepClone () const =0

  Create a deep clone of this PDF object if appropriate. Note that simple immutable objects are not actually cloned and instead the same object is returned. If the object is composite (an IPDFArray, IPDFDictionary or IPDFStream) then the object will be cloned as well as the consituent objects in a recursive fashion.
- virtual bool getIsExecutable () const =0

  Get whether or not the object is considered executable, such as an executable name, array, or operator.
- virtual bool containsReferences () const =0

  Get whether or not the object contains indirect references to other objects (that is, does the object or any constituent object contain IPDFReference or IPDFFarReference objects).
- virtual void **emitPostScriptCode** (const IOutputStreamPtr &dest) const =0

  Convert the object to PostScript code and write to the given stream. Only allowed for objects that are representable in PostScript.
- virtual bool getNumber (double &number) const =0

  If the object is a number (a real or integer) obtain that number.

- virtual bool getNumber (float &number) const =0

  If the object is a number (a real or integer) within the range of a 32-bit float, obtain that number. If the object is an integer, thiw will fail if the integer cannot be represented exactly in a single precision float.
- virtual bool getNumber (int32 &number) const =0

  If the object is an integer (a integral real or integer) within the range of a 32-bit integer, obtain that number.
- virtual bool getString (RawString &string) const =0

  If the object is a string or a name, retrieve the raw string data.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from JawsMako::IHashable

- virtual uint64 hash () const

  Obtain a 64-bit hash of the receiving object.
- virtual void updateHash (uint64 &hash) const =0

  Update the given hash to include the receiver.

Static Public Member Functions

- static JAWSMAKO_API IPDFRealPtr create (double real)

  Create a real object.

Static Public Member Functions inherited from JawsMako::IPDFObject

- static JAWSMAKO_API IPDFObjectPtr createNumber (double number)

  If the given number is an integer, create an IPDFInteger, otherwise create an IPDFReal.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.382.1 Detailed Description

A simple immutable floating-point PDF object type.

7.382.2 Member Function Documentation

create()

```
static JAWSMAKO_API IPDFRealPtr JawsMako::IPDFReal::create (
            double real )  [static]
```

Create a real object.

Parameters

real	The value.

Returns

IPDFRealPtr The created object.

getValue()

```
virtual double JawsMako::IPDFReal::getValue ( ) const  [pure virtual]
```

Get the value of the real.

Returns

double The value of the real.

The documentation for this class was generated from the following file:

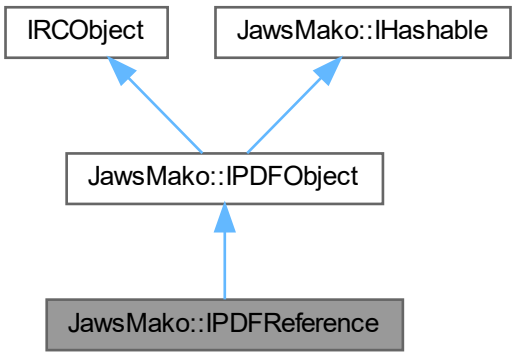
- pdfobjects.h

7.383 JawsMako::IPDFReference Class Reference

A simple class representing an immutable PDF indirect reference.

```
#include <pdfobjects.h>
```

Inheritance diagram for JawsMako::IPDFReference:



Public Member Functions

- virtual const CPDFReference & getValue () const =0

  Obtain the CPDFReference data for this reference.
- int32 getObjectNum ()

  Obtain the object number for this reference.
- int32 getGeneration ()

  Obtain the geneartion number for this reference.

Public Member Functions inherited from JawsMako::IPDFObject

- virtual ePDFObjectType getType () const =0

  Get the type of this PDF object.
- virtual IPDFObjectPtr clone () const =0

  Create a clone of this PDF object if appropriate. Note that simple immutable objects are not actually cloned and instead the same object is returned.
- virtual IPDFObjectPtr deepClone () const =0

  Create a deep clone of this PDF object if appropriate. Note that simple immutable objects are not actually cloned and instead the same object is returned. If the object is composite (an IPDFArray, IPDFDictionary or IPDFStream) then the object will be cloned as well as the consituent objects in a recursive fashion.
- virtual bool getIsExecutable () const =0

  Get whether or not the object is considered executable, such as an executable name, array, or operator.
- virtual bool containsReferences () const =0

  Get whether or not the object contains indirect references to other objects (that is, does the object or any constituent object contain IPDFReference or IPDFFarReference objects).
- virtual void **emitPostScriptCode** (const IOutputStreamPtr &dest) const =0

  Convert the object to PostScript code and write to the given stream. Only allowed for objects that are representable in PostScript.
- virtual bool getNumber (double &number) const =0

  If the object is a number (a real or integer) obtain that number.
- virtual bool getNumber (float &number) const =0

  If the object is a number (a real or integer) within the range of a 32-bit float, obtain that number. If the object is an integer, thiw will fail if the integer cannot be represented exactly in a single precision float.
- virtual bool getNumber (int32 &number) const =0

  If the object is an integer (a integral real or integer) within the range of a 32-bit integer, obtain that number.
- virtual bool getString (RawString &string) const =0

  If the object is a string or a name, retrieve the raw string data.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from JawsMako::IHashable

- virtual uint64 hash () const

  Obtain a 64-bit hash of the receiving object.
- virtual void updateHash (uint64 &hash) const =0

  Update the given hash to include the receiver.

Static Public Member Functions

- static JAWSMAKO_API IPDFReferencePtr create (const CPDFReference &reference)

  Create an indirect reference object from CPDFReference data.
- static JAWSMAKO_API IPDFReferencePtr create (int32 objectNum, int32 generation)

  Create an indirect reference object from an object number and generation.

Static Public Member Functions inherited from JawsMako::IPDFObject

- static JAWSMAKO_API IPDFObjectPtr createNumber (double number)

  If the given number is an integer, create an IPDFInteger, otherwise create an IPDFReal.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.383.1 Detailed Description

A simple class representing an immutable PDF indirect reference.

7.383.2 Member Function Documentation

create() [1/2]

```
static JAWSMAKO_API IPDFReferencePtr JawsMako::IPDFReference::create (
            const CPDFReference & reference )  [static]
```

Create an indirect reference object from CPDFReference data.

Parameters

reference	The CPDFReference data to use.

Returns

IPDFReferencePtr The created object.

create() [2/2]

```
static JAWSMAKO_API IPDFReferencePtr JawsMako::IPDFReference::create (
            int32 objectNum,
            int32 generation )  [inline], [static]
```

Create an indirect reference object from an object number and generation.

Parameters

objectNum	The object number to use.
generation	The generation number to use.

Returns

IPDFReferencePtr The created object.

getGeneration()

```
int32 JawsMako::IPDFReference::getGeneration ( )  [inline]
```

Obtain the geneartion number for this reference.

Returns

int32 The generation.

getObjectNum()

```
int32 JawsMako::IPDFReference::getObjectNum ( )  [inline]
```

Obtain the object number for this reference.

Returns

int32 The object number.

getValue()

```
virtual const CPDFReference & JawsMako::IPDFReference::getValue ( ) const  [pure virtual]
```

Obtain the CPDFReference data for this reference.

Returns

CPDFReference The reference data.

The documentation for this class was generated from the following file:

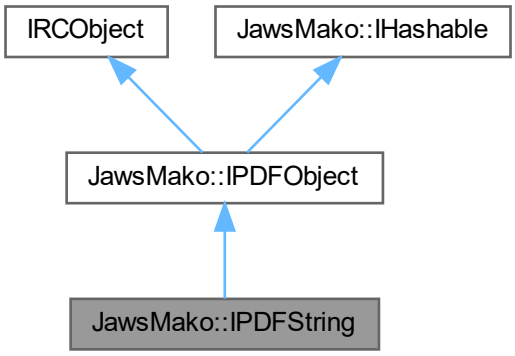
- pdfobjects.h

7.384 JawsMako::IPDFString Class Reference

A simple immutable string pdf object containing raw unencoded data or PDF Text information.

`#include <pdfobjects.h>`

Inheritance diagram for JawsMako::IPDFString:



Public Member Functions

- virtual bool isUtf8Encoded ()=0

  Determine if the string is UTF-8 encoded. Will only be true for strings from PDF 2.0 or later versions.
- virtual const RawString & getValue () const =0

  Get the raw value of the string.
- virtual U8String getTextValue () const =0

  Get the UTF-8 value of this PDF Text-encoded string.

Public Member Functions inherited from JawsMako::IPDFObject

- virtual ePDFObjectType getType () const =0

  Get the type of this PDF object.
- virtual IPDFObjectPtr clone () const =0

  Create a clone of this PDF object if appropriate. Note that simple immutable objects are not actually cloned and instead the same object is returned.
- virtual IPDFObjectPtr deepClone () const =0

  Create a deep clone of this PDF object if appropriate. Note that simple immutable objects are not actually cloned and instead the same object is returned. If the object is composite (an IPDFArray, IPDFDictionary or IPDFStream) then the object will be cloned as well as the consituent objects in a recursive fashion.
- virtual bool getIsExecutable () const =0

  Get whether or not the object is considered executable, such as an executable name, array, or operator.
- virtual bool containsReferences () const =0

  Get whether or not the object contains indirect references to other objects (that is, does the object or any constituent object contain IPDFReference or IPDFFarReference objects).
- virtual void **emitPostScriptCode** (const IOutputStreamPtr &dest) const =0

Convert the object to PostScript code and write to the given stream. Only allowed for objects that are representable in PostScript.

- virtual bool getNumber (double &number) const =0

  If the object is a number (a real or integer) obtain that number.

- virtual bool getNumber (float &number) const =0

  If the object is a number (a real or integer) within the range of a 32-bit float, obtain that number. If the object is an integer, thiw will fail if the integer cannot be represented exactly in a single precision float.

- virtual bool getNumber (int32 &number) const =0

  If the object is an integer (a integral real or integer) within the range of a 32-bit integer, obtain that number.

- virtual bool getString (RawString &string) const =0

  If the object is a string or a name, retrieve the raw string data.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.

- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.

- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from JawsMako::IHashable

- virtual uint64 hash () const

  Obtain a 64-bit hash of the receiving object.

- virtual void updateHash (uint64 &hash) const =0

  Update the given hash to include the receiver.

Static Public Member Functions

- static JAWSMAKO_API IPDFStringPtr create (const RawString &string, bool executable=false, bool pdf2=false)

  Create a string object from a raw unencoded string.

- static JAWSMAKO_API IPDFStringPtr createText (const U8String &string, bool executable=false, bool pdf2=false)

  Create a string object in PDF encoding from a UTF-8 string.

- static JAWSMAKO_API IPDFStringPtr create (const IInputStreamPtr &stream, bool executable=false, bool pdf2=false, int64 startOffset=0)

  Create a raw string object using the contents of a string.

Static Public Member Functions inherited from JawsMako::IPDFObject

- static JAWSMAKO_API IPDFObjectPtr createNumber (double number)

  If the given number is an integer, create an IPDFInteger, otherwise create an IPDFReal.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()
  Virtual destructor.

7.384.1 Detailed Description

A simple immutable string pdf object containing raw unencoded data or PDF Text information.

A simple immutable name pdf object.

7.384.2 Member Function Documentation

create() [1/2]

```
static JAWSMAKO_API IPDFStringPtr JawsMako::IPDFString::create (
            const IInputStreamPtr & stream,
            bool executable = false,
            bool pdf2 = false,
            int64 startOffset = 0 )  [static]
```

Create a raw string object using the contents of a string.

Parameters

stream	The stream to use as source data.
executable	If this string should be marked executable.
pdf2	True if this string was created from a PDF 2.0 or later document.
startOffset	The starting offse to copy from the stream.

Returns

IPDFStringPtr The created object.

create() [2/2]

```
static JAWSMAKO_API IPDFStringPtr JawsMako::IPDFString::create (
            const RawString & string,
            bool executable = false,
            bool pdf2 = false )  [static]
```

Create a string object from a raw unencoded string.

Parameters

string	The raw string value.
executable	If this string should be marked executable.
pdf2	True if this string was created from a PDF 2.0 or later document.

Returns

IPDFStringPtr The created object.

createText()

```
static JAWSMAKO_API IPDFStringPtr JawsMako::IPDFString::createText (
            const U8String & string,
            bool executable = false,
            bool pdf2 = false )  [static]
```

Create a string object in PDF encoding from a UTF-8 string.

Parameters

string	The string value.
executable	If this string should be marked executable.
pdf2	True if this string was created from a PDF 2.0 or later document.

Returns

IPDFStringPtr The created object.

getTextValue()

```
virtual U8String JawsMako::IPDFString::getTextValue ( ) const  [pure virtual]
```

Get the UTF-8 value of this PDF Text-encoded string.

Returns

U8String The UTF-8 value of this string.

getValue()

```
virtual const RawString & JawsMako::IPDFString::getValue ( ) const  [pure virtual]
```

Get the raw value of the string.

Returns

RawString The value of the real.

isUtf8Encoded()

```
virtual bool JawsMako::IPDFString::isUtf8Encoded ( )  [pure virtual]
```

Determine if the string is UTF-8 encoded. Will only be true for strings from PDF 2.0 or later versions.

Returns

bool True if UTF-8 encoded.

The documentation for this class was generated from the following file:

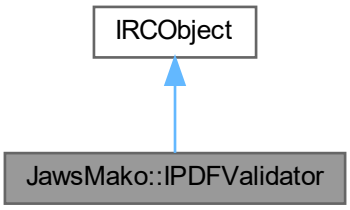
- pdfobjects.h

7.385 JawsMako::IPDFValidator Class Reference

A class for validating PDF documents against published PDF standards such as PDF/X.

```
#include <validate.h>
```

Inheritance diagram for JawsMako::IPDFValidator:



Classes

- class CContentError

  A class describing validation errors present in a particular location on a page.
- class CGeneralError

  A class describing validation errors found in a PDF that are not tied to a location on a page.
- class CPageErrors

  A collection of all the errors associated with a page.

Public Types

- enum ePDFValidateVersion

  Supported validation standards.

Public Member Functions

- virtual bool validate (const IInputStreamPtr &pdfStream, CGeneralErrorVect &globalErrors, CPageErrorsVect &pageErrors)=0

  Perform validation on an entire PDF stream.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IPDFValidatorPtr **create** (const IJawsMakoPtr &jawsMako, ePDFValidateVersion version)

  Create an IPDFValidator instance that validates against the given version.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.385.1 Detailed Description

A class for validating PDF documents against published PDF standards such as PDF/X.

7.385.2 Member Function Documentation

validate()

```
virtual bool JawsMako::IPDFValidator::validate (
            const IInputStreamPtr & pdfStream,
            CGeneralErrorVect & globalErrors,
            CPageErrorsVect & pageErrors )  [pure virtual]
```

Perform validation on an entire PDF stream.

```
Please note that currently not all errors are caught by this validator.
In particular the following will not trigger failure:
- Common errors in PDF content that are worked around by Mako due to
  their prevalence in the field.
- Issues with Halftone dictionaries.
- Correctness of OPI information, including embedded files (PDF/X-1-2001 only)
- Correctness of XMP Metadata information
- Whether or not an output intent exists in a standard registry.
- Internal ICC profile structure.
- Default Transfer functions (TR/TR2) in some rare situations.
- Correctness of TrapNet annotations.
- Contents of embedded files (however embedded files are prohibited in most PDF
  standards).
- Actions in outlines that simply reference a destination (GoTo).
- With the exception of transparency, features that are not present in the target
  version of the PDF specification are not flagged as errors.
```

Parameters

pdfStream	The PDF stream.
globalErrors	A reference to receive the list of global errors (that is, errors not attached to a particular page).
pageErrors	A reference to receive the list of page errors (that is, errors attributable to a given page). The page errors will be a vector where the vector entry 0 is for the first page and so forth.

Returns

bool True if no errors were found, false otherwise.

The documentation for this class was generated from the following file:

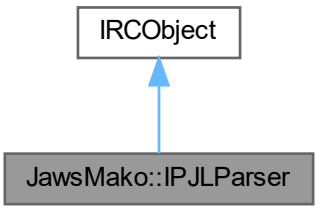
- validate.h

7.386 JawsMako::IPJLParser Class Reference

A PJL (Printer Job Language) parser for Mako.

`#include <pjl.h>`

Inheritance diagram for JawsMako::IPJLParser:



Classes

- class CPjlAttributeValue

  A captured PJL attribute.

Public Types

- enum ePjlResult { ePREnterPclXl , ePREnterPcl , ePREnterPostScript , ePREnterPdf , ePREnterHpgl2 , ePREndOfFile }

  Result code for PJL parsing.
- enum eDuplexBindingMode { eDBMLongEdge , eDBMShortEdge }

  Duplex binding mode enumeration.

Public Member Functions

- virtual IPJLParserPtr **clone** ()=0

  Clone the parser and the PJL environment.
- virtual ePjlResult parse (const IInputPushbackStreamPtr &stream)=0

  Parse from the given stream, until another language is encountered or the end of the stream is reached.
- virtual CPjlAttributeVect getAttributes (const RawString &command, const RawString &key=RawString())=0

  Retrieves all attributes set with the given PJL command and key.
- virtual void **setDefaultPaperSize** (const U8String &paperSize)=0

  Set the default paper size in the PJL Environment. Equivalent to having parsed "@PJL SET PAPER = <papersize>".
- virtual void **setDefaultLandscape** (bool defaultLandscape)=0

  Set the default orientation in the PJL Environment. Equivalent to having parsed "@PJL SET ORIENTATION = <↩ PORTRAIT or LANDSCAPE>".
- virtual void **setDefaultCopies** (uint32 defaultCopies)=0

  Set the default number of copies in the PJL Environment. Equivalent to having parsed "@PJL SET COPIES = <default copies>".
- virtual void **setDefaultDuplex** (bool defaultDuplex)=0

  Set the default duplex in the PJL Environment. Equivalent to having parsed "@PJL SET DUPLEX = <ON or OFF>".
- virtual void **setDefaultDuplexBindingMode** (eDuplexBindingMode defaultDuplexBinding)=0

  Set the default binding mode in the PJL Environment. Equivalent to having parsed "@PJL SET BINDING = <↩ LONGEDGE or SHORTEDGE>".
- virtual void **setDefaultManualFeed** (bool defaultManualFeed)=0

  Set the default manual feed mode in the PJL Environment. Equivalent to having parsed "@PJL SET MANUALFEED = <ON or OFF>".

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IPJLParserPtr create (const IJawsMakoPtr &jawsMako)

  Create a PJL parser instance.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.386.1 Detailed Description

A PJL (Printer Job Language) parser for Mako.

7.386.2 Member Enumeration Documentation

eDuplexBindingMode

```
enum JawsMako::IPJLParser::eDuplexBindingMode
```

Duplex binding mode enumeration.

Enumerator

eDBMLongEdge	Bind on long edge.
eDBMShortEdge	Bind on short edge.

ePjlResult

```
enum JawsMako::IPJLParser::ePjlResult
```

Result code for PJL parsing.

Enumerator

ePREnterPclXl	Parse content that follows as PCL/XL.
ePREnterPcl	Parse content that follows as PCL5.
ePREnterPostScript	Parse content that follows as PostScript.
ePREnterPdf	Parse content that follows as PDF.
ePREnterHpgl2	Parse content that follows as HPGL/2.
ePREndOfFile	An end of file was encountered.

7.386.3 Member Function Documentation

create()

```
static JAWSMAKO_API IPJLParserPtr JawsMako::IPJLParser::create (
            const IJawsMakoPtr & jawsMako )  [static]
```

Create a PJL parser instance.

Returns

IPJLParserPtr The PJL input

getAttributes()

```
virtual CPjlAttributeVect JawsMako::IPJLParser::getAttributes (
            const RawString & command,
            const RawString & key = RawString() )  [pure virtual]
```

Retrieves all attributes set with the given PJL command and key.

If the key is empty, all attributes set with the given command will be returned.

parse()

```
virtual ePjlResult JawsMako::IPJLParser::parse (
            const IInputPushbackStreamPtr & stream )  [pure virtual]
```

Parse from the given stream, until another language is encountered or the end of the stream is reached.

Exceptions of type IError are thrown on error.

Parameters

stream	The stream to parse. This must be a pushback-capable stream, as the parser must be able to sniff ahead. The stream must be open; the parser will neither open nor close the stream.

Returns

ePjlResult The result of parsing.

The documentation for this class was generated from the following file:

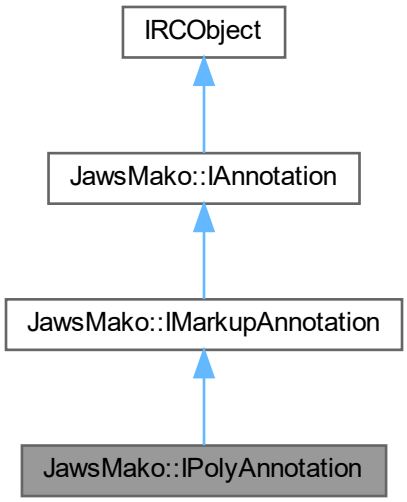
- pjl.h

7.387 JawsMako::IPolyAnnotation Class Reference

An interface class for a polygon or polyline annotation. It is intended that future releases of JawsMako will extend this interface.

```
#include <interactive.h>
```

Inheritance diagram for JawsMako::IPolyAnnotation:



Public Member Functions

- virtual CFPointVect getPoints () const =0

  Get the points comprising the vertices of the polygon. The points are relative to the annotation rect.
- virtual void setPoints (const CFPointVect &points)=0

  Set the points comprising the vertices of the polygon. The points are relative to the annotation rect.

Public Member Functions inherited from JawsMako::IMarkupAnnotation

- virtual U8String getAuthor () const =0

  Get the Author of the markup annotation. This is the "T" entry in the annotation, which by convention is the author.
- virtual void setAuthor (const U8String &author)=0

  Set the Author of the markup annotation. This is the "T" entry in the annotation, which by convention is the author.
- virtual IEDLTimePtr getCreationTime () const =0

  Get the creation date and time of the annotation, if present.
- virtual void setCreationTime (const IEDLTimePtr &creationTime)=0

  Set the creation date and time of the annotation.
- virtual float getOpacity () const =0

  Get the opacity of the markup annotation.
- virtual void setOpacity (float opacity)=0

  Set the opacity of the markup annotation.
- virtual IAnnotationReferencePtr getPopupReference () const =0

  Get a reference to the popup, if present.
- virtual void setPopup (const IAnnotationReferencePtr &popupReference)=0

  Set or clear the popup, by reference.
- virtual void setPopup (const IPopupAnnotationPtr &popup)=0

  Set a reference to a popup, if present.
- virtual IAnnotationAppearancePtr createNormalAppearance () const =0

  Create a basic appearance given the current annotation's parameters. This can then be installed by using IAnnotation::addAppearance().

Public Member Functions inherited from JawsMako::IAnnotation

- virtual eAnnotationType getType () const =0

  Get the type of the annotation.
- virtual const FRect & getRect () const =0

  Get the rect in which the appearances should be displayed.
- virtual void setRect (const FRect &rect)=0

  Set the rect in which the appearances should be displayed.
- virtual U8String getContents () const =0

  Get the Contents entry in UTF-8.
- virtual void setContents (const U8String &contents)=0

  Set the Contents entry in UTF-8.
- virtual IDOMColorPtr getColor () const =0

  Get the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.
- virtual void setColor (const IDOMColorPtr &color)=0

  Set the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.
- virtual IEDLTimePtr getModificationTime () const =0

  Get the Modification date and time of the annotation, if present.
- virtual void setModificationTime (const IEDLTimePtr &modificationTime)=0

Set the Modification date and time of the annotation.

- virtual CAnnotationBorder getBorder () const =0

  Get the annotation's border. See CAnnotationBorder for details.

- virtual void setBorder (const CAnnotationBorder &border)=0

  Set the annotation's border.

- virtual uint32 getFlags () const =0

  Get the annotation flags. To interpret these flags please see section 8.4.2 "Annotation Flags" in the PDF 1.7 specification.

- virtual void setFlags (uint32 flags)=0

  Set the annotation flags. Please see section 8.4.2 "Annotation Flags" for details about these flags.

- virtual void rotate (uint16 rotate, double pageWidth, double pageHeight)=0

  Rotate the annotation clockwise as if the page was rotated by the same amount.

- virtual CAnnotationAppearanceVect getAppearances () const =0

  Return all the annotation appearances in a vector.

- virtual void **removeAppearances** ()=0

  Remove all annotation appearances.

- virtual U8String getState () const =0

  Get the current annotation state.

- virtual void setState (const U8String &state)=0

  Set the current annotation state.

- virtual IAnnotationAppearancePtr getAppearance (eAppearanceUsage usage, const U8String &state=U8String()) const =0

  Fetch the annotation appearance that should be used for the given annotation usage and state according to the following rules:

- virtual void addAppearance (const IAnnotationAppearancePtr &appearance)=0

  Add or replace an appearance.

- virtual bool hasNormalAppearance () const =0

  Does the annotation have a normal appearance? Convenience utility function.

- virtual IAnnotationPtr clone () const =0

  Clone the annotation. This is a deep clone. The annotation reference will remain the same.

- virtual bool matchesReference (const IAnnotationReferencePtr &reference) const =0

  Does this annotation match the given IAnnotationReference?

- virtual IAnnotationReferencePtr getReference () const =0

  Get a reference that can be used to refer to this annotation.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.

- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.

- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Public Types inherited from JawsMako::IAnnotation

- enum eAnnotationType {
  eAT3D , eATCaret , eATCircle , eATFileAttachment ,
  eATFreeText , eATHighlight , eATInk , eATLine ,
  eATLink , eATMovie , eATPolygon , eATPolyLine ,
  eATPopup , eATPrinterMark , eATProjection , eATRedact ,
  eATRichMedia , eATScreen , eATSound , eATSquare ,
  eATSquiggly , eATStamp , eATStrikeOut , eATText ,
  eATTrapNet , eATUnderline , eATWatermark , eATWidget ,
  eATOther }

  Types of annotations, listed with the earliest version of PDF that supported them.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.387.1 Detailed Description

An interface class for a polygon or polyline annotation. It is intended that future releases of JawsMako will extend this interface.

7.387.2 Member Function Documentation

getPoints()

```
virtual CFPointVect JawsMako::IPolyAnnotation::getPoints ( ) const  [pure virtual]
```

Get the points comprising the vertices of the polygon. The points are relative to the annotation rect.

Returns

CFPointVect The vector of points

setPoints()

```
virtual void JawsMako::IPolyAnnotation::setPoints (
            const CFPointVect & points )  [pure virtual]
```

Set the points comprising the vertices of the polygon. The points are relative to the annotation rect.

Parameters

points	The vector of points

The documentation for this class was generated from the following file:

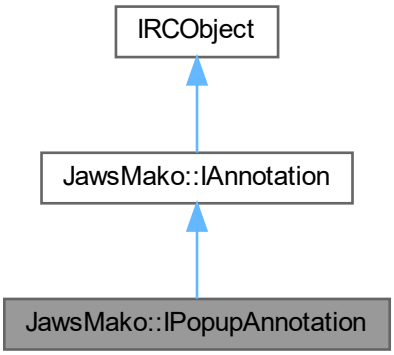
- interactive.h

7.388 JawsMako::IPopupAnnotation Class Reference

An interface class for a popup annotation, which should not exist as a standalone, but is associated with a Markup Annotation. No appearances can be added to this annotation type. It is intended that future releases of JawsMako will extend this interface.

`#include <interactive.h>`

Inheritance diagram for JawsMako::IPopupAnnotation:



Public Member Functions

- virtual bool getOpen () const =0
  Get the annotation's open status.
- virtual void setOpen (bool open)=0
  Set the annotation's open status.

Public Member Functions inherited from JawsMako::IAnnotation

- virtual eAnnotationType getType () const =0
  Get the type of the annotation.
- virtual const FRect & getRect () const =0
  Get the rect in which the appearances should be displayed.
- virtual void setRect (const FRect &rect)=0
  Set the rect in which the appearances should be displayed.
- virtual U8String getContents () const =0
  Get the Contents entry in UTF-8.
- virtual void setContents (const U8String &contents)=0

Set the Contents entry in UTF-8.

- virtual IDOMColorPtr getColor () const =0

  Get the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.
- virtual void setColor (const IDOMColorPtr &color)=0

  Set the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.
- virtual IEDLTimePtr getModificationTime () const =0

  Get the Modification date and time of the annotation, if present.
- virtual void setModificationTime (const IEDLTimePtr &modificationTime)=0

  Set the Modification date and time of the annotation.
- virtual CAnnotationBorder getBorder () const =0

  Get the annotation's border. See CAnnotationBorder for details.
- virtual void setBorder (const CAnnotationBorder &border)=0

  Set the annotation's border.
- virtual uint32 getFlags () const =0

  Get the annotation flags. To interpret these flags please see section 8.4.2 "Annotation Flags" in the PDF 1.7 specification.
- virtual void setFlags (uint32 flags)=0

  Set the annotation flags. Please see section 8.4.2 "Annotation Flags" for details about these flags.
- virtual void rotate (uint16 rotate, double pageWidth, double pageHeight)=0

  Rotate the annotation clockwise as if the page was rotated by the same amount.
- virtual CAnnotationAppearanceVect getAppearances () const =0

  Return all the annotation appearances in a vector.
- virtual void **removeAppearances** ()=0

  Remove all annotation appearances.
- virtual U8String getState () const =0

  Get the current annotation state.
- virtual void setState (const U8String &state)=0

  Set the current annotation state.
- virtual IAnnotationAppearancePtr getAppearance (eAppearanceUsage usage, const U8String &state=U8String()) const =0

  Fetch the annotation appearance that should be used for the given annotation usage and state according to the following rules:
- virtual void addAppearance (const IAnnotationAppearancePtr &appearance)=0

  Add or replace an appearance.
- virtual bool hasNormalAppearance () const =0

  Does the annotation have a normal appearance? Convenience utility function.
- virtual IAnnotationPtr clone () const =0

  Clone the annotation. This is a deep clone. The annotation reference will remain the same.
- virtual bool matchesReference (const IAnnotationReferencePtr &reference) const =0

  Does this annotation match the given IAnnotationReference?
- virtual IAnnotationReferencePtr getReference () const =0

  Get a reference that can be used to refer to this annotation.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IPopupAnnotationPtr create (const IJawsMakoPtr &jawsMako, const FRect &rect, bool open=false)

  Create a popup annotation.

Additional Inherited Members

Public Types inherited from JawsMako::IAnnotation

- enum eAnnotationType {
  eAT3D , eATCaret , eATCircle , eATFileAttachment ,
  eATFreeText , eATHighlight , eATInk , eATLine ,
  eATLink , eATMovie , eATPolygon , eATPolyLine ,
  eATPopup , eATPrinterMark , eATProjection , eATRedact ,
  eATRichMedia , eATScreen , eATSound , eATSquare ,
  eATSquiggly , eATStamp , eATStrikeOut , eATText ,
  eATTrapNet , eATUnderline , eATWatermark , eATWidget ,
  eATOther }

  Types of annotations, listed with the earliest version of PDF that supported them.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.388.1 Detailed Description

An interface class for a popup annotation, which should not exist as a standalone, but is associated with a Markup Annotation. No appearances can be added to this annotation type. It is intended that future releases of JawsMako will extend this interface.

7.388.2 Member Function Documentation

create()

```
static JAWSMAKO_API IPopupAnnotationPtr JawsMako::IPopupAnnotation::create (
            const IJawsMakoPtr & jawsMako,
            const FRect & rect,
            bool open = false )  [static]
```

Create a popup annotation.

Parameters

jawsMako	The JawsMako instance
rect	The annotations bounds. Must not be empty.
open	Optional; whether or not the note is open. Default is false.

Returns

IPopupAnnotationPtr A smart pointer to the new popup annotation

getOpen()

```
virtual bool JawsMako::IPopupAnnotation::getOpen ( ) const  [pure virtual]
```

Get the annotation's open status.

Returns

bool True if open, false if closed

setOpen()

```
virtual void JawsMako::IPopupAnnotation::setOpen (
            bool open )  [pure virtual]
```

Set the annotation's open status.

Parameters

open	Set to true for open, false for closed

The documentation for this class was generated from the following file:

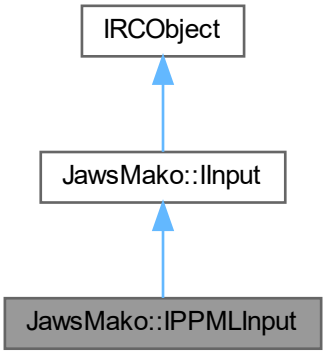
- interactive.h

7.389 JawsMako::IPPMLInput Class Reference

An instance of the JawsMako PPML input class.

```
#include <ppmlinput.h>
```

Inheritance diagram for JawsMako::IPPMLInput:



Static Public Member Functions

- static JAWSMAKO_API IPPMLInputPtr create (const IJawsMakoPtr &jawsMako, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create an input for reading source documents in PPML format.

Static Public Member Functions inherited from JawsMako::IInput

- static JAWSMAKO_API IInputPtr create (const IJawsMakoPtr &jawsMako, eFileFormat format, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create an input for reading source documents in the given format. The following formats are currently supported:

Additional Inherited Members

Public Member Functions inherited from JawsMako::IInput

- virtual IDocumentAssemblyPtr open (const U8String &pathToFile)=0

  Open a file on disk, returning the IDocumentAssembly representing the contents.
- virtual IDocumentAssemblyPtr open (const String &pathToFile)=0

  Open a file on disk, returning the IDocumentAssembly representing the contents. Takes a wide character string.
- virtual IDocumentAssemblyPtr open (const IInputStreamPtr &inputStream)=0

  Open a stream, returning the IDocumentAssembly representing the contents.
- virtual void setSequentialMode (bool sequential)=0

  Set/unset sequential mode on this input.
- virtual void setParameter (const U8String &param, const U8String &value)=0

  Apply a key value pair output parameter with a string value. The parameter name is case insensitive. Please refer to the supplied documentation for the details of the available parameters and their ranges.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.389.1 Detailed Description

An instance of the JawsMako PPML input class.

7.389.2 Member Function Documentation

create()

```
static JAWSMAKO_API IPPMLInputPtr JawsMako::IPPMLInput::create (
            const IJawsMakoPtr & jawsMako,
            const IProgressMonitorPtr & progressMonitor = IProgressMonitorPtr() )  [static]
```

Create an input for reading source documents in PPML format.

Parameters

jawsMako	The IJawsMako object
progressMonitor	A smart pointer to an IProgressMonitor object which can be NULL if no such object was passed in.

Returns

IPPMLInputPtr the PPML input

The documentation for this class was generated from the following file:

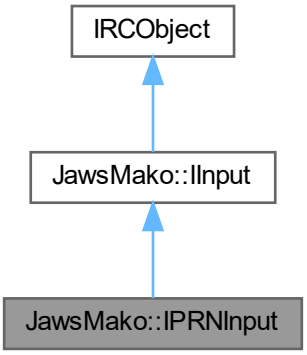
- ppmlinput.h

7.390 JawsMako::IPRNInput Class Reference

An instance of the JawsMako PRN input class.

```
#include <prninput.h>
```

Inheritance diagram for JawsMako::IPRNInput:



Public Member Functions

- virtual void setIgnorePrescribe (bool ignorePrescribe)=0

  Set whether or not to ignore Kyocera PRESCRIBE commands.

Public Member Functions inherited from JawsMako::IInput

- virtual IDocumentAssemblyPtr open (const U8String &pathToFile)=0

  Open a file on disk, returning the IDocumentAssembly representing the contents.
- virtual IDocumentAssemblyPtr open (const String &pathToFile)=0

  Open a file on disk, returning the IDocumentAssembly representing the contents. Takes a wide character string.
- virtual IDocumentAssemblyPtr open (const IInputStreamPtr &inputStream)=0

  Open a stream, returning the IDocumentAssembly representing the contents.
- virtual void setSequentialMode (bool sequential)=0

  Set/unset sequential mode on this input.
- virtual void setParameter (const U8String &param, const U8String &value)=0

  Apply a key value pair output parameter with a string value. The parameter name is case insensitive. Please refer to the supplied documentation for the details of the available parameters and their ranges.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IPRNInputPtr create (const IJawsMakoPtr &jawsMako)

  Create an input for reading printer (PRN) source documents.

Static Public Member Functions inherited from JawsMako::IInput

- static JAWSMAKO_API IInputPtr create (const IJawsMakoPtr &jawsMako, eFileFormat format, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create an input for reading source documents in the given format. The following formats are currently supported:

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.390.1 Detailed Description

An instance of the JawsMako PRN input class.

7.390.2 Member Function Documentation

create()

```
static JAWSMAKO_API IPRNInputPtr JawsMako::IPRNInput::create (
            const IJawsMakoPtr & jawsMako )  [static]
```

Create an input for reading printer (PRN) source documents.

The IPRNInput interface allows PRN documents to be imported into JawsMako. These are files that have a PJL header and one or more documents in a given PDL, immediately preceded by a @PJL ENTER LANGUAGE tag. The language tags that are currently recognized are PCL, PCLXL, HPGL2, POSTSCRIPT and PDF.

Parameters

jawsMako	The IJawsMako object

Returns

IPRNInputPtr the PRN input

setIgnorePrescribe()

```
virtual void JawsMako::IPRNInput::setIgnorePrescribe (
            bool ignorePrescribe )  [pure virtual]
```

Set whether or not to ignore Kyocera PRESCRIBE commands.

```
When enabled, an attempt will be made to ignore Kyocera PRESCRIBE
commands in PCL5 streams by skipping data from the start sequence
!R! to the end sequence EXIT;

Note that !R! may appear in PCL5 data as text and so this parameter
should only be enabled when opening PCL5 streams with Kyocera PRESCRIBE.

The default is false.
```

The documentation for this class was generated from the following file:

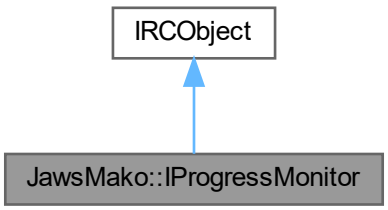
- prninput.h

7.391 JawsMako::IProgressMonitor Class Reference

An abstract class encapsulating both an IProgressTick and an IAbort so that the caller can monitor the progress and/or abort the processing.

`#include <jawsmako.h>`

Inheritance diagram for JawsMako::IProgressMonitor:



Public Member Functions

- virtual IProgressTickPtr getProgressTick () const =0

  Return the IProgressTick object or NULL if there was no such object.
- virtual IAbortPtr getAbort () const =0

  Return the smart pointer to an IAbort object.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IProgressMonitorPtr create (const IProgressTickPtr &progressTick, const IAbortPtr &abort)

  Create smart pointer to an IProgressMonitor object.
- static JAWSMAKO_API IProgressMonitorPtr create (const IProgressTickPtr &progressTick)

  Create smart pointer to an IProgressMonitor object.
- static JAWSMAKO_API IProgressMonitorPtr create (const IAbortPtr &abort)

  Create smart pointer to an IProgressMonitor object.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.391.1 Detailed Description

An abstract class encapsulating both an IProgressTick and an IAbort so that the caller can monitor the progress and/or abort the processing.

7.391.2 Member Function Documentation

create() [1/3]

```
static JAWSMAKO_API IProgressMonitorPtr JawsMako::IProgressMonitor::create (
            const IAbortPtr & abort )  [static]
```

Create smart pointer to an IProgressMonitor object.

Parameters

in	*abort*	A smart pointer to an IAbort object which can be NULL if no object was passed in.

Returns

IProgressMonitorPtr The IProgressMonitorPtr object

create() [2/3]

```
static JAWSMAKO_API IProgressMonitorPtr JawsMako::IProgressMonitor::create (
            const IProgressTickPtr & progressTick )  [static]
```

Create smart pointer to an IProgressMonitor object.

Parameters

in	*progressTick*	A smart pointer to an IProgressTick object which can be NULL if no object was passed in.

Returns

IProgressMonitorPtr The IProgressMonitorPtr object

create() [3/3]

```
static JAWSMAKO_API IProgressMonitorPtr JawsMako::IProgressMonitor::create (
            const IProgressTickPtr & progressTick,
            const IAbortPtr & abort )  [static]
```

Create smart pointer to an IProgressMonitor object.

Parameters

in	*progressTick*	A smart pointer to an IProgressTick object which can be NULL of no object was passed in.
in	*abort*	A smart pointer to an IAbort object which can be NULL of no object was passed in.

Returns

IProgressMonitorPtr The IProgressMonitorPtr object

getAbort()

```
virtual IAbortPtr JawsMako::IProgressMonitor::getAbort ( ) const  [pure virtual]
```

Return the smart pointer to an IAbort object.

Returns

IAbortPtr The smart pointer to the IAbort object or NULL if there is no such object.

getProgressTick()

```
virtual IProgressTickPtr JawsMako::IProgressMonitor::getProgressTick ( ) const  [pure virtual]
```

Return the IProgressTick object or NULL if there was no such object.

Returns

IProgressTickPtr The smart pointer to the IProgressTick object or NULL if there is no such object.

The documentation for this class was generated from the following file:

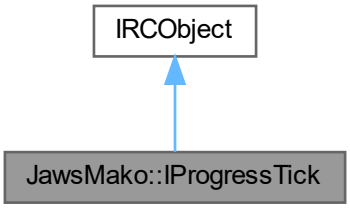
- jawsmako.h

7.392 JawsMako::IProgressTick Class Reference

An abstract class allowing a callback to monitor the progress of a task. The callback can either pass an integer (from 0 to a maximum value) or a float (0.0 to 1.0) which represents the progress of the task. The task needs to call the tick() method to indicate incremental progress. If the task has completed (even if it's premature) then tickMax() should be called instead of tick().

`#include <jawsmako.h>`

Inheritance diagram for JawsMako::IProgressTick:



Public Types

- typedef void(∗ **FloatProgressCallbackFunc**) (void ∗priv, float progress)

  A callback type for receiving progress information from 0.0 (starting) through 1.0 (complete)
- typedef void(∗ **IntProgressCallbackFunc**) (void ∗priv, uint32 currentCount, uint32 maxCount)

  A callback type for receiving progress information as an integer. The value of currentCount goes from 1 to maxCount.

Public Member Functions

- virtual void setCallback (IntProgressCallbackFunc cbFunc, void ∗cbPriv)=0

  Set the integer type callback function.
- virtual void setCallback (FloatProgressCallbackFunc cbFunc, void ∗cbPriv)=0

  Set the integer type callback function.
- virtual void **tickReset** ()=0

  Reset the internal counter to 0.
- virtual void **tick** ()=0

  Increment the tick counter and optionally invoke the callback function.
- virtual void **tickMax** ()=0

  Set the tick counter to the maximum value and invoke the callback function.
- virtual void setTickMax (const uint32 tickMax)=0

  Set the maximum counter value.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IProgressTickPtr create (IntProgressCallbackFunc cbFunc, void *cbPriv)

  Create the IProgressTick object with an integer callback function.
- static JAWSMAKO_API IProgressTickPtr create (FloatProgressCallbackFunc cbFunc, void *cbPriv)

  Create the IProgressTick object with an floating point callback function.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.392.1 Detailed Description

An abstract class allowing a callback to monitor the progress of a task. The callback can either pass an integer (from 0 to a maximum value) or a float (0.0 to 1.0) which represents the progress of the task. The task needs to call the tick() method to indicate incremental progress. If the task has completed (even if it's premature) then tickMax() should be called instead of tick().

7.392.2 Member Function Documentation

create() [1/2]

```
static JAWSMAKO_API IProgressTickPtr JawsMako::IProgressTick::create (
            FloatProgressCallbackFunc cbFunc,
            void * cbPriv )  [static]
```

Create the IProgressTick object with an floating point callback function.

Parameters

in	*cbFunc*	A floating point callback function that will be called on each tick.
in	*cbPriv*	The parameter that will be passed to the callback function.

Returns

IProgressTickPtr The IProgressTick object

create() [2/2]

```
static JAWSMAKO_API IProgressTickPtr JawsMako::IProgressTick::create (
            IntProgressCallbackFunc cbFunc,
            void * cbPriv )  [static]
```

Create the IProgressTick object with an integer callback function.

Parameters

in	*cbFunc*	An integer callback function that will be called on each tick.
in	*cbPriv*	The parameter that will be passed to the callback function.

Returns

IProgressTickPtr The IProgressTick object

setCallback() [1/2]

```
virtual void JawsMako::IProgressTick::setCallback (
            FloatProgressCallbackFunc cbFunc,
            void * cbPriv )  [pure virtual]
```

Set the integer type callback function.

Parameters

in	*cbFunc*	The integer type callback function to be invoked
in	*cbPriv*	The parameter to be passed back to the callback whenever it is invoked.

setCallback() [2/2]

```
virtual void JawsMako::IProgressTick::setCallback (
            IntProgressCallbackFunc cbFunc,
            void * cbPriv )  [pure virtual]
```

Set the integer type callback function.

Parameters

in	*cbFunc*	The integer type callback function to be invoked
in	*cbPriv*	The parameter to be passed back to the callback whenever it is invoked.

setTickMax()

```
virtual void JawsMako::IProgressTick::setTickMax (
            const uint32 tickMax )  [pure virtual]
```

Set the maximum counter value.

Parameters

in	*tickMax*	The maximum tick counter value.

The documentation for this class was generated from the following file:

- jawsmako.h

7.393 JawsMako::IPSCommentMonitor Class Reference

Interface allowing users to monitor comments in PostScript input. Use setCommentMonitor() to install subclasses of this type.

```
#include <distiller.h>
```

Public Member Functions

- virtual const CRawStringVect & getComments () const =0
  Gets a list of comment keys.
- virtual void comment (const RawString &comment)=0
  Invoked when a comment is detected in the input.

7.393.1 Detailed Description

Interface allowing users to monitor comments in PostScript input. Use setCommentMonitor() to install subclasses of this type.

7.393.2 Member Function Documentation

comment()

```
virtual void JawsMako::IPSCommentMonitor::comment (
            const RawString & comment )  [pure virtual]
```

Invoked when a comment is detected in the input.

Use getComments() to supply a list of comments to monitor.

Parameters

comment	The comment line from the PostScript input.

getComments()

```
virtual const CRawStringVect & JawsMako::IPSCommentMonitor::getComments ( ) const  [pure virtual]
```

Gets a list of comment keys.

The comment() function will be invoked for any PostScript comments encountered that begin with the keys returned from this function.

Returns

CRawStringVect The list of comments to monitor.

The documentation for this class was generated from the following file:

- distiller.h

7.394 JawsMako::IPSInjector Class Reference

Interface allowing users of IPSOutput to inject raw PostScript directly into the output stream at strategic points in the output process. Use IPSOutput::setInjector() to install subclasses of this type.

```
#include <psoutput.h>
```

Public Member Functions

- virtual void beforeFirstByte (const IOutputStreamPtr &psStream)

  Invoked by the PostScript writer after the output PostScript stream is opened, but before any data is written to the output stream.
- virtual void beforePsHeader (const IOutputStreamPtr &psStream)

  Invoked by the PostScript writer just before the PostScript header is written to the stream. Currently, this is effectively the same as beforeFirstByte() above as Mako will not currently write anything at the top of the PostScript stream before the PostScript header.
- virtual void afterBeginSetup (const IOutputStreamPtr &psStream)

  Invoked by the PostScript writer just after %BeginSetup is written to the stream.
- virtual void beforeEndSetup (const IOutputStreamPtr &psStream)

  Invoked by the PostScript writer just before %EndSetup is written to the stream.
- virtual void afterBeginPageSetup (uint32 pageNumber, const IOutputStreamPtr &psStream)

  Invoked by the PostScript writer just after %BeginPageSetup is written to the stream.
- virtual void beforeEndPageSetup (uint32 pageNumber, const IOutputStreamPtr &psStream)

  Invoked by the PostScript writer just before %EndPageSetup is written to the stream.
- virtual void beforeShowpage (uint32 pageNumber, const IOutputStreamPtr &psStream)

  Invoked by the PostScript writer just before "showpage" is written to the stream.
- virtual void afterLastByte (const IOutputStreamPtr &psStream)

  Invoked by the PostScript writer after the last byte of normal PostScript data is written to the stream.

7.394.1 Detailed Description

Interface allowing users of IPSOutput to inject raw PostScript directly into the output stream at strategic points in the output process. Use IPSOutput::setInjector() to install subclasses of this type.

7.394.2 Member Function Documentation

afterBeginPageSetup()

```
virtual void JawsMako::IPSInjector::afterBeginPageSetup (
            uint32 pageNumber,
            const IOutputStreamPtr & psStream )  [inline], [virtual]
```

Invoked by the PostScript writer just after %BeginPageSetup is written to the stream.

Parameters

pageNumber	The page number being prepared, with 0 indicating the first page.
psStream	The PostScript stream. Do not retain this stream after your handler exits, and only write during the execution of your handler.

afterBeginSetup()

```
virtual void JawsMako::IPSInjector::afterBeginSetup (
            const IOutputStreamPtr & psStream )  [inline], [virtual]
```

Invoked by the PostScript writer just after %BeginSetup is written to the stream.

Parameters

psStream	The PostScript stream. Do not retain this stream after your handler exits, and only write during the execution of your handler.

afterLastByte()

```
virtual void JawsMako::IPSInjector::afterLastByte (
            const IOutputStreamPtr & psStream )  [inline], [virtual]
```

Invoked by the PostScript writer after the last byte of normal PostScript data is written to the stream.

Parameters

psStream	The PostScript stream. Do not retain this stream after your handler exits, and only write during the execution of your handler.

beforeEndPageSetup()

```
virtual void JawsMako::IPSInjector::beforeEndPageSetup (
            uint32 pageNumber,
            const IOutputStreamPtr & psStream )  [inline], [virtual]
```

Invoked by the PostScript writer just before %EndPageSetup is written to the stream.

Parameters

pageNumber	The page number being prepared, with 0 indicating the first page.
psStream	The PostScript stream. Do not retain this stream after your handler exits, and only write during the execution of your handler.

beforeEndSetup()

```
virtual void JawsMako::IPSInjector::beforeEndSetup (
            const IOutputStreamPtr & psStream )  [inline], [virtual]
```

Invoked by the PostScript writer just before %EndSetup is written to the stream.

Parameters

psStream	The PostScript stream. Do not retain this stream after your handler exits, and only write during the execution of your handler.

beforeFirstByte()

```
virtual void JawsMako::IPSInjector::beforeFirstByte (
            const IOutputStreamPtr & psStream )  [inline], [virtual]
```

Invoked by the PostScript writer after the output PostScript stream is opened, but before any data is written to the output stream.

Parameters

psStream	The PostScript stream. Do not retain this stream after your handler exits, and only write during the execution of your handler.

beforePsHeader()

```
virtual void JawsMako::IPSInjector::beforePsHeader (
            const IOutputStreamPtr & psStream )  [inline], [virtual]
```

Invoked by the PostScript writer just before the PostScript header is written to the stream. Currently, this is effectively the same as beforeFirstByte() above as Mako will not currently write anything at the top of the PostScript stream before the PostScript header.

Parameters

psStream	The PostScript stream. Do not retain this stream after your handler exits, and only write during the execution of your handler.

beforeShowpage()

```
virtual void JawsMako::IPSInjector::beforeShowpage (
            uint32 pageNumber,
            const IOutputStreamPtr & psStream )  [inline], [virtual]
```

Invoked by the PostScript writer just before "showpage" is written to the stream.

Parameters

pageNumber	The page number being output, with 0 indicating the first page.
psStream	The PostScript stream. Do not retain this stream after your handler exits, and only write during the execution of your handler.

The documentation for this class was generated from the following file:

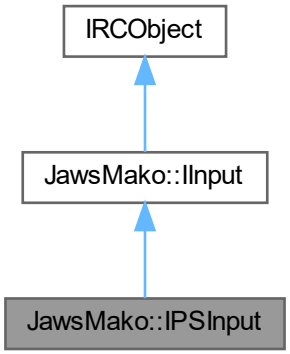
- psoutput.h

7.395 JawsMako::IPSInput Class Reference

An instance of the JawsMako PS input class.

```
#include <psinput.h>
```

Inheritance diagram for JawsMako::IPSInput:



Public Member Functions

- virtual void enableUnencapsulatedMode (bool unencapsulated)=0

  Enable or disable unencapsulated mode.
- virtual void setProlog (const IInputStreamPtr &prolog)=0

  Set a prolog stream to be consumed by the PostScript interpreter before the input stream is processed, or NULL to clear.

Public Member Functions inherited from JawsMako::IInput

- virtual IDocumentAssemblyPtr open (const U8String &pathToFile)=0

  Open a file on disk, returning the IDocumentAssembly representing the contents.
- virtual IDocumentAssemblyPtr open (const String &pathToFile)=0

  Open a file on disk, returning the IDocumentAssembly representing the contents. Takes a wide character string.
- virtual IDocumentAssemblyPtr open (const IInputStreamPtr &inputStream)=0

  Open a stream, returning the IDocumentAssembly representing the contents.
- virtual void setSequentialMode (bool sequential)=0

  Set/unset sequential mode on this input.
- virtual void setParameter (const U8String &param, const U8String &value)=0

  Apply a key value pair output parameter with a string value. The parameter name is case insensitive. Please refer to the supplied documentation for the details of the available parameters and their ranges.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IPSInputPtr create (const IJawsMakoPtr &jawsMako, bool eps=false, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create an input for reading source documents in PostScript format.

Static Public Member Functions inherited from JawsMako::IInput

- static JAWSMAKO_API IInputPtr create (const IJawsMakoPtr &jawsMako, eFileFormat format, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create an input for reading source documents in the given format. The following formats are currently supported:

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.395.1 Detailed Description

An instance of the JawsMako PS input class.

7.395.2 Member Function Documentation

create()

```
static JAWSMAKO_API IPSInputPtr JawsMako::IPSInput::create (
            const IJawsMakoPtr & jawsMako,
            bool eps = false,
            const IProgressMonitorPtr & progressMonitor = IProgressMonitorPtr() )  [static]
```

Create an input for reading source documents in PostScript format.

The IPSInput interface allows PostScript to be imported into JawsMako. The input PostScript is converted to PDF and then read in using the IPDFInput interface.

Parameters

jawsMako	The IJawsMako object
eps	Whether or not the source is Encapsulated PostScript. The default is false.
progressMonitor	A smart pointer to an IProgressMonitor object which can be NULL if no such object was passed in.

Returns

IPSInputPtr the PS input

enableUnencapsulatedMode()

```
virtual void JawsMako::IPSInput::enableUnencapsulatedMode (
            bool unencapsulated )  [pure virtual]
```

Enable or disable unencapsulated mode.

```
Setting this mode to true changes the parser behaviour
as follows.

- The input stream is not opened; it is assumed to point to
  the start of a PostScript stream.
- The PostScript interpreter will not consume data after the point the
  interpreter either exhausts the stream or a UEL is encountered.

Whereas the default (false) will:

- open() the input stream.
- attempt to consume the entire stream

As such unencapsulated mode will only work with the stream version of IInput::open()
```

setProlog()

```
virtual void JawsMako::IPSInput::setProlog (
            const IInputStreamPtr & prolog )  [pure virtual]
```

Set a prolog stream to be consumed by the PostScript interpreter before the input stream is processed, or NULL to clear.

This stream will be both opened and closed, and will not be cloned; take care when opening multiple PostScript inputs with the same prolog simultaneously.

Parameters

prolog	The prolog stream to use.

The documentation for this class was generated from the following file:

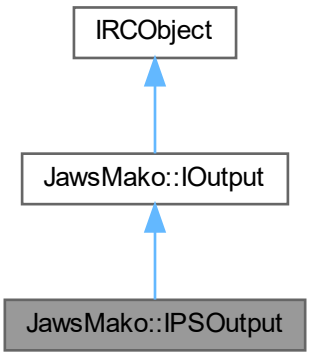
- psinput.h

7.396 JawsMako::IPSOutput Class Reference

Interface for the PS IOutput class.

```
#include <psoutput.h>
```

Inheritance diagram for JawsMako::IPSOutput:



Public Types

- enum eImageCompression

  Enumeration for image compression formats.

Public Member Functions

- virtual void **setResolution** (float resolution)=0

  Set the target resolution for the output, in dots per inch. The default is 600. Equivalent to calling setParameter with the parameter name "TargetResolution".
- virtual void setStreamingOutput (bool enable=true)=0

  Set whether or not streaming output should be enabled.
- virtual void setTargetColorSpace (const IDOMColorSpacePtr &targetSpace)=0

  Set the target colour space for the intended output device. The default is DeviceCMYK. Equivalent to calling setParameter with the param name "TargetColorSpace" with appropriate values (please refer to documentation).
- virtual void setTargetProfile (const IDOMICCProfilePtr &profile)=0

  Set the target color space for the intended output device using an ICC profile. Equivalent to calling setParameter() with the param name "TargetProfile" with the value as the path to the profile.
- virtual void setConvertAllObjectsToTargetColorSpace (bool convert=true)=0

  Sets whether or not all objects should be converted to the target color space. If false, only objects that require rendering, or use a color space that cannot will represented in PostScript will be forcibly converted to the target colour space. Exquivalent to calling setParameter() with the param name "ConvertAllObjectsToTargetColorSpace".
- virtual void **setIgnoreDeviceGrayDuringColorConversion** (bool ignore=true)=0

  Sets whether to ignore DeviceGray objects when color conversion is to be performed. The default is true. Equivalent to calling setParameter() with the param name "IgnoreDeviceGrayDuringColorConversion".
- virtual void **setInjector** (IPSInjector *injector)=0

  Set the IPSInjector instance to use to inject PostScript directly into the output PostScript stream at strategic points in the output process. Note that this routine does not take ownership of the passed pointer, which must remain allocated until after the output is complete.
- virtual void setPreferredColorImageCompression (eImageCompression compression)=0

  Set the desired image compression for color images that need to be reencoded. The default is eICDCT.
- virtual void setPreferredGrayImageCompression (eImageCompression compression)=0

  Set the desired image compression for gray images that need to be reencoded. The default is eICDCT.
- virtual void setPreferredMonoImageCompression (eImageCompression compression)=0

  Set the desired image compression for monochrome images that need to be reencoded. The default is eICCCITT. JPEG/DCT is not allowed for mono images.
- virtual void setJPEGQuality (uint8 quality)=0

  Set the JPEG quality to use when compressing images in DCT format. Equivalent to calling setParameter() with the parameter name "JPEGQuality" and the value being the required quality. The default is 3 - normal quality.
- virtual void setColorImageMaxResolution (float resolution, float threshold=0.0f, IDOMImageDownsamplerFilter::eDownsampling method=IDOMImageDownsamplerFilter::eBicubic)=0

  Set the desired maximum resolution, threshold and downsampling method for colour images.
- virtual void setGrayImageMaxResolution (float resolution, float threshold=0.0f, IDOMImageDownsamplerFilter::eDownsamplingI method=IDOMImageDownsamplerFilter::eBicubic)=0

  Set the desired maximum resolution, threshold and downsampling method for gray images.
- virtual void setMonoImageMaxResolution (float resolution, float threshold=0.0f, IDOMImageDownsamplerFilter::eDownsampling method=IDOMImageDownsamplerFilter::eSubsample)=0

  Set the desired maximum resolution, threshold and downsampling method for monochrome images.

Public Member Functions inherited from JawsMako::IOutput

- virtual void **setPreset** (const U8String &preset)=0

  Configure the output according to a general preset. Please see the supplied documentation for details of these presets. The default is "Preserve" which will attempt to produce output as close to the input as possible for the output format. A string value can be used for any parameter and will be converted as necessary.
- virtual void **setParameter** (const U8String &param, const U8String &value)=0

  Apply a key value pair output parameter with a string value. The parameter name is case insensitive. Please refer to the supplied documentation for the details of the available parameters and their ranges.

- virtual void setAllowedPermissionsFlags (uint32 allowedPermissions)=0

  Control whether or not assemblies with certain security permission flags are allowed to be written by this output.
- virtual void writeAssembly (const IDocumentAssemblyPtr &assembly, const U8String &pathToFile)=0

  Write the given document assembly to a file on disk.
- virtual void writeAssembly (const IDocumentAssemblyPtr &assembly, const String &pathToFile)=0

  Write the given document assembly to a file on disk, specified by a wide character string.
- virtual void writeAssembly (const IDocumentAssemblyPtr &assembly, const IOutputStreamPtr &stream)=0

  Write the given document assembly to a stream.
- virtual IOutputWriterPtr openWriter (const IDocumentAssemblyPtr &assembly, const U8String &pathToFile)=0

  Create an output writer for the given assembly, targeting a file on disk. This is designed to allow streaming output, or to deal with situations where an operation would require too much memory to hold an entire edited assembly in memory at once.
- virtual IOutputWriterPtr openWriter (const IDocumentAssemblyPtr &assembly, const String &pathToFile)=0

  Create an output writer for the given assembly, targeting a file on disk. As above, but with the file specified in a wide character string.
- virtual IOutputWriterPtr openWriter (const IDocumentAssemblyPtr &assembly, const IOutputStreamPtr &stream)=0

  Create an output writer for the given assembly, targeting a stream.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IPSOutputPtr create (const IJawsMakoPtr &jawsMako, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create a PS Output instance.

Static Public Member Functions inherited from JawsMako::IOutput

- static JAWSMAKO_API IOutputPtr create (const IJawsMakoPtr &jawsMako, eFileFormat format, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create an output for writing source in the given format.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.396.1 Detailed Description

Interface for the PS IOutput class.

```
Note that there is a limit on the number of PDF and PS
output operations that may be in progress simultaneously.
These are:
- iOS - 1 PDF or PS output operation at a time
- Android - 1 PDF or PS output operation at a time
- Windows/MacOS/X 32 bit - 8 PDF or PS output operations at a time, subject to available memory
- Windows/MacOS/X 64 bit - 96 PDF or PS output operations at a time for most tool chains
                         - VS2015 Static builds are currently limited to 48
- Windows UWP - 1 PDF or PS output operation at a time
```

7.396.2 Member Function Documentation

create()

```
static JAWSMAKO_API IPSOutputPtr JawsMako::IPSOutput::create (
            const IJawsMakoPtr & jawsMako,
            const IProgressMonitorPtr & progressMonitor = IProgressMonitorPtr() )  [static]
```

Create a PS Output instance.

Parameters

jawsMako	The JawsMako instance.
progressMonitor	The progress monitor which allows aborting an operation or registering a progress callback.

setColorImageMaxResolution()

```
virtual void JawsMako::IPSOutput::setColorImageMaxResolution (
            float resolution,
            float threshold = 0.0f,
            IDOMImageDownsamplerFilter::eDownsamplingMethod method = IDOMImageDownsamplerFilter↩
::eBicubic )  [pure virtual]
```

Set the desired maximum resolution, threshold and downsampling method for colour images.

The default behaviour is leave the image resolution unchanged. Equivalent to calling setParameter() with the param names "ColorImageDownsamplingResolution", "ColorImageDownsamplingThreshold" and "ColorImage↩ DownsamplingMethod" with the respective values.

Parameters

resolution	The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default.
threshold	The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution.
method	The method to use when downsampling. The default is bicubic for colour images.

setConvertAllObjectsToTargetColorSpace()

```
virtual void JawsMako::IPSOutput::setConvertAllObjectsToTargetColorSpace (
            bool convert = true )  [pure virtual]
```

Sets whether or not all objects should be converted to the target color space. If false, only objects that require rendering, or use a color space that cannot will represented in PostScript will be forcibly converted to the target colour space. Exquivalent to calling setParameter() with the param name "ConvertAllObjectsToTargetColorSpace".

The default is true.

setGrayImageMaxResolution()

```
virtual void JawsMako::IPSOutput::setGrayImageMaxResolution (
            float resolution,
            float threshold = 0.0f,
            IDOMImageDownsamplerFilter::eDownsamplingMethod method = IDOMImageDownsamplerFilter↩
::eBicubic )  [pure virtual]
```

Set the desired maximum resolution, threshold and downsampling method for gray images.

The default behaviour is leave the image resolution unchanged. Equivalent to calling setParameter() with the param names "GrayImageDownsamplingResolution", "GrayImageDownsamplingThreshold" and "GrayImage↩ DownsamplingMethod" with the respective values.

Parameters

resolution	The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default.
threshold	The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution.
method	The method to use when downsampling. The default is bicubic for gray images.

setJPEGQuality()

```
virtual void JawsMako::IPSOutput::setJPEGQuality (
            uint8 quality )  [pure virtual]
```

Set the JPEG quality to use when compressing images in DCT format. Equivalent to calling setParameter() with the parameter name "JPEGQuality" and the value being the required quality. The default is 3 - normal quality.

Parameters

quality	The desired quality level, with 1 being lowest quality and 5 being highest quality.

setMonoImageMaxResolution()

```
virtual void JawsMako::IPSOutput::setMonoImageMaxResolution (
            float resolution,
            float threshold = 0.0f,
```

```
            IDOMImageDownsamplerFilter::eDownsamplingMethod method = IDOMImageDownsamplerFilter↩
::eSubsample )  [pure virtual]
```

Set the desired maximum resolution, threshold and downsampling method for monochrome images.

The default behaviour is leave the image resolution unchanged. Equivalent to calling setParameter() with the param names "MonoImageDownsamplingResolution", "MonoImageDownsamplingThreshold" and "MonoImage↩ DownsamplingMethod" with the respective values.

Parameters

resolution	The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default.
threshold	The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution.
method	The method to use when downsampling. The default is subsample for monochrome images; using any other method will result in grayscale output.

setPreferredColorImageCompression()

```
virtual void JawsMako::IPSOutput::setPreferredColorImageCompression (
            eImageCompression compression )  [pure virtual]
```

Set the desired image compression for color images that need to be reencoded. The default is eICDCT.

CCITT is not allowed for colour images.

Note: this is advisory only and may not be honoured in all cases.

Equivalent to calling setParameter() with the parameter name "ColorImageCompression" with appropriate values (please refer to documentation).

Parameters

compression	The desired compression.

setPreferredGrayImageCompression()

```
virtual void JawsMako::IPSOutput::setPreferredGrayImageCompression (
            eImageCompression compression )  [pure virtual]
```

Set the desired image compression for gray images that need to be reencoded. The default is eICDCT.

CCITT is not allowed for gray images.

Note: this is advisory only and may not be honoured in all cases.

Equivalent to calling setParameter() with the parameter name "GrayImageCompression" with appropriate values (please refer to documentation).

Parameters

compression	The desired compression.

setPreferredMonoImageCompression()

```
virtual void JawsMako::IPSOutput::setPreferredMonoImageCompression (
            eImageCompression compression )  [pure virtual]
```

Set the desired image compression for monochrome images that need to be reencoded. The default is eICCCITT. JPEG/DCT is not allowed for mono images.

Note: this is advisory only and may not be honoured in all cases.

Equivalent to calling setParameter() with the parameter name "MonoImageCompression" with appropriate values (please refer to documentation).

Parameters

compression	The desired compression.

setStreamingOutput()

```
virtual void JawsMako::IPSOutput::setStreamingOutput (
            bool enable = true )  [pure virtual]
```

Set whether or not streaming output should be enabled.

```
    Streaming output provides output as soon as it is ready, sending
    it to the output file or stream. Fonts are defined incrementally
    and a number of standard document structure comments are
    deferred to the end job trailer. This mode is suitable for use
    with streaming consumers, such as printers.

    If streaming output is set to false, then the output is withheld
    until the entire assembly has been processed. This enables
    providing all document resources (such as fonts and color
    spaces up front), and may be more useful for applications that
    further process the PostScript.
    The default is true,

    Equivalent to calling setParameter with the parameter name
    "StreamingOutput".
```

setTargetColorSpace()

```
virtual void JawsMako::IPSOutput::setTargetColorSpace (
            const IDOMColorSpacePtr & targetSpace )  [pure virtual]
```

Set the target colour space for the intended output device. The default is DeviceCMYK. Equivalent to calling set↩ Parameter with the param name "TargetColorSpace" with appropriate values (please refer to documentation).

Parameters

targetSpace	The desired color space. Must not be LAB, Indexed, DeviceN or scRGB. Any ICC space must have one, three, or four components.

setTargetProfile()

```
virtual void JawsMako::IPSOutput::setTargetProfile (
            const IDOMICCProfilePtr & profile )  [pure virtual]
```

Set the target color space for the intended output device using an ICC profile. Equivalent to calling setParameter() with the param name "TargetProfile" with the value as the path to the profile.

Parameters

profile	The desired profile.

The documentation for this class was generated from the following file:

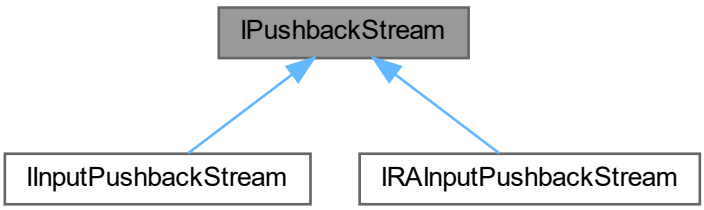
- psoutput.h

7.397 IPushbackStream Class Reference

Abstract base class (for input streams only) that provides a "push back" mechanism. When used with random access streams, the pushback buffer is invalidated by setPos().

```
#include <edlstream.h>
```

Inheritance diagram for IPushbackStream:



Public Member Functions

- virtual ~**IPushbackStream** ()
  Virtual destructor.
- virtual bool pushBack (uint8 byte)=0
  Push back a byte.
- virtual bool pushBack (const void *buffer, int32 count)=0
  Push back from a buffer.

7.397.1 Detailed Description

Abstract base class (for input streams only) that provides a "push back" mechanism. When used with random access streams, the pushback buffer is invalidated by setPos().

7.397.2 Member Function Documentation

pushBack() [1/2]

```
virtual bool IPushbackStream::pushBack (
            const void * buffer,
            int32 count )  [pure virtual]
```

Push back from a buffer.

Parameters

buffer	Buffer to push back from.
count	Number of bytes to push back.

Returns

bool True if method succeeded.

pushBack() [2/2]

```
virtual bool IPushbackStream::pushBack (
            uint8 byte )  [pure virtual]
```

Push back a byte.

Parameters

byte	Byte to be pushed back.

Returns

bool True if method succeeded.

The documentation for this class was generated from the following file:

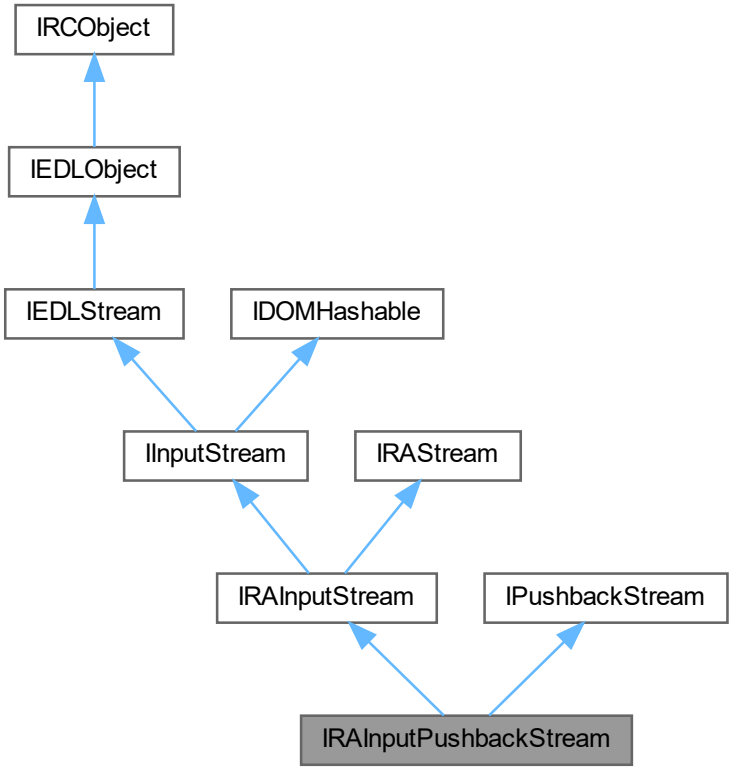
- edlstream.h

7.398 IRAInputPushbackStream Class Reference

Random-access Input Stream with pushback support.

```
#include <edlstream.h>
```

Inheritance diagram for IRAInputPushbackStream:



Additional Inherited Members

Public Member Functions inherited from IInputStream

- virtual int32 read (void *buffer, int32 count)=0

  Read specified number of bytes from a stream into buffer.
- virtual int8 read ()=0

  Read single byte from a stream.
- virtual bool eof () const =0

  Determine if the stream has exhausted.
- virtual int64 skip (int64 count)

  Skip a specified number of bytes.
- virtual bool getSourceFilePath (EDLSysString &sourcePath)=0

  If available, find the file path of the file that this stream references, or for streams that take their input from other streams, the ultimate source stream. If at all possible, the source path will be canonicalised, but this is not guaranteed. If a canonical name is absolutely required, check getCanonicalSourceFilePath().
- virtual bool getCanonicalSourceFilePath (EDLSysString &sourceCanonicalPath)=0

If available, find the canonical file path of the file that this stream references, or for streams that take their input from other streams, the ultimate source stream. For canonical names to be discovered on all platforms, the underlying file must exist. If the canonical file path cannot be determined, false will be returned. In which case, you may wish to try getSourceFilePath() above.

- virtual bool completeRead (void *buffer, int32 count)

  Perform a complete read.
- virtual void completeReadE (void *buffer, int32 count)

  As completeRead(), but throws an exception if the operation fails.
- virtual bool hash (uint64 &hash)

  Obtain a 64-bit hash of the stream. Please note that this requires reading the stream and is therefore not thread safe. If thread safety is desired, make a clone of the stream first.

Public Member Functions inherited from IEDLStream

- virtual bool isValid () const =0

  Determine stream validity.
- virtual bool open ()

  Opens the stream.
- virtual void **openE** ()=0

  As per open(), but will throw an exception on failure (IEDLError) that for some stream types may contain additional failure information.
- virtual void **close** ()=0

  Closes the stream.
- virtual int64 getPos ()=0

  Get current stream position.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Public Member Functions inherited from IRAStream

- virtual ~**IRAStream** ()

  Virtual destructor.
- virtual int64 length ()=0

  Get length of the stream.
- virtual bool setPos (int64 newPos)=0

  Set stream position.
- virtual void setPosE (int64 newPos)

  Set stream position, but throw an exception on failure.

Public Member Functions inherited from IPushbackStream

- virtual ~**IPushbackStream** ()

  Virtual destructor.
- virtual bool pushBack (uint8 byte)=0

  Push back a byte.
- virtual bool pushBack (const void *buffer, int32 count)=0

  Push back from a buffer.

Static Public Member Functions inherited from IInputStream

- static EDL_API IRAInputStreamPtr createFromFile (IEDLClassFactory *pFactory, const EDLSysString &path)

  Creation function for an IInputStream for a file on disk. Throws an IEDLError exception on failure.
- static EDL_API IRAInputStreamPtr createFromFile (IEDLClassFactory *pFactory, const EDLString &path)

  Creation function for an IInputStream for a file on disk. Throws an IEDLError exception on failure.
- static EDL_API IRAInputStreamPtr createFromFileShared (IEDLClassFactory *pFactory, const EDLSys↩ String &path)

  Creation function for an IInputStream for a file on disk. Similar to createFromFile, but if this file is cloned, only one file handle will be open at a time. This is slower, but is useful if many users are likely to want to have this file open at the same file. On some systems, having too many files open may result in errors. Performance should be acceptable providing there is not a great deal of contention and simultaneous access.
- static EDL_API IRAInputStreamPtr createFromFileShared (IEDLClassFactory *pFactory, const EDLString &path)

  Creation function for an IInputStream for a file on disk. Similar to createFromFile, but if this stream is cloned, only one file handle will be open at a time. This is slower, but is useful if many users are likely to want to have this file open at the same file. On some systems, having too many files open may result in errors. Performance should be acceptable providing there is not a great deal of contention and simultaneous access.
- static EDL_API IRAInputStreamPtr createSharedFromStream (IEDLClassFactory *pFactory, const IRAInputStreamPtr &stream)

  Creation function for a shared stream overlaying an existing stream. If this stream is cloned, the underlying file will not be cloned. This is useful if many users are likely to want to have this file open at the same time. On some systems, having too many files open may result in errors. Performance should be acceptable providing there is not a great deal of contention and simultaneous access. Only possible for random access streams. If the stream is already shared, the stream will be returned as is.
- static EDL_API IRAInputStreamPtr createFromMemory (IEDLClassFactory *pFactory, const void *mem, uint32 length, bool copy=false, bool free=true)

  Creation function for an IInputStream for data in memory. Throws an IEDLError exception on failure.
- static EDL_API IInputStreamPtr createFromUserReadFunc (IEDLClassFactory *pFactory, UserStream↩ ReadFunc readFunc, void *priv)

  Creation function for an IInputStream from a user function that provides data. Throws an IEDLError exception on failure.

- static EDL_API IRAInputStreamPtr createFromRAUserFunc (IEDLClassFactory *pFactory, int64 length, UserRAReadFunc readFunc, void *priv)

  Creation function for an IRAInputStream from a user function that provides random access. Throws an IEDLError exception on failure.
- static EDL_API IRAInputStreamPtr createRandomAccessFromNonRandomAccess (IEDLClassFactory *p↩ Factory, const IInputStreamPtr &stream)

  Creation function to make a random access stream from a stream that is not random access. Currently, the temporary store will be used to store a copy of the source data, but a concrete implementation method should not be assumed. If the source stream is random access, it will be returned as is. Throws an IEDLError exception on failure.
- static EDL_API IRAInputStreamPtr createFromNewFileWithContents (IEDLClassFactory *pFactory, const EDLSysString &path, const IInputStreamPtr &stream)

  Creation function for an IInputStream for a file on disk created with the contents of an existing stream. Throws an IEDLError exception on failure.
- static EDL_API IRAInputStreamPtr createFromNewFileWithContents (IEDLClassFactory *pFactory, const EDLString &path, const IInputStreamPtr &stream)

  Creation function for an IInputStream for a file on disk created with the contents of an existing stream. Throws an IEDLError exception on failure.
- static EDL_API IInputStreamPtr createSubFile (IEDLClassFactory *pFactory, const IInputStreamPtr &stream, int64 offset, int64 length)

  Creation routine for a stream representing a portion of a file on disk. If the source file is random access, then the created file shall be also. Throws an IEDLError exception on failure.
- static EDL_API IInputStreamPtr createFromFlateCompressed (IEDLClassFactory *pFactory, const IInput↩ StreamPtr &stream, bool raw=true, bool ignoreChecksums=false)

  Creation routine for a input stream for decompressing a flate stream. Throws an IEDLError exception on failure.
- static EDL_API IInputStreamPtr createFromLz4Compressed (IEDLClassFactory *pFactory, const IInput↩ StreamPtr &stream)

  Creation routine for a input stream for decompressing an lz4 block compressed stream. Throws an IEDLError exception on failure. Note: This is not intended for interoperability with other LZ4 formats, but is useful for things like temporary storage.
- static EDL_API IInputStreamPtr createFromPredictorCompressed (IEDLClassFactory *pFactory, const IInputStreamPtr &stream, uint8 predictor, uint8 colors, uint8 bitsPerComponent, uint32 columns)

  Creation routine for a input stream for applying a predictor algorithm to a compressed stream. Throws an IEDLError exception on failure.
- static EDL_API IInputStreamPtr createCompositeStream (IEDLClassFactory *pFactory, const CIInput↩ StreamVect &streams)

  Creation routine for creating a composite input stream representing the concatenation of a series of streams.
- static EDL_API IInputPushbackStreamPtr createPushbackStream (IEDLClassFactory *pFactory, const IInputStreamPtr &sourceStream, bool clone=true)

  Creation routine for creating a push-back stream using a non-pushback stream as a data source. If the source stream is random access, the resulting stream will also be random access.
- static EDL_API IInputStreamPtr createUelStream (IEDLClassFactory *pFactory, const IInputStreamPtr &sourceStream)

  Creation routine for a stream that emulates an end-of-file condition should a Universal End of Language (UEL) sequence be encountered in an underlying data source. The resulting stream is not random access. The underlying stream is neither opened nor closed. This stream type is useful for restricting parsing of substreams inside a print stream. Streams of this type are not clonable. Please also note, that this stream must read ahead of read requests in order to search for UEL sequences. However it will never read beyond the UEL sequence.
- static EDL_API IInputStreamPtr createTbcpStream (IEDLClassFactory *pFactory, const IInputStreamPtr &sourceStream, bool openSource=false)

  Creation routine for a stream that reads data from an underlying stream encoded using Adobe's Tagged Binary Communication Protocol (TBCP). The resulting stream is not random access. Streams of this type are not clonable.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.398.1 Detailed Description

Random-access Input Stream with pushback support.

The documentation for this class was generated from the following file:

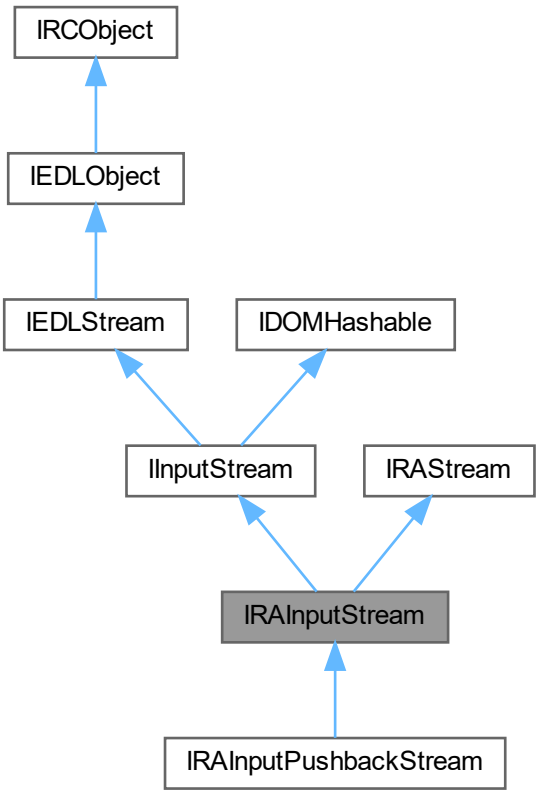
- edlstream.h

7.399 IRAInputStream Class Reference

Random Access Input Stream.

`#include <edlstream.h>`

Inheritance diagram for IRAInputStream:



Additional Inherited Members

Public Member Functions inherited from IInputStream

- virtual int32 read (void *buffer, int32 count)=0

  Read specified number of bytes from a stream into buffer.
- virtual int8 read ()=0

  Read single byte from a stream.
- virtual bool eof () const =0

  Determine if the stream has exhausted.
- virtual int64 skip (int64 count)

  Skip a specified number of bytes.
- virtual bool getSourceFilePath (EDLSysString &sourcePath)=0

  If available, find the file path of the file that this stream references, or for streams that take their input from other streams, the ultimate source stream. If at all possible, the source path will be canonicalised, but this is not guaranteed. If a canonical name is absolutely required, check getCanonicalSourceFilePath().
- virtual bool getCanonicalSourceFilePath (EDLSysString &sourceCanonicalPath)=0

  If available, find the canonical file path of the file that this stream references, or for streams that take their input from other streams, the ultimate source stream. For canonical names to be discovered on all platforms, the underlying file must exist. If the canonical file path cannot be determined, false will be returned. In which case, you may wish to try getSourceFilePath() above.
- virtual bool completeRead (void *buffer, int32 count)

  Perform a complete read.
- virtual void completeReadE (void *buffer, int32 count)

  As completeRead(), but throws an exception if the operation fails.
- virtual bool hash (uint64 &hash)

  Obtain a 64-bit hash of the stream. Please note that this requires reading the stream and is therefore not thread safe. If thread safety is desired, make a clone of the stream first.

Public Member Functions inherited from IEDLStream

- virtual bool isValid () const =0

  Determine stream validity.
- virtual bool open ()

  Opens the stream.
- virtual void **openE** ()=0

  As per open(), but will throw an exception on failure (IEDLError) that for some stream types may contain additional failure information.
- virtual void **close** ()=0

  Closes the stream.
- virtual int64 getPos ()=0

  Get current stream position.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IDOMHashable

- virtual ~**IDOMHashable** ()

  Virtual destructor.
- virtual uint64 hashE ()

  As hash(), but throws an exception if the operation fails.

Public Member Functions inherited from IRAStream

- virtual ~**IRAStream** ()

  Virtual destructor.
- virtual int64 length ()=0

  Get length of the stream.
- virtual bool setPos (int64 newPos)=0

  Set stream position.
- virtual void setPosE (int64 newPos)

  Set stream position, but throw an exception on failure.

Static Public Member Functions inherited from IInputStream

- static EDL_API IRAInputStreamPtr createFromFile (IEDLClassFactory *pFactory, const EDLSysString &path)

  Creation function for an IInputStream for a file on disk. Throws an IEDLError exception on failure.
- static EDL_API IRAInputStreamPtr createFromFile (IEDLClassFactory *pFactory, const EDLString &path)

  Creation function for an IInputStream for a file on disk. Throws an IEDLError exception on failure.
- static EDL_API IRAInputStreamPtr createFromFileShared (IEDLClassFactory *pFactory, const EDLSys↩ String &path)

  Creation function for an IInputStream for a file on disk. Similar to createFromFile, but if this file is cloned, only one file handle will be open at a time. This is slower, but is useful if many users are likely to want to have this file open at the same file. On some systems, having too many files open may result in errors. Performance should be acceptable providing there is not a great deal of contention and simultaneous access.
- static EDL_API IRAInputStreamPtr createFromFileShared (IEDLClassFactory *pFactory, const EDLString &path)

  Creation function for an IInputStream for a file on disk. Similar to createFromFile, but if this stream is cloned, only one file handle will be open at a time. This is slower, but is useful if many users are likely to want to have this file open at the same file. On some systems, having too many files open may result in errors. Performance should be acceptable providing there is not a great deal of contention and simultaneous access.
- static EDL_API IRAInputStreamPtr createSharedFromStream (IEDLClassFactory *pFactory, const IRAInputStreamPtr &stream)

  Creation function for a shared stream overlaying an existing stream. If this stream is cloned, the underlying file will not be cloned. This is useful if many users are likely to want to have this file open at the same time. On some systems, having too many files open may result in errors. Performance should be acceptable providing there is not a great deal of contention and simultaneous access. Only possible for random access streams. If the stream is already shared, the stream will be returned as is.

- static EDL_API IRAInputStreamPtr createFromMemory (IEDLClassFactory *pFactory, const void *mem, uint32 length, bool copy=false, bool free=true)

  Creation function for an IInputStream for data in memory. Throws an IEDLError exception on failure.
- static EDL_API IInputStreamPtr createFromUserReadFunc (IEDLClassFactory *pFactory, UserStream↩ ReadFunc readFunc, void *priv)

  Creation function for an IInputStream from a user function that provides data. Throws an IEDLError exception on failure.
- static EDL_API IRAInputStreamPtr createFromRAUserFunc (IEDLClassFactory *pFactory, int64 length, UserRAReadFunc readFunc, void *priv)

  Creation function for an IRAInputStream from a user function that provides random access. Throws an IEDLError exception on failure.
- static EDL_API IRAInputStreamPtr createRandomAccessFromNonRandomAccess (IEDLClassFactory *p↩ Factory, const IInputStreamPtr &stream)

  Creation function to make a random access stream from a stream that is not random access. Currently, the temporary store will be used to store a copy of the source data, but a concrete implementation method should not be assumed. If the source stream is random access, it will be returned as is. Throws an IEDLError exception on failure.
- static EDL_API IRAInputStreamPtr createFromNewFileWithContents (IEDLClassFactory *pFactory, const EDLSysString &path, const IInputStreamPtr &stream)

  Creation function for an IInputStream for a file on disk created with the contents of an existing stream. Throws an IEDLError exception on failure.
- static EDL_API IRAInputStreamPtr createFromNewFileWithContents (IEDLClassFactory *pFactory, const EDLString &path, const IInputStreamPtr &stream)

  Creation function for an IInputStream for a file on disk created with the contents of an existing stream. Throws an IEDLError exception on failure.
- static EDL_API IInputStreamPtr createSubFile (IEDLClassFactory *pFactory, const IInputStreamPtr &stream, int64 offset, int64 length)

  Creation routine for a stream representing a portion of a file on disk. If the source file is random access, then the created file shall be also. Throws an IEDLError exception on failure.
- static EDL_API IInputStreamPtr createFromFlateCompressed (IEDLClassFactory *pFactory, const IInput↩ StreamPtr &stream, bool raw=true, bool ignoreChecksums=false)

  Creation routine for a input stream for decompressing a flate stream. Throws an IEDLError exception on failure.
- static EDL_API IInputStreamPtr createFromLz4Compressed (IEDLClassFactory *pFactory, const IInput↩ StreamPtr &stream)

  Creation routine for a input stream for decompressing an lz4 block compressed stream. Throws an IEDLError exception on failure. Note: This is not intended for interoperability with other LZ4 formats, but is useful for things like temporary storage.
- static EDL_API IInputStreamPtr createFromPredictorCompressed (IEDLClassFactory *pFactory, const IInputStreamPtr &stream, uint8 predictor, uint8 colors, uint8 bitsPerComponent, uint32 columns)

  Creation routine for a input stream for applying a predictor algorithm to a compressed stream. Throws an IEDLError exception on failure.
- static EDL_API IInputStreamPtr createCompositeStream (IEDLClassFactory *pFactory, const CIInput↩ StreamVect &streams)

  Creation routine for creating a composite input stream representing the concatenation of a series of streams.
- static EDL_API IInputPushbackStreamPtr createPushbackStream (IEDLClassFactory *pFactory, const IInputStreamPtr &sourceStream, bool clone=true)

  Creation routine for creating a push-back stream using a non-pushback stream as a data source. If the source stream is random access, the resulting stream will also be random access.
- static EDL_API IInputStreamPtr createUelStream (IEDLClassFactory *pFactory, const IInputStreamPtr &sourceStream)

  Creation routine for a stream that emulates an end-of-file condition should a Universal End of Language (UEL) sequence be encountered in an underlying data source. The resulting stream is not random access. The underlying stream is neither opened nor closed. This stream type is useful for restricting parsing of substreams inside a print stream. Streams of this type are not clonable. Please also note, that this stream must read ahead of read requests in order to search for UEL sequences. However it will never read beyond the UEL sequence.
- static EDL_API IInputStreamPtr createTbcpStream (IEDLClassFactory *pFactory, const IInputStreamPtr &sourceStream, bool openSource=false)

Creation routine for a stream that reads data from an underlying stream encoded using Adobe's Tagged Binary Communication Protocol (TBCP). The resulting stream is not random access. Streams of this type are not clonable.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.399.1 Detailed Description

Random Access Input Stream.

The documentation for this class was generated from the following file:

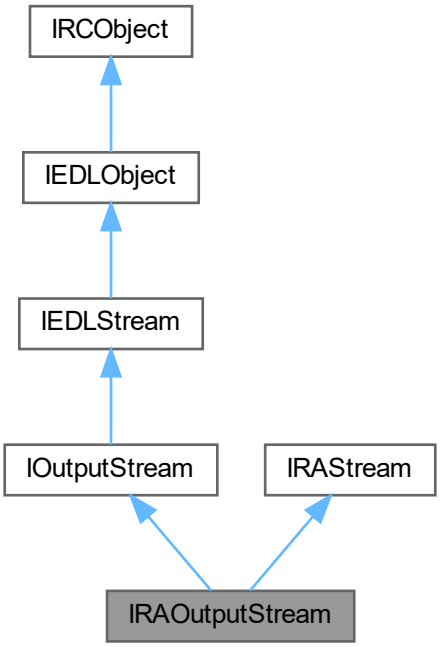
- edlstream.h

7.400 IRAOutputStream Class Reference

Random Access Output Stream.

`#include <edlstream.h>`

Inheritance diagram for IRAOutputStream:



Additional Inherited Members

Public Member Functions inherited from IOutputStream

- virtual int32 write (const char *str)

  Perform a write.
- virtual int32 writeFormatted (const char *fmt,...)

  Perform a formatted write as per fprintf().
- virtual void writeFormattedE (const char *fmt,...)

  As writeFormatted(), but throws an exception if the operation fails.
- virtual bool completeWrite (const void *buffer, int32 count)

  Perform a complete write.
- virtual bool completeWrite (const char *str)

  Perform a complete write.
- virtual void completeWriteE (const void *buffer, int32 count)

  As completeWrite(), but throws an exception if the operation fails.
- virtual void completeWriteE (const char *str)

  As completeWrite(), but throws an exception if the operation fails.

Public Member Functions inherited from IEDLStream

- virtual bool isValid () const =0

  Determine stream validity.
- virtual bool open ()

  Opens the stream.
- virtual void **openE** ()=0

  As per open(), but will throw an exception on failure (IEDLError) that for some stream types may contain additional failure information.
- virtual void **close** ()=0

  Closes the stream.
- virtual int64 getPos ()=0

  Get current stream position.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Public Member Functions inherited from IRAStream

- virtual ~**IRAStream** ()

  Virtual destructor.
- virtual int64 length ()=0

  Get length of the stream.
- virtual bool setPos (int64 newPos)=0

  Set stream position.
- virtual void setPosE (int64 newPos)

  Set stream position, but throw an exception on failure.

Static Public Member Functions inherited from IOutputStream

- static EDL_API IRAOutputStreamPtr createToFile (IEDLClassFactory *pFactory, const EDLSysString &path, bool append=false)

  Creation function for an IOutputStream for a file on disk. Throws an IEDLError exception on failure.
- static EDL_API IRAOutputStreamPtr createToFile (IEDLClassFactory *pFactory, const EDLString &path, bool append=false)

  Creation function for an IOutputStream for a file on disk. Throws an IEDLError exception on failure.
- static EDL_API IOutputStreamPtr createFromUserWriteFunc (IEDLClassFactory *pFactory, UserStream↩ WriteFunc writeFunc, void *priv)

  Creation function for an IOutputStream from a user function that provides data. Throws an IEDLError exception on failure.
- static EDL_API IOutputStreamPtr createToFlateCompressed (IEDLClassFactory *pFactory, const IOutput↩ StreamPtr &stream, uint32 compressionLevel, bool raw=true)

  Creation routine for an output stream for compressing a flate stream. Throws an IEDLError exception on failure.
- static EDL_API IOutputStreamPtr createToLz4Compressed (IEDLClassFactory *pFactory, const IOutput↩ StreamPtr &stream, bool openSourceStream=true)

  Creation routine for an output stream for compressing an lz4 stream. Throws an IEDLError exception on failure. Note: This is not intended for interoperability with other LZ4 formats, but is useful for things like temporary storage.
- static EDL_API int64 copy (const IInputStreamPtr &inStream, const IOutputStreamPtr &outStream)

  Copy a source stream to a destination stream. Opens and closes both the input and output streams. Throws an IEDLError exception on failure.
- static EDL_API int64 writeStream (const IInputStreamPtr &inStream, const IOutputStreamPtr &outStream)

  Write the contents of the given stream to an output stream. Opens and closes the input, but does not open or close the output. Throws an IEDLError exception on failure.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.400.1 Detailed Description

Random Access Output Stream.

The documentation for this class was generated from the following file:

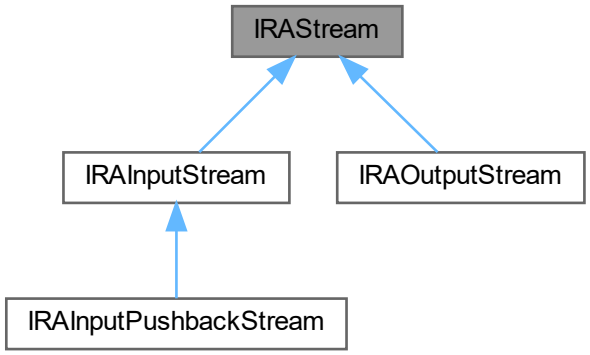
- edlstream.h

7.401 IRAStream Class Reference

Abstract base class for "Random-Access" streams i.e. streams that can be arbitrarily re-positioned.

`#include <edlstream.h>`

Inheritance diagram for IRAStream:



Public Member Functions

- virtual ~**IRAStream** ()
  Virtual destructor.
- virtual int64 length ()=0
  Get length of the stream.
- virtual bool setPos (int64 newPos)=0
  Set stream position.
- virtual void setPosE (int64 newPos)
  Set stream position, but throw an exception on failure.

7.401.1 Detailed Description

Abstract base class for "Random-Access" streams i.e. streams that can be arbitrarily re-positioned.

7.401.2 Member Function Documentation

length()

`virtual int64 IRAStream::length ( )  [pure virtual]`

Get length of the stream.

Returns

int64 The length of the stream.

setPos()

```
virtual bool IRAStream::setPos (
            int64 newPos )  [pure virtual]
```

Set stream position.

Parameters

newPos	The desired new position.

Returns

bool True if method succeeded.

setPosE()

```
virtual void IRAStream::setPosE (
            int64 newPos )  [inline], [virtual]
```

Set stream position, but throw an exception on failure.

Parameters

newPos	The desired new position.

The documentation for this class was generated from the following file:

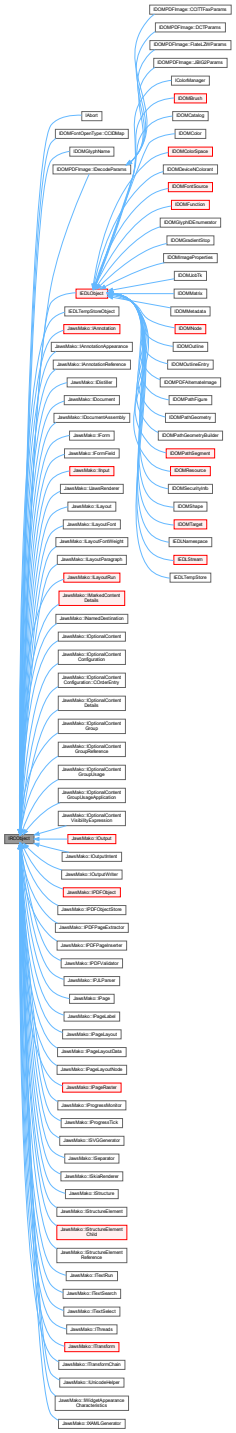
- edlstream.h

7.402 IRCObject Class Reference

Base class Interface for all Reference Counted objects.

```
#include <ircobject.h>
```

Inheritance diagram for IRCObject:



Public Member Functions

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.

- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.

- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Protected Member Functions

- virtual ~**IRCObject** ()

  Virtual destructor.

7.402.1 Detailed Description

Base class Interface for all Reference Counted objects.

7.402.2 Member Function Documentation

decRef()

```
virtual bool IRCObject::decRef ( ) const  [pure virtual]
```

Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.

Returns

bool Returns true on success

getRefCount()

```
virtual int32 IRCObject::getRefCount ( ) const  [pure virtual]
```

Retrieve the current reference count of the actual object pointed to.

Returns

int32 The current reference count

The documentation for this class was generated from the following file:

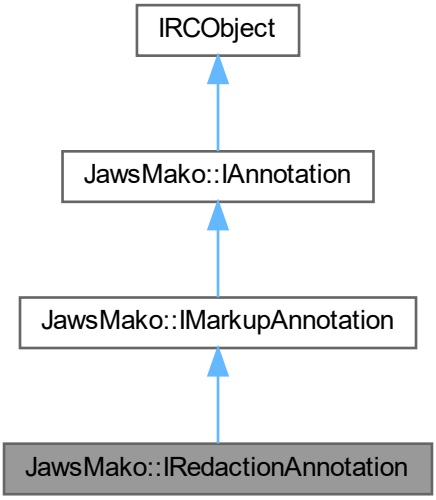
- ircobject.h

7.403 JawsMako::IRedactionAnnotation Class Reference

A generic interface class for a redaction annotation.

`#include <interactive.h>`

Inheritance diagram for JawsMako::IRedactionAnnotation:



Public Member Functions

- virtual CQuadPointVect getQuadPoints () const =0

  Get the redaction annotation's quad points if present. The points are relative to the annotation rect.
- virtual void setQuadPoints (const CQuadPointVect &quadPoints)=0

  Set the redaction annotation's quad points.
- virtual IDOMColorPtr getInteriorColor () const =0

  Get the interior color of the fill used for the shape.
- virtual void setInteriorColor (const IDOMColorPtr &color)=0

  Set the interior color to be used to fill the line endings.

Public Member Functions inherited from JawsMako::IMarkupAnnotation

- virtual U8String getAuthor () const =0

  Get the Author of the markup annotation. This is the "T" entry in the annotation, which by convention is the author.
- virtual void setAuthor (const U8String &author)=0

  Set the Author of the markup annotation. This is the "T" entry in the annotation, which by convention is the author.
- virtual IEDLTimePtr getCreationTime () const =0

  Get the creation date and time of the annotation, if present.
- virtual void setCreationTime (const IEDLTimePtr &creationTime)=0

  Set the creation date and time of the annotation.

- virtual float getOpacity () const =0

  Get the opacity of the markup annotation.
- virtual void setOpacity (float opacity)=0

  Set the opacity of the markup annotation.
- virtual IAnnotationReferencePtr getPopupReference () const =0

  Get a reference to the popup, if present.
- virtual void setPopup (const IAnnotationReferencePtr &popupReference)=0

  Set or clear the popup, by reference.
- virtual void setPopup (const IPopupAnnotationPtr &popup)=0

  Set a reference to a popup, if present.
- virtual IAnnotationAppearancePtr createNormalAppearance () const =0

  Create a basic appearance given the current annotation's parameters. This can then be installed by using IAnnotation::addAppearance().

Public Member Functions inherited from JawsMako::IAnnotation

- virtual eAnnotationType getType () const =0

  Get the type of the annotation.
- virtual const FRect & getRect () const =0

  Get the rect in which the appearances should be displayed.
- virtual void setRect (const FRect &rect)=0

  Set the rect in which the appearances should be displayed.
- virtual U8String getContents () const =0

  Get the Contents entry in UTF-8.
- virtual void setContents (const U8String &contents)=0

  Set the Contents entry in UTF-8.
- virtual IDOMColorPtr getColor () const =0

  Get the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.
- virtual void setColor (const IDOMColorPtr &color)=0

  Set the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.
- virtual IEDLTimePtr getModificationTime () const =0

  Get the Modification date and time of the annotation, if present.
- virtual void setModificationTime (const IEDLTimePtr &modificationTime)=0

  Set the Modification date and time of the annotation.
- virtual CAnnotationBorder getBorder () const =0

  Get the annotation's border. See CAnnotationBorder for details.
- virtual void setBorder (const CAnnotationBorder &border)=0

  Set the annotation's border.
- virtual uint32 getFlags () const =0

  Get the annotation flags. To interpret these flags please see section 8.4.2 "Annotation Flags" in the PDF 1.7 specification.
- virtual void setFlags (uint32 flags)=0

  Set the annotation flags. Please see section 8.4.2 "Annotation Flags" for details about these flags.
- virtual void rotate (uint16 rotate, double pageWidth, double pageHeight)=0

  Rotate the annotation clockwise as if the page was rotated by the same amount.
- virtual CAnnotationAppearanceVect getAppearances () const =0

  Return all the annotation appearances in a vector.
- virtual void **removeAppearances** ()=0

  Remove all annotation appearances.
- virtual U8String getState () const =0

Get the current annotation state.

- virtual void setState (const U8String &state)=0

  Set the current annotation state.

- virtual IAnnotationAppearancePtr getAppearance (eAppearanceUsage usage, const U8String &state=U8String()) const =0

  Fetch the annotation appearance that should be used for the given annotation usage and state according to the following rules:

- virtual void addAppearance (const IAnnotationAppearancePtr &appearance)=0

  Add or replace an appearance.

- virtual bool hasNormalAppearance () const =0

  Does the annotation have a normal appearance? Convenience utility function.

- virtual IAnnotationPtr clone () const =0

  Clone the annotation. This is a deep clone. The annotation reference will remain the same.

- virtual bool matchesReference (const IAnnotationReferencePtr &reference) const =0

  Does this annotation match the given IAnnotationReference?

- virtual IAnnotationReferencePtr getReference () const =0

  Get a reference that can be used to refer to this annotation.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.

- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.

- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IRedactionAnnotationPtr create (const IJawsMakoPtr &jawsMako, const FRect &rect, const CQuadPointVect &quadPoints=CQuadPointVect(), const IDOMColorPtr &color=IDOMColorPtr(), const IDOMColorPtr &interiorColor=IDOMColorPtr())

  Create a redaction annotation.

Additional Inherited Members

Public Types inherited from JawsMako::IAnnotation

- enum eAnnotationType {
  eAT3D , eATCaret , eATCircle , eATFileAttachment ,
  eATFreeText , eATHighlight , eATInk , eATLine ,
  eATLink , eATMovie , eATPolygon , eATPolyLine ,
  eATPopup , eATPrinterMark , eATProjection , eATRedact ,
  eATRichMedia , eATScreen , eATSound , eATSquare ,
  eATSquiggly , eATStamp , eATStrikeOut , eATText ,
  eATTrapNet , eATUnderline , eATWatermark , eATWidget ,
  eATOther }

  Types of annotations, listed with the earliest version of PDF that supported them.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.403.1 Detailed Description

A generic interface class for a redaction annotation.

7.403.2 Member Function Documentation

create()

```
static JAWSMAKO_API IRedactionAnnotationPtr JawsMako::IRedactionAnnotation::create (
            const IJawsMakoPtr & jawsMako,
            const FRect & rect,
            const CQuadPointVect & quadPoints = CQuadPointVect(),
            const IDOMColorPtr & color = IDOMColorPtr(),
            const IDOMColorPtr & interiorColor = IDOMColorPtr() )  [static]
```

Create a redaction annotation.

Parameters

jawsMako	The JawsMako instance
rect	The annotations bounds. Must not be empty.
quadPoints	The quadrilaterals defining the redaction area. May be empty, in which case the annotation rect is used.
color	The outline color for the annotation when viewed in the user interface. Pass NULL for the default black
interiorColor	The color used to fill the annotation when it is applied. An empty colour will result in white being used. The color, if provided, must use the DeviceRGB color space.

Returns

IRedactionAnnotationPtr A smart pointer to the new redaction annotation

getInteriorColor()

```
virtual IDOMColorPtr JawsMako::IRedactionAnnotation::getInteriorColor ( ) const  [pure virtual]
```

Get the interior color of the fill used for the shape.

Returns

IDOMColorPtr The interior color, or NULL if there is no such color

getQuadPoints()

```
virtual CQuadPointVect JawsMako::IRedactionAnnotation::getQuadPoints ( ) const  [pure virtual]
```

Get the redaction annotation's quad points if present. The points are relative to the annotation rect.

Returns

CQuadPointVect The annotation's quad points

setInteriorColor()

```
virtual void JawsMako::IRedactionAnnotation::setInteriorColor (
            const IDOMColorPtr & color )  [pure virtual]
```

Set the interior color to be used to fill the line endings.

Parameters

color	The desired color. Pass NULL to remove the color (default is white). If a color is passed, it must use the DeviceRGB color space.

setQuadPoints()

```
virtual void JawsMako::IRedactionAnnotation::setQuadPoints (
            const CQuadPointVect & quadPoints )  [pure virtual]
```

Set the redaction annotation's quad points.

Parameters

quadPoints	The annotation's desired quad points

The documentation for this class was generated from the following file:

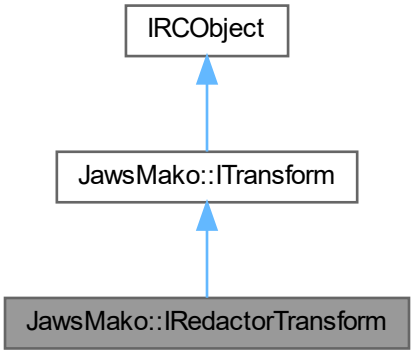
- interactive.h

7.404 JawsMako::IRedactorTransform Class Reference

A transform for applying redaction redactions.

```
#include <transforms.h>
```

Inheritance diagram for JawsMako::IRedactorTransform:



Public Member Functions

- virtual void **setTargetSpace** (const IDOMColorSpacePtr &targetSpace)=0

  Set the color space to use when rendering redacted sections. The default is DeviceRGB. The color space must be supported by the IRendererTransform.
- virtual void **setResolution** (uint32 resolution)=0

  Set the target resolution to use when rendering redacted sections. The default is 300dpi. The resolution must be supported by the IRendererTransform.
- virtual void **setGenerateMasks** (bool generateMasks)=0

  Set whether masks should be generated when rendering redacted sections. The default is true.

Public Member Functions inherited from JawsMako::ITransform

- virtual IDOMBrushPtr transform (const IDOMBrushPtr &brush, eBrushUsage usage=eBUGeneral, const CTransformState &state=CTransformState())=0

  Apply the transform to the given brush, if applicable. These transforms are thread safe.
- virtual IDOMImagePtr transform (const IDOMImagePtr &image, const CTransformState &state=CTransformState())=0

  Apply the transform to the given image, if applicable. These transforms are thread safe.
- virtual IDOMColorPtr transform (const IDOMColorPtr &color, const CTransformState &state=CTransformState())=0

  Apply the transform to the given color, if applicable. These transforms are thread safe.
- virtual IDOMColorSpacePtr transform (const IDOMColorSpacePtr &colorSpace, const CTransformState &state=CTransformState())=0

  Apply the transform to the given colorspace, if applicable. These transforms are thread safe.
- virtual IDOMNodePtr transform (const IDOMNodePtr &node, bool &changed, bool transformChildren=true, const CTransformState &state=CTransformState())=0

  Apply the transform to the given node, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time.
- virtual void transformPage (const IPagePtr &page, bool transformContent=true, bool transform↩Annotations=true)=0

  Apply the transform to the given page, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time. The transform will also apply to the annotations appearances.
- virtual void **flushCaches** ()=0

Flush the caches used by the transform. Most transforms cache recently transformed results to improve the performance of repeated transformations of equivalent results. However, it is possible that some cached results may point to entities that no longer exist, such as content inside an XPS file that no longer exists. If you are deleting or replacing files where transforms have been used, it is advisable to invoke this routine to clear the caches.

- virtual void setProgressMonitor (const IProgressMonitorPtr &progressMonitor)=0

  Set the IProgressMonitor object for this transform to allow for monitoring the progress of the transform.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IRedactorTransformPtr create (const IJawsMakoPtr &jawsMako, const IAbortPtr &abort=IAbortPtr())

  Create the transform.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.404.1 Detailed Description

A transform for applying redaction redactions.

```
This transform uses rendering to render the area covered by each
redaction annotation including that annotation into an image, and
removing any objects underneath that intersect with the annotation
from the DOM tree entirely.
This transform is only effective for page content, not for
annotations and other off-page graphical content.

As it requires knowledge of the annotations, this transform may
only be applied to IPage objects; anything else will result in an
exception.

As such only transformPage() can be used and any other transform()
routine will throw an exception. If the transformContent argument to
transformPage() is not true the transform will do nothing. Further,
the transformAnnotations parameter will be ignored.

The redaction annotations are removed after they are applied.
```

7.404.2 Member Function Documentation

create()

```
static JAWSMAKO_API IRedactorTransformPtr JawsMako::IRedactorTransform::create (
            const IJawsMakoPtr & jawsMako,
            const IAbortPtr & abort = IAbortPtr() )  [static]
```

Create the transform.

Parameters

jawsMako	The JawsMako instance.
abort	An abort callback handler.

Returns

The new instance.

The documentation for this class was generated from the following file:

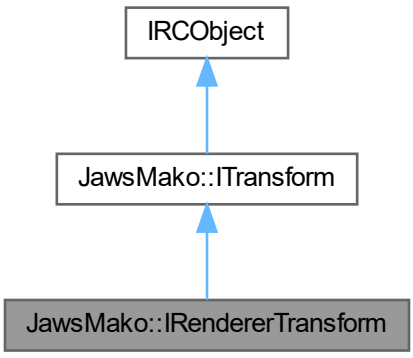
- transforms.h

7.405 JawsMako::IRendererTransform Class Reference

A transform for selective rendering of sections of a DOM tree, replacing the rendered items with an image representation. Currently only operates on IDOMFixedPages; this restriction should be eased in future versions.

```
#include <transforms.h>
```

Inheritance diagram for JawsMako::IRendererTransform:



Public Member Functions

- virtual bool **probe** (const IDOMNodePtr &node)=0

  Probe the node, checking to see if the renderer transform would have any effect on this node if transformed.
- virtual void **setResolution** (uint32 resolution)=0

  Sets the target resolution for rendering, in dpi. The default is 300dpi.
- virtual uint32 **getResolution** () const =0

  Get the target resolution for rendering.
- virtual void **setTargetSpace** (const IDOMColorSpacePtr &space)=0

Sets the target final color space for rendered content. scRGB, Indexed, DeviceN or LAB spaces are not allowed. Further limitations exist if monochrome mode is used; see the description of setMonochromeMode() for details. The default is DeviceRGB.

- virtual IDOMColorSpacePtr **getTargetSpace** () const =0

  Get the target color space for rendering.
- virtual void **setSWOPTargetSpace** ()=0

  Convenience; Set the target space to CMYK using the default SWOP profile. This will fail if monochrome mode is used; see the description of setMonochromeMode() for details.
- virtual void **setTargetProfile** (const IDOMICCProfilePtr &profile)=0

  Sets the target final color space for rendered content, using a profile.
- virtual void setEnableVectorMode (bool enable)=0

  Enable "vector" flattening mode.
- virtual void setUseImageResolutionForRenderingWherePossible (bool enable)=0

  Set whether or not areas with compatible images may be rendered at image resolution if possible when vector flatting mode is enabled (see setEnableVectorMode())
- virtual void setVectorAndTextResolution (uint32 resolution)=0

  Set the desired resolution for vector and textual content when vector flattening mode is enabled (see setEnableVectorMode()).
- virtual void setRasterFallbackThreshold (uint32 threshold)=0

  For cases where the vector mode is used, set the threshold at which to fall back to raster mode for a particular renderable area.
- virtual void setRasterFallbackResolution (uint32 resolution)=0

  Set the resolution to be used for areas where vector mode falls back to raster mode. That is, if an area being rendered is more complex than that specified by setRasterFallbackThreshold, this is the resolution that should be used.
- virtual void setVectorImageReuseCacheSize (uint32 reuseCacheSizeMB)=0

  Set whether or not the renderer, with vector mode enabled, should attempt to reused flattened sections from previously rendered areas, and if so, set the maximum size of the cache in megabytes.
- virtual void setEnableFormSnappingForVectorReuse (bool enable)=0

  Set whether or not the renderer, with vector mode amd vector image reuse enabled, should be allowed to align forms to pixel boundaries to improve caching reuse.
- virtual void setVectorAreaFormReuseCacheSize (uint32 formCount)=0

  Set whether or not the renderer, with vector mode enabled, should cache entire flattened sections as IDOMForm objects, and if so, set the maximum size of the cache (in items).
- virtual void setVectorAreaSourceReuseCacheSize (uint32 count)=0

  Set whether or not the renderer, with vector mode enabled, should attempt to cache entire flattened areas based on the source content, and if so, set the maximum size of the cache (in items).
- virtual void setMarkVectorFlattenedFontsForEmbedding (bool embed)=0

  Set whether or not fonts that are used for vector flattening should be marked for embedding.
- virtual void setFormReconstructionCacheSize (uint32 formCount)=0

  Set whether or not the renderer should attempt to reconstruct forms that it needed to unpack due to the rendering of content within the form, and if so, set the maximum size of the cache to use (in items).
- virtual void setRasterImageReuseCacheSize (uint32 reuseCacheSizeMB)=0

  Set whether or not the renderer should attempt to cache rendered results for raster rendering, and if so, set the maximum size of the cache, in megabytes.
- virtual void setMonochromeMode (bool enable=true)=0

  Sets whether or not to use monochrome output.
- virtual void setSpotHalftone (float frequency, bool useFullResolutionForFlattening=false)=0

  Set the simple spot haltone to be used when monochrome mode is enabled.
- virtual void setThresholdHalftone (uint32 width, uint32 height, const CThresholdArray &thresholds)=0

  Set a threshold array halftone to be used when monochrome mode is enabled.
- virtual void setPreserveSpots (const CSpotColorNames &preservedSpots)=0

  Provide a list of spots to retain in the output, if present. In the default case, no spots will be retained. This function will fail if monochrome mode is used; see the description of setMonochromeMode() for details.
- virtual void setPreserveAllSpots (bool preserve)=0

Alternatively preserve all spot colors in the output. The default is false. This function will fail if monochrome mode is used; see the description of setMonochromeMode() for details.

- virtual void setMergeSpots (const CSpotColorNames &mergedSpots)=0

  Merge the given spot components into the process components after rendering. This is only currently possible when rendering to a CMYK based process color space, and is ignored otherwise. Only affects rendered content. In the default case, no spots will be merged. This feature is especially useful for emulating overprint of spot components.

- virtual void setMergeAllSpots (bool merge)=0

  Alternatively merge all spot colors in the output. This is only currently possible when rendering to a CMYK based process color space, and is ignored otherwise. Only affects rendered content. In the default case, no spots will be merged. This feature is especially useful for emulating overprint of spot components.

- virtual void setDropSpots (const CSpotColorNames &droppedSpots)=0

  Provide a list of spots to drop from rendered content. In the default case, no spots will be dropped. Note that this feature does not affect content that is not rendered; please see IColorConverterTransform::setDeviceNHandling which can be used for that purpose. This function will fail if monochrome mode is used; see the description of setMonochromeMode() for details.

- virtual void setBlackPreservation (bool preserveForText, bool preserveForOther)=0

  Enable/control 100% black preservation.

- virtual void setGenerateMasks (bool generate)=0

  Set whether or not to generated masked images as the rendererd result. If true, the individual images generated as part of rendering will have a bilevel alpha channel providing a mask to just the silhouette of the dom nodes targettted for rendering. This generally results in better looking results at the intersection between rendered and non-rendered sections. The default is true.

- virtual void **setUseMultipleThreads** (bool useMultipleThreads)=0

  Set whether the transform should use multiple threads when rendering sections of the tree. When enabled, a global thread pool is used. The default is true.

- virtual void **setRenderObjectsIndividually** (bool renderIndividually)=0

  Set whether or not to render each renderable object into its own image. The default is false. Normally the renderer will attempt to produce a single image for overlapping or nearby objects in an attempt to reduce the number of images generated and improve performance. Setting this to true will instead produce a single image in the output for every object that would be rendered. If set to true, this overrides any setting made with setRenderOncePerRenderableArea().

- virtual void **setRenderOncePerRenderableArea** (bool renderOnce)=0

  Set whether or not to render just one image per renderable region. That is, if true, the renderer will not produce any overlapping rendered images in the result. The default is false. This setting is ignored if setRenderObjectsIndividually() has been set to true.

- virtual void **setMaximumImageAreaMultiple** (float limit)=0

  Set the maximum amount of allowed overdraw in the rendered results. Ordinarily the renderer attempts to retain non-renderable content in its original form, which can require rendering to multiple images. For cases where the dom tree consists of highly layered combinations of renderable and non-renderable content, the amount of image data generated can be very high. This provides a method of putting a cap on the amount of image data that can be generated. The parameter is a threshold in units of multiples of the entire candidate area. That is if a value of 5 is specified, and in ordinary circumstances rendering would generate more than 5 times the candidate area, then the renderer will render the results in a single image.

- virtual void setMaximumRenderedResultPixels (uint64 maximum)=0

  Set the maximum size (in pixels) of a rendered result.

- virtual void setGenerateFlateCompressedPDFImages (bool enable)=0

  Set whether or not the images generated by the renderer transform should be compressed with Flate as IDOMPDFImage objects. Currently this setting is only honored when vector mode is enabled.

- virtual void setMinimumFlateCompressedImageSize (uint32 minSizePixels)=0

  Set the minimum size, in pixels, of images that would be compressed with Flate according to setGenerateFlateCompressedPDFImages.

- virtual void **renderNodesWithRenderFlagSet** (bool render)=0

  Set whether nodes with the IDOMNodeFlags::eNodeRenderFlag set should be rendered. This is useful if arbitrary objects need to be marked for rendering within a tree, or if none of the rules below are a good fit. The default is false.

- virtual void **setShouldRenderCallback** (ShouldRenderNodeFunc func, void *priv)=0

  Provides a callback function, called for each node that is not chosen for rendering according to any of the other rules. This function returns true if the node should be rendered. Perhaps simpler in some situations than applying the marker flag externally and using renderNodesWithRenderFlagSet(true). priv is passed to the callback function for every call. The function can also interrogate the children siblings and parent of the node.

- virtual void **renderTransparentNodes** (bool render)=0

  Set whether transparent objects should be renderered. Useful for blanket flattening of all transparency in a DOM tree. The default is false.
- virtual void setRenderTransparentNodesOnPageGroupMismatch (bool render)=0

  Set whether to render transparent nodes if the page blending group color space differs from the rendering target color space.
- virtual void **renderNonNormalBlendedNodes** (bool render)=0

  Set whether nodes using a blend mode other than eBlendModeNormal should be rendered. Useful for preparing PDF content for consumers that do not support these blend modes, such as XPS. The default is false.
- virtual void **renderNonCanvasTransparencyGroups** (bool render)=0

  Set whether transparency groups that cannot be represented as a canvas should be rendered. The default is false.
- virtual void **renderMaskedImages** (bool render)=0

  Set whether images with boolean mask channels should be rendered. The default is false.
- virtual void **renderOpacityMaskedNodes** (bool render)=0

  Set whether nodes with opacity masks should be rendered. The default is false.
- virtual void **renderBrush** (IDOMBrush::eBrushType brushType, bool render=true)=0

  Set whether the brush of the given type should be rendered. In all cases, the default is false.
- virtual void **ignoreBrushWhileScanning** (IDOMBrush::eBrushType brushType, bool ignore=true)=0

  Set whether the brush of the given type should be ignored when scanning for items to render. If true, brushes of that type will not trigger rendering. In all cases the default is false.
- virtual void **renderComplexType3ShadingPatterns** (bool render)=0

  Set whether to render Type 3 shading pattern brushes that cannot currently be converted to a simpler form by IDOMShadingPatternType3Brush::getEquivalentSimpleBrush(). The default is false.
- virtual void **renderComplexTilingPatterns** (bool render)=0

  Set whether to render tiling pattern brushes that cannot currently be converted to a visual brush form by IDOMTilingPatternBrush::getEquivalentVisualBrush() such as self-intersecting patterns. The default is false.
- virtual void **renderVisualBrushesWithViewPortLargerThan** (double val)=0

  Set whether visual brushes with a ViewPort brushes above a certain size should be rendered. Some consumers cannot reliably handle large visual brushes, and this provides a method of dealing with this. The default is 0.0, which disables this feature.
- virtual void **renderConeGradients** (bool render)=0

  Set whether radial gradients with a cone shape should be rendererd. The default is false.
- virtual void **renderGlyphsWithStyleSimulation** (bool render)=0

  Set whether glyphs with style simulations should be rendered. The default is false.
- virtual void **renderGlyphsWithRestrictedFont** (bool render)=0

  Set whether glyphs using a restricted font should be rendered. That is, a font which has bit 1 of the OS/2 fsType field set, or are marked as bitmap only. The default is false.
- virtual void **renderGlyphsWithNonSubsettableFont** (bool render)=0

  Set whether glyphs using a font that is not allowed to be subset (according to its OS/2 fsType field) should be rendered. The default is false.
- virtual void **renderGlyphsWithNonEditableFont** (bool render)=0

  Set whether glyphs using a font that is not allowed in an edited document (according to its OS/2 fsType field) should be rendered. The default is false.
- virtual void **renderGlyphsWithType3Font** (bool render)=0

  Set whether glyphs using a Type 3 font should be rendered. The default is false.
- virtual void **enableTrueTypeNotDef** (bool enable)=0

  Set whether a TrueType font's .notdef glyph should be rendered. Note that this does not control whether or not a containing glyphs node is actually rendered. However, if a glyphs node is required to be rendered, and a TrueType notdef glyph is required, this controls whether or not the notdef glyph will be present in the rendered result. The default is false.

Public Member Functions inherited from JawsMako::ITransform

- virtual IDOMBrushPtr transform (const IDOMBrushPtr &brush, eBrushUsage usage=eBUGeneral, const CTransformState &state=CTransformState())=0

  Apply the transform to the given brush, if applicable. These transforms are thread safe.
- virtual IDOMImagePtr transform (const IDOMImagePtr &image, const CTransformState &state=CTransformState())=0

  Apply the transform to the given image, if applicable. These transforms are thread safe.
- virtual IDOMColorPtr transform (const IDOMColorPtr &color, const CTransformState &state=CTransformState())=0

  Apply the transform to the given color, if applicable. These transforms are thread safe.
- virtual IDOMColorSpacePtr transform (const IDOMColorSpacePtr &colorSpace, const CTransformState &state=CTransformState())=0

  Apply the transform to the given colorspace, if applicable. These transforms are thread safe.
- virtual IDOMNodePtr transform (const IDOMNodePtr &node, bool &changed, bool transformChildren=true, const CTransformState &state=CTransformState())=0

  Apply the transform to the given node, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time.
- virtual void transformPage (const IPagePtr &page, bool transformContent=true, bool transform↩ Annotations=true)=0

  Apply the transform to the given page, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time. The transform will also apply to the annotations appearances.
- virtual void **flushCaches** ()=0

  Flush the caches used by the transform. Most transforms cache recently transformed results to improve the performance of repeated transformations of equivalent results. However, it is possible that some cached results may point to entities that no longer exist, such as content inside an XPS file that no longer exists. If you are deleting or replacing files where transforms have been used, it is advisable to invoke this routine to clear the caches.
- virtual void setProgressMonitor (const IProgressMonitorPtr &progressMonitor)=0

  Set the IProgressMonitor object for this transform to allow for monitoring the progress of the transform.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IRendererTransformPtr create (const IJawsMakoPtr &jawsMako, const IAbortPtr &abort=IAbortPtr())

  Create the transform.
- static JAWSMAKO_API CInkInfoVect **findInks** (const IJawsMakoPtr &jawsMako, const IDOMNodePtr &nodeTree, const IAbortPtr &abort=IAbortPtr())

  Find all the inks used in the given DOM node tree. Utility. Also makes an attempt to determine the color value of the colorant.
- static JAWSMAKO_API CInkInfoVect **findInksInColorSpace** (const IJawsMakoPtr &jawsMako, const IDOMColorSpacePtr &colorSpace)

  Find all the inks we can determine from the given color space. Utility. Also makes an attempt to determine the color value of the colorant.
- static JAWSMAKO_API IDOMColorSpaceDeviceN::CColorantInfoVect inkInfoToColorantInfo (const IJaws↩ MakoPtr &jawsMako, const CInkInfoVect &inkInfo, const IDOMColorSpacePtr &processSpace)

  Static utility to convert a CInkInfoVect into DeviceN-compatible CColorantInfo vector.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.405.1 Detailed Description

A transform for selective rendering of sections of a DOM tree, replacing the rendered items with an image representation. Currently only operates on IDOMFixedPages; this restriction should be eased in future versions.

Useful for rendering sections of the DOM that could not be consumed by a consumer any other way than as an image. For example, this transform is used to flatten objects with certain PDF transparency attributes before generating XAML or XPS output.

7.405.2 Member Function Documentation

create()

```
static JAWSMAKO_API IRendererTransformPtr JawsMako::IRendererTransform::create (
            const IJawsMakoPtr & jawsMako,
            const IAbortPtr & abort = IAbortPtr() )  [static]
```

Create the transform.

Parameters

jawsMako	The JawsMako instance.
abort	An abort callback handler.

Returns

The new instance.

inkInfoToColorantInfo()

```
static JAWSMAKO_API IDOMColorSpaceDeviceN::CColorantInfoVect JawsMako::IRendererTransform↩
::inkInfoToColorantInfo (
            const IJawsMakoPtr & jawsMako,
            const CInkInfoVect & inkInfo,
            const IDOMColorSpacePtr & processSpace )  [static]
```

Static utility to convert a CInkInfoVect into DeviceN-compatible CColorantInfo vector.

Parameters

jawsMako	The Mako instance to use.
inkInfo	The ink info to convert.
processSpace	The process color space to use for the components.

Returns

IDOMColorSpaceDeviceN::CColorantInfoVect The color component vector.

setBlackPreservation()

```
virtual void JawsMako::IRendererTransform::setBlackPreservation (
            bool preserveForText,
            bool preserveForOther )  [pure virtual]
```

Enable/control 100% black preservation.

```
100% black preservation preserves a 100% black output when rendering/
flattening when enabled for certain solid colors.

A 100% black object is an object painted with one of the following
colors:

- 0 0 0 1 in any CMYK color space (including ICC)
- 0 0 0 in any RGB color space (including ICC)
- 0 in any Gray colorspace (including ICC)
- 1 in a single channel DeviceN (aka Separation) Black color space
- 1 in a multi-channel DeviceN colorspace where only the Black channel
  is set.

If black preservation applies then the resulting color will be either
(depending on the target color space):

- c m y 1 in any CMYK colorspace (including ICC)
- 0 0 0 in any RGB colorspace (including ICC)
- 0 in any Gray colorspace (including ICC)

Note that the CMY values in a CMYK result may not be zero if color
managed white conversion would not produce pure white.

Note also that where transparency is involved this conversion only
applies when converting to the blending colorspace, and has no
further effect for further conversions.

May be enabled separately for text and vector art. Applies to only
solid color brushes (IDOMSolidColorBrush) and uncolored tiling
patterns only.

The default is set by #IColorManager::setDefaultBlackPreservation(),
which defaults to false, false. The default is applied at the time
of transform creation.
```

Parameters

preserveForText	Whether or not to apply black preservation for text.
preserveForOther	Whether or not to apply for non-text objects.

setDropSpots()

```
virtual void JawsMako::IRendererTransform::setDropSpots (
            const CSpotColorNames & droppedSpots )  [pure virtual]
```

Provide a list of spots to drop from rendered content. In the default case, no spots will be dropped. Note that this feature does not affect content that is not rendered; please see IColorConverterTransform::setDeviceNHandling

which can be used for that purpose. This function will fail if monochrome mode is used; see the description of setMonochromeMode() for details.

Please note that if any spot colorant is subject to setPreserveSpots, setPreserveAllSpots, setMergeSpots, setMergeAllSpots and/or setDropSpots, then the following rules apply in order:

- If the colorant is subject to setDropSpots, it is dropped, otherwise
- If the colorant is subject to setPreserveSpots() (not setPreserveAllSpots()!) it will be preserved (regardless of merge settings), otherwise
- If the colorant is subject to setMergeAllSpots() or setMergeSpots(), it will be merged, otherwise
- If setPreserveAllSpots() has been set to true, it will be preserved.

setEnableFormSnappingForVectorReuse()

```
virtual void JawsMako::IRendererTransform::setEnableFormSnappingForVectorReuse (
            bool enable )  [pure virtual]
```

Set whether or not the renderer, with vector mode amd vector image reuse enabled, should be allowed to align forms to pixel boundaries to improve caching reuse.

This allows better reuse if forms containing rendered content are repeated at different locations on the page (but with the same scale).

The default is false, meaning that no alignment is attempted.

setEnableVectorMode()

```
virtual void JawsMako::IRendererTransform::setEnableVectorMode (
            bool enable )  [pure virtual]
```

Enable "vector" flattening mode.

```
    NOTE: This is functionality is beta quality only at this time.

    With this mode set, the renderer will attempt to maintain vector
    and textual information as much as possible in its results, and
    may not need to render some objects at all.

    However, for very complicated cases this may result in a large
    amount of output geometry that could result in larger output
    files or files that process more slowly than an equivalent
    raster render.

    If monochrome mode is enabled, this setting is completely ignored.

    The default is false.

    Currently, setting this parameter to true will result in the
    following parameters to have different semantics:

    - setRenderOncePerRenderableArea() (will be forced to false)
    - renderMaskedImages() (will only be used where a fallback to
      raster is required; see setRasterFallbackThreshold())
```

setFormReconstructionCacheSize()

```
virtual void JawsMako::IRendererTransform::setFormReconstructionCacheSize (
            uint32 formCount )  [pure virtual]
```

Set whether or not the renderer should attempt to reconstruct forms that it needed to unpack due to the rendering of content within the form, and if so, set the maximum size of the cache to use (in items).

This feature is only useful if the content is to be written to a PDF output where form objects can have benefits.

In order to accurately position rendered/flattened content, the replacement content is inserted at the top level of the DOM tree. This may require the forms to be split at the point the rendering occurred.

This uses a cache to reuse recently rendered forms. Form reuse is performed by examining the content of the unpacked forms based on their graphical attributes only, and different forms in the input that share appearances may result in a single form in the output. As such, if your use case depends on non-printing information present in such objects, such as document structure or optional content, it is best not to enable this feature.

The default is 0, meaning that no forms are reconstructed and no reuse is attempted.

setGenerateFlateCompressedPDFImages()

```
virtual void JawsMako::IRendererTransform::setGenerateFlateCompressedPDFImages (
            bool enable )  [pure virtual]
```

Set whether or not the images generated by the renderer transform should be compressed with Flate as IDOMPDFImage objects. Currently this setting is only honored when vector mode is enabled.

It may not be desirable to compress very small images in this way, as it precludes the use of Indexed color spaces for very small images. As such another API is provided to provide a lower limit for image size that will be subject to this flate compression.

If the intended use is PDF output, this can result in faster processing as images will already be in the correct format for copying directly into the PDF file, and also the image will generally require less temporary storage.

The default is false.

setGenerateMasks()

```
virtual void JawsMako::IRendererTransform::setGenerateMasks (
            bool generate )  [pure virtual]
```

Set whether or not to generated masked images as the rendererd result. If true, the individual images generated as part of rendering will have a bilevel alpha channel providing a mask to just the silhouette of the dom nodes targettted for rendering. This generally results in better looking results at the intersection between rendered and non-rendered sections. The default is true.

If monochrome mode is used this function may only be called with false. See the description of setMonochromeMode() for details.

setMarkVectorFlattenedFontsForEmbedding()

```
virtual void JawsMako::IRendererTransform::setMarkVectorFlattenedFontsForEmbedding (
            bool embed )  [pure virtual]
```

Set whether or not fonts that are used for vector flattening should be marked for embedding.

If true, any font used for any geometry that defines a flattened geometry will be marked for embedding, providing that the font in question is not restricted from embedding. This does not affect non-flattened content on the page. Please note that PDF and PS output will ignore this flag for standard built-in fonts for their respective formats.

If false, the embedding status is preserved as is.

The default is true.

setMaximumRenderedResultPixels()

```
virtual void JawsMako::IRendererTransform::setMaximumRenderedResultPixels (
            uint64 maximum )  [pure virtual]
```

Set the maximum size (in pixels) of a rendered result.

```
On large jobs or at very high resolutions, a rendered section may result
in very large images, potentially in the gigabyte range. Depending on
how the output is used, this may result in output that will cause
problems with downstream consumers.

This allows a limit to be applied for each rendered section. The limit is
specified in pixels.

If the limit is exceeded, the renderer will divide the rendered section
into horizontal bands no larger than this limit. However, it will not
break apart rendered areas to smaller units than a single scanline, so in
the unlikely case where a single rendered scanline is extremely wide,
this limit will be exceeded.

Set to zero to disable the limit entirely.

The default is 125 million samples.
```

setMergeAllSpots()

```
virtual void JawsMako::IRendererTransform::setMergeAllSpots (
            bool merge )  [pure virtual]
```

Alternatively merge all spot colors in the output. This is only currently possible when rendering to a CMYK based process color space, and is ignored otherwise. Only affects rendered content. In the default case, no spots will be merged. This feature is especially useful for emulating overprint of spot components.

Please note that if any spot colorant is subject to setPreserveSpots, setPreserveAllSpots, setMergeSpots, setMergeAllSpots and/or setDropSpots, then the following rules apply in order:

- If the colorant is subject to setDropSpots, it is dropped, otherwise
- If the colorant is subject to setPreserveSpots() (not setPreserveAllSpots()!) it will be preserved (regardless of merge settings), otherwise
- If the colorant is subject to setMergeAllSpots() or setMergeSpots(), it will be merged, otherwise
- If setPreserveAllSpots() has been set to true, it will be preserved.

setMergeSpots()

```
virtual void JawsMako::IRendererTransform::setMergeSpots (
            const CSpotColorNames & mergedSpots )  [pure virtual]
```

Merge the given spot components into the process components after rendering. This is only currently possible when rendering to a CMYK based process color space, and is ignored otherwise. Only affects rendered content. In the default case, no spots will be merged. This feature is especially useful for emulating overprint of spot components.

Please note that if any spot colorant is subject to setPreserveSpots, setPreserveAllSpots, setMergeSpots, setMergeAllSpots and/or setDropSpots, then the following rules apply in order:

- If the colorant is subject to setDropSpots, it is dropped, otherwise
- If the colorant is subject to setPreserveSpots() (not setPreserveAllSpots()!) it will be preserved (regardless of merge settings), otherwise
- If the colorant is subject to setMergeAllSpots() or setMergeSpots(), it will be merged, otherwise
- If setPreserveAllSpots() has been set to true, it will be preserved.

setMinimumFlateCompressedImageSize()

```
virtual void JawsMako::IRendererTransform::setMinimumFlateCompressedImageSize (
            uint32 minSizePixels )  [pure virtual]
```

Set the minimum size, in pixels, of images that would be compressed with Flate according to setGenerateFlateCompressedPDFImages

As noted, it may not be desirable to compress very small images using flate at the renderer stage. Please see setGenerateFlateCompressedPDFImages for details.

If the intended use is PDF output, this can result in faster processing as images will already be in the correct format for copying directly into the PDF file, and also the image will generally require less temporary storage.

The default is 0, which enforces that all images generated from vector-based rendering would be compressed using flate (if setGenerateFlateCompressedPDFImages is set to true).

setMonochromeMode()

```
virtual void JawsMako::IRendererTransform::setMonochromeMode (
            bool enable = true )  [pure virtual]
```

Sets whether or not to use monochrome output.

```
    Use of this mode is supported only in specific circumstances.
    please contact Global Graphics support for details.

    The default is false; that is, continuous tone output will be
    produced, in color or grayscale as determined by the selected
    target color space or profile.

    If set to true, the output will be one-bit black and white
    halftoned images for each sections. This should only be used
    for specialised output requirements where the characteristics
    of the output device are well known and understood.

    If enabling:
      - If the current target color space is not DeviceGray or
        sGray, sGray will be set as the target color space.
      - setGenerateMasks() will be set to false. Any attempt to
        change to masked output with setGenerateMasks() will result
        in an exception being thrown.
      - setPreserveSpots() is set to an empty list. Any attemp to
        re-enable spot color preservation will result in an
        exception being thrown.
```

setPreserveAllSpots()

```
virtual void JawsMako::IRendererTransform::setPreserveAllSpots (
            bool preserve )  [pure virtual]
```

Alternatively preserve all spot colors in the output. The default is false. This function will fail if monochrome mode is used; see the description of setMonochromeMode() for details.

Please note that if any spot colorant is subject to setPreserveSpots, setPreserveAllSpots, setMergeSpots, setMergeAllSpots and/or setDropSpots, then the following rules apply in order:

- If the colorant is subject to setDropSpots, it is dropped, otherwise
- If the colorant is subject to setPreserveSpots() (not setPreserveAllSpots()!) it will be preserved (regardless of merge settings), otherwise
- If the colorant is subject to setMergeAllSpots() or setMergeSpots(), it will be merged, otherwise
- If setPreserveAllSpots() has been set to true, it will be preserved.

setPreserveSpots()

```
virtual void JawsMako::IRendererTransform::setPreserveSpots (
            const CSpotColorNames & preservedSpots )  [pure virtual]
```

Provide a list of spots to retain in the output, if present. In the default case, no spots will be retained. This function will fail if monochrome mode is used; see the description of setMonochromeMode() for details.

Please note that if any spot colorant is subject to setPreserveSpots, setPreserveAllSpots, setMergeSpots, setMergeAllSpots and/or setDropSpots, then the following rules apply in order:

- If the colorant is subject to setDropSpots, it is dropped, otherwise
- If the colorant is subject to setPreserveSpots() (not setPreserveAllSpots()!) it will be preserved (regardless of merge settings), otherwise
- If the colorant is subject to setMergeAllSpots() or setMergeSpots(), it will be merged, otherwise
- If setPreserveAllSpots() has been set to true, it will be preserved.

setRasterFallbackResolution()

```
virtual void JawsMako::IRendererTransform::setRasterFallbackResolution (
            uint32 resolution )  [pure virtual]
```

Set the resolution to be used for areas where vector mode falls back to raster mode. That is, if an area being rendered is more complex than that specified by setRasterFallbackThreshold, this is the resolution that should be used.

Using the vector flattener usually allows lower resolutions to be used for compositing image and shading content, as the resolution of the edges is high. This member allows the use of a higher resolution should raster rendering be triggered where the sharp edges defined by the vector content will not be present.

The default is the same as that set by #setRenderResolution(), and indeed, calling #setRenderResolution will result in this resolution being updated also. Therefore, this needs to be called after #setRenderResolution in order to take effect.

setRasterFallbackThreshold()

```
virtual void JawsMako::IRendererTransform::setRasterFallbackThreshold (
            uint32 threshold )  [pure virtual]
```

For cases where the vector mode is used, set the threshold at which to fall back to raster mode for a particular renderable area.

The threshold is specified in terms of rendered objects per square inch. If a rendererable area would result in more generated objects than this threshold, the renderer will fall back to a pure raster.

The default value is 100.

A value of 0 will cause the transform to always use a vector approach.

This value is ignored if setEnableVectorMode() is false.

setRasterImageReuseCacheSize()

```
virtual void JawsMako::IRendererTransform::setRasterImageReuseCacheSize (
            uint32 reuseCacheSizeMB )  [pure virtual]
```

Set whether or not the renderer should attempt to cache rendered results for raster rendering, and if so, set the maximum size of the cache, in megabytes.

This can provide significant performance improvements for cases where jobs contain reused content that is subject to the renderer.

The default is 0, meaning that no raster results are cached.

setRenderTransparentNodesOnPageGroupMismatch()

```
virtual void JawsMako::IRendererTransform::setRenderTransparentNodesOnPageGroupMismatch (
            bool render )  [pure virtual]
```

Set whether to render transparent nodes if the page blending group color space differs from the rendering target color space.

This affects how blending is performed, and non-rendered transparent content may result in color differences compared with rendered content if there is a mismatch.

To avoid this issue, use true here. If true, and the renderer transform detects that the page blending group color space and the target color space, then all transparent content will be rendered. IDOMColorSpace::similar() is used for this purpose.

If false, the current renderTransparentNodes() setting will be honored.

The default is false.

setSpotHalftone()

```
virtual void JawsMako::IRendererTransform::setSpotHalftone (
            float frequency,
            bool useFullResolutionForFlattening = false )  [pure virtual]
```

Set the simple spot haltone to be used when monochrome mode is enabled.

The spot halftone is always at 45 degrees. The default is 60.0f. The halftone is not used if monochrome mode is not enabled. The halftone frequency must be greater than 0. Normally, the resolution at which compositing for transparent images occurs is capped at twice the halftone frequency. For some content, this may not be appropriate. Set useFullResolutionForFlattening for such cases.

setThresholdHalftone()

```
virtual void JawsMako::IRendererTransform::setThresholdHalftone (
            uint32 width,
            uint32 height,
            const CThresholdArray & thresholds )  [pure virtual]
```

Set a threshold array halftone to be used when monochrome mode is enabled.

Only 8 bit threshold arrays are supported, and the threshold array must not be larger than 65535 entries. Please refer to section 7.4.5 of the PostScript language reference manual, 3rd edition.

Parameters

width	The width of the halftone cell, in pixels
height	The height of the halftone cell, in pixels
thresholds	The threshold array. Must be width $*$ height in length, and no longer than 65535 entries.

setUseImageResolutionForRenderingWherePossible()

```
virtual void JawsMako::IRendererTransform::setUseImageResolutionForRenderingWherePossible (
            bool enable )  [pure virtual]
```

Set whether or not areas with compatible images may be rendered at image resolution if possible when vector flatting mode is enabled (see setEnableVectorMode())

If true, and a section of the page that requires flattening contains only images sharing the same scaling and constant fills, and the images are at a lower resolution than the target resolution, then the image data will be flattened at the resolution of the image, and not the target resolution.

If false, such images will be flattened at the target resolution instead.

This is ideal when there is for example, a section of the page where the only non-constant-filled content is a transparent image. If no other images or complex object overlaps, then the image may be replaced with a flattened result at the same resolution as the source image.

The default is true.

setVectorAndTextResolution()

```
virtual void JawsMako::IRendererTransform::setVectorAndTextResolution (
            uint32 resolution )  [pure virtual]
```

Set the desired resolution for vector and textual content when vector flattening mode is enabled (see setEnableVectorMode()).

The default is 1200dpi.

setVectorAreaFormReuseCacheSize()

```
virtual void JawsMako::IRendererTransform::setVectorAreaFormReuseCacheSize (
            uint32 formCount )  [pure virtual]
```

Set whether or not the renderer, with vector mode enabled, should cache entire flattened sections as IDOMForm objects, and if so, set the maximum size of the cache (in items).

When used in concert with setVectorImageReuseCacheSize() above, this can both improve performance and also reduce output size if large areas of content requiring flattening/rendering are reused.

The default is 0, meaning that no forms are generated and no reuse is attempted.

setVectorAreaSourceReuseCacheSize()

```
virtual void JawsMako::IRendererTransform::setVectorAreaSourceReuseCacheSize (
            uint32 count )  [pure virtual]
```

Set whether or not the renderer, with vector mode enabled, should attempt to cache entire flattened areas based on the source content, and if so, set the maximum size of the cache (in items).

This is another level of caching that allows for caching of flattened sections at the DOM stage and before any vector flattening computations have taken place. Highly useful when processing cases that contain large amounts of reuse, especially in concert with setVectorAreaFormReuseCacheSize.

The default is 0, meaning that the cache is not used.

setVectorImageReuseCacheSize()

```
virtual void JawsMako::IRendererTransform::setVectorImageReuseCacheSize (
            uint32 reuseCacheSizeMB )  [pure virtual]
```

Set whether or not the renderer, with vector mode enabled, should attempt to reused flattened sections from previously rendered areas, and if so, set the maximum size of the cache in megabytes.

This can provide significant performance benefits particularly for jobs where many pages share content that needs to be flattened/rendered.

The default is 0, meaning that no reuse is attempted.

The documentation for this class was generated from the following file:

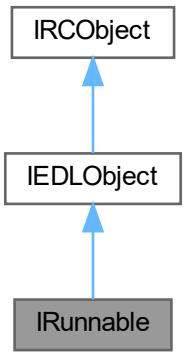
- transforms.h

7.406 IRunnable Class Reference

Interface to filter's runnable classes.

`#include <isession.h>`

Inheritance diagram for IRunnable:



Public Member Functions

- virtual RunCode run ()=0

  Run runnable object.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()
  Virtual destructor.

7.406.1 Detailed Description

Interface to filter's runnable classes.

7.406.2 Member Function Documentation

run()

```
virtual RunCode IRunnable::run ( )  [pure virtual]
```

Run runnable object.

Returns

RunCode Returns run code

The documentation for this class was generated from the following file:

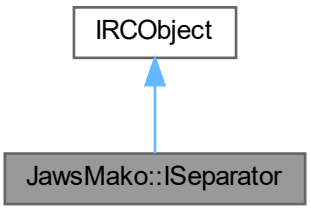
- isession.h

7.407 JawsMako::ISeparator Class Reference

An instance of the JawsMako Separator class.

```
#include <separator.h>
```

Inheritance diagram for JawsMako::ISeparator:



Public Member Functions

- virtual void separate (const IDOMFixedPagePtr &content)=0

  Separate a fixed page.
- virtual void getSeparation (uint32 index, IDOMFixedPagePtr &content, U8String &name) const =0

  Retrieve a separated page.
- virtual void getSeparation (uint32 index, IDOMFixedPagePtr &content)

  Retrieve a separated page.
- virtual uint32 getNumSeparations () const =0

  Return the number of separations. Returns the number of available separated pages generated with separate().
- virtual void **setTransformSpots** (bool transformSpots)=0

  Set whether or not spot colours should be transformed to the target color space when separating. The default is set when creating the separator instance with create().
- virtual void setEnableVectorMode (bool enableVectorMode)=0

  Enable "vector" flattening mode.
- virtual void setRasterFallbackThreshold (uint32 rasterFallbackThreshold)=0

  For cases where the vector mode is used, set the threshold at which to fall back to raster mode for a particular renderable area.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API ISeparatorPtr create (const IJawsMakoPtr &jawsMako, bool transformSpots=false, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create a separator instance.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.407.1 Detailed Description

An instance of the JawsMako Separator class.

7.407.2 Member Function Documentation

create()

```
static JAWSMAKO_API ISeparatorPtr JawsMako::ISeparator::create (
            const IJawsMakoPtr & jawsMako,
            bool transformSpots = false,
            const IProgressMonitorPtr & progressMonitor = IProgressMonitorPtr() )  [static]
```

Create a separator instance.

Parameters

jawsMako	The JawsMako instance.
transformSpots	Whether spot colours should be transformed to the target color space. The default is false.
progressMonitor	The progress monitor which allows aborting an operation or registering a progress callback.

Returns

ISeparatorPtr The new separator instance.

getNumSeparations()

```
virtual uint32 JawsMako::ISeparator::getNumSeparations ( ) const  [pure virtual]
```

Return the number of separations. Returns the number of available separated pages generated with separate().

Returns

uint32 The number of separations.

getSeparation() [1/2]

```
virtual void JawsMako::ISeparator::getSeparation (
            uint32 index,
            IDOMFixedPagePtr & content )  [inline], [virtual]
```

Retrieve a separated page.

Parameters

index	The index of the separated fixed page to retreive. Will throw an exception if there is no separated page at the given index.
content	The separated fixed page.

getSeparation() [2/2]

```
virtual void JawsMako::ISeparator::getSeparation (
            uint32 index,
            IDOMFixedPagePtr & content,
            U8String & name ) const  [pure virtual]
```

Retrieve a separated page.

Parameters

index	The index of the separated fixed page to retreive. Will throw an exception if there is no separated page at the given index.
content	The separated fixed page.
name	Reference to receive the colorant name of the separated page.

separate()

```
virtual void JawsMako::ISeparator::separate (
            const IDOMFixedPagePtr & content )  [pure virtual]
```

Separate a fixed page.

Parameters

content	The input fixed page to separate. Use getSeparation() to get the separated pages.

setEnableVectorMode()

```
virtual void JawsMako::ISeparator::setEnableVectorMode (
            bool enableVectorMode )  [pure virtual]
```

Enable "vector" flattening mode.

```
NOTE: This is functionality is beta quality only at this time.

When flattening transparency with this mode set, the renderer
transform used by ISeparator will attempt to maintain vector
and textual information as much as possible in its results,
and may not need to render some objects at all.

However, for very complicated cases this may result in a large
amount of output geometry that could result in larger output
files or files that process more slowly than an equivalent
raster render.

The default is false.
```

setRasterFallbackThreshold()

```
virtual void JawsMako::ISeparator::setRasterFallbackThreshold (
            uint32 rasterFallbackThreshold )  [pure virtual]
```

For cases where the vector mode is used, set the threshold at which to fall back to raster mode for a particular renderable area.

The threshold is specified in terms of rendered objects per square inch. If a rendererable area would result in more generated objects than this threshold, the renderer will fall back to a pure raster.

The default value is 100.

A value of 0 will cause the transform to always use a vector approach.

This value is ignored if setEnableVectorMode() is false.

The documentation for this class was generated from the following file:

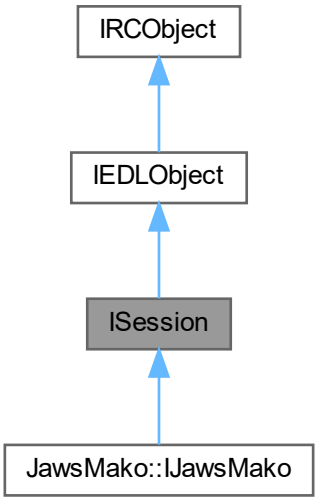
- separator.h

7.408 ISession Class Reference

EDL session class.

`#include <isession.h>`

Inheritance diagram for ISession:



Public Member Functions

- virtual bool setFactory (IEDLClassFactory *pFactory)=0

  initializes Session by setting EDL class Factory
- virtual IEDLClassFactory * getFactory ()=0

  EDL Class Factory getter method.
- virtual IMessageHandlerPtr getMessageHandler ()=0

  Obtain the session's message handler.
- virtual ILiteMessageHandlerPtr getLiteMessageHandler ()=0

  Obtain the session's litemessage handler.
- virtual bool setTemporaryDirectory (const EDLSysString &sTempDirectory)=0

  Set the temporary directory to be used by EDL and filters for this session.
- virtual bool getTemporaryDirectory (EDLSysString &sTempDirectory)=0

  Get the temporary directory to be used by EDL and filters for this session.
- virtual IEDLTempStorePtr getTempStore ()=0

  Get the temporary store for this session. The temporary directory must be set before calling this member. Throws an IEDLError on failure.
- virtual bool setStartupDirectory (const EDLSysString &sStartupDirectory)=0

  Set the startup directory. This is meaningfull for executable environments. Note the client must set this during initalisation for it to be valid.
- virtual bool getStartupDirectory (EDLSysString &sStartupDirectory)=0

  Get the startup directory.

Public Member Functions inherited from IEDLObject

- virtual const CClassID & getClassID () const =0

  Returns class ID of IEDLObject.
- virtual bool init (CClassParams *pData)

  The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.
- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)

  Create a copy of EDLObject.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.408.1 Detailed Description

EDL session class.

7.408.2 Member Function Documentation

getFactory()

```
virtual IEDLClassFactory * ISession::getFactory ( )  [pure virtual]
```

EDL Class Factory getter method.

Returns

IEDLClassFactory* Returns pointer to EDL class factory

getLiteMessageHandler()

```
virtual ILiteMessageHandlerPtr ISession::getLiteMessageHandler ( )  [pure virtual]
```

Obtain the session's litemessage handler.

Each session also contains a litemessage handler, used to deal with litemessages. The Litemessage framework will eventually be taken over by the proper messaging framework.

Returns

ILiteMessageHandlerPtr Returns Smart pointer to a litemessage handler object.

getMessageHandler()

```
virtual IMessageHandlerPtr ISession::getMessageHandler ( )  [pure virtual]
```

Obtain the session's message handler.

Each session has a message handler, used to deal with messages sent between various components being used in the context of a particular session. This returns a pointer to a session's message handler.

Returns

IMessageHandlerPtr Smart pointer to a message handler object.

getStartupDirectory()

```
virtual bool ISession::getStartupDirectory (
            EDLSysString & sStartupDirectory )  [pure virtual]
```

Get the startup directory.

Parameters

sStartupDirectory	The path of the directory that the execuatble started.

Returns

bool Returns true on success.

getTemporaryDirectory()

```
virtual bool ISession::getTemporaryDirectory (
            EDLSysString & sTempDirectory )  [pure virtual]
```

Get the temporary directory to be used by EDL and filters for this session.

Parameters

sTempDirectory	A reference to receive path to the temp directory.

Returns

bool Returns true on success.

getTempStore()

```
virtual IEDLTempStorePtr ISession::getTempStore ( )  [pure virtual]
```

Get the temporary store for this session. The temporary directory must be set before calling this member. Throws an IEDLError on failure.

Returns

IEDLTempStorePtr The temp store.

setFactory()

```
virtual bool ISession::setFactory (
            IEDLClassFactory * pFactory )  [pure virtual]
```

initializes Session by setting EDL class Factory

Parameters

pFactory	Ptr to EDL Class Factory

Returns

bool Returns true on success

setStartupDirectory()

```
virtual bool ISession::setStartupDirectory (
            const EDLSysString & sStartupDirectory )  [pure virtual]
```

Set the startup directory. This is meaningfull for executable environments. Note the client must set this during initalisation for it to be valid.

Parameters

sStartupDirectory	The path of the directory that the executable started.

Returns

bool Returns true on success.

setTemporaryDirectory()

```
virtual bool ISession::setTemporaryDirectory (
            const EDLSysString & sTempDirectory )  [pure virtual]
```

Set the temporary directory to be used by EDL and filters for this session.

Parameters

sTempDirectory	The path to the temp directory. This directory must exist and the path must not be zero length. Note that for safety, this should be an absolute path.

Returns

bool Returns true on success.

The documentation for this class was generated from the following file:

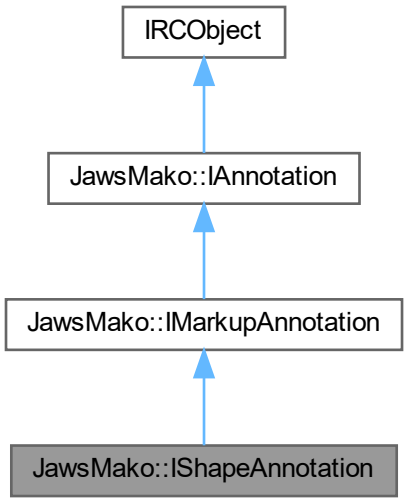
- isession.h

7.409 JawsMako::IShapeAnnotation Class Reference

A generic interface class for circle and square annotations. It is intended that future releases of JawsMako will extend this interface.

`#include <interactive.h>`

Inheritance diagram for JawsMako::IShapeAnnotation:



Public Member Functions

- virtual CRectInset getRectInset () const =0

  Get the rect inset describing, relative to the annotation rect, the shape area within the annotation.
- virtual IDOMColorPtr getInteriorColor () const =0

  Get the interior color of the fill used for the shape.
- virtual void setInteriorColor (const IDOMColorPtr &color)=0

  Set the interior color to be used to fill the shapes.

Public Member Functions inherited from JawsMako::IMarkupAnnotation

- virtual U8String getAuthor () const =0

  Get the Author of the markup annotation. This is the "T" entry in the annotation, which by convention is the author.
- virtual void setAuthor (const U8String &author)=0

  Set the Author of the markup annotation. This is the "T" entry in the annotation, which by convention is the author.
- virtual IEDLTimePtr getCreationTime () const =0

  Get the creation date and time of the annotation, if present.
- virtual void setCreationTime (const IEDLTimePtr &creationTime)=0

  Set the creation date and time of the annotation.
- virtual float getOpacity () const =0

  Get the opacity of the markup annotation.
- virtual void setOpacity (float opacity)=0

  Set the opacity of the markup annotation.
- virtual IAnnotationReferencePtr getPopupReference () const =0

  Get a reference to the popup, if present.
- virtual void setPopup (const IAnnotationReferencePtr &popupReference)=0

  Set or clear the popup, by reference.
- virtual void setPopup (const IPopupAnnotationPtr &popup)=0

  Set a reference to a popup, if present.
- virtual IAnnotationAppearancePtr createNormalAppearance () const =0

  Create a basic appearance given the current annotation's parameters. This can then be installed by using IAnnotation::addAppearance().

Public Member Functions inherited from JawsMako::IAnnotation

- virtual eAnnotationType getType () const =0

  Get the type of the annotation.
- virtual const FRect & getRect () const =0

  Get the rect in which the appearances should be displayed.
- virtual void setRect (const FRect &rect)=0

  Set the rect in which the appearances should be displayed.
- virtual U8String getContents () const =0

  Get the Contents entry in UTF-8.
- virtual void setContents (const U8String &contents)=0

  Set the Contents entry in UTF-8.
- virtual IDOMColorPtr getColor () const =0

  Get the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.
- virtual void setColor (const IDOMColorPtr &color)=0

  Set the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.
- virtual IEDLTimePtr getModificationTime () const =0

Get the Modification date and time of the annotation, if present.

- virtual void setModificationTime (const IEDLTimePtr &modificationTime)=0

  Set the Modification date and time of the annotation.

- virtual CAnnotationBorder getBorder () const =0

  Get the annotation's border. See CAnnotationBorder for details.

- virtual void setBorder (const CAnnotationBorder &border)=0

  Set the annotation's border.

- virtual uint32 getFlags () const =0

  Get the annotation flags. To interpret these flags please see section 8.4.2 "Annotation Flags" in the PDF 1.7 specification.

- virtual void setFlags (uint32 flags)=0

  Set the annotation flags. Please see section 8.4.2 "Annotation Flags" for details about these flags.

- virtual void rotate (uint16 rotate, double pageWidth, double pageHeight)=0

  Rotate the annotation clockwise as if the page was rotated by the same amount.

- virtual CAnnotationAppearanceVect getAppearances () const =0

  Return all the annotation appearances in a vector.

- virtual void **removeAppearances** ()=0

  Remove all annotation appearances.

- virtual U8String getState () const =0

  Get the current annotation state.

- virtual void setState (const U8String &state)=0

  Set the current annotation state.

- virtual IAnnotationAppearancePtr getAppearance (eAppearanceUsage usage, const U8String &state=U8String()) const =0

  Fetch the annotation appearance that should be used for the given annotation usage and state according to the following rules:

- virtual void addAppearance (const IAnnotationAppearancePtr &appearance)=0

  Add or replace an appearance.

- virtual bool hasNormalAppearance () const =0

  Does the annotation have a normal appearance? Convenience utility function.

- virtual IAnnotationPtr clone () const =0

  Clone the annotation. This is a deep clone. The annotation reference will remain the same.

- virtual bool matchesReference (const IAnnotationReferencePtr &reference) const =0

  Does this annotation match the given IAnnotationReference?

- virtual IAnnotationReferencePtr getReference () const =0

  Get a reference that can be used to refer to this annotation.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.

- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.

- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IShapeAnnotationPtr create (const IJawsMakoPtr &jawsMako, const FRect &rect, IAnnotation::eAnnotationType type, const IDOMColorPtr &color=IDOMColorPtr(), const CAnnotationBorder &border=CAnnotationBorder(), const IDOMColorPtr &interior=IDOMColorPtr(), const CRectInset &inset=CRectInset())

  Create a shape (Rectangle or Circle) annotation with the given parameters. An appearance for the new annotation will be created based on the parameters using createNormalAppearance().

Additional Inherited Members

Public Types inherited from JawsMako::IAnnotation

- enum eAnnotationType {
  eAT3D , eATCaret , eATCircle , eATFileAttachment ,
  eATFreeText , eATHighlight , eATInk , eATLine ,
  eATLink , eATMovie , eATPolygon , eATPolyLine ,
  eATPopup , eATPrinterMark , eATProjection , eATRedact ,
  eATRichMedia , eATScreen , eATSound , eATSquare ,
  eATSquiggly , eATStamp , eATStrikeOut , eATText ,
  eATTrapNet , eATUnderline , eATWatermark , eATWidget ,
  eATOther }

  Types of annotations, listed with the earliest version of PDF that supported them.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.409.1 Detailed Description

A generic interface class for circle and square annotations. It is intended that future releases of JawsMako will extend this interface.

7.409.2 Member Function Documentation

create()

```
static JAWSMAKO_API IShapeAnnotationPtr JawsMako::IShapeAnnotation::create (
            const IJawsMakoPtr & jawsMako,
            const FRect & rect,
            IAnnotation::eAnnotationType type,
            const IDOMColorPtr & color = IDOMColorPtr(),
            const CAnnotationBorder & border = CAnnotationBorder(),
            const IDOMColorPtr & interior = IDOMColorPtr(),
            const CRectInset & inset = CRectInset() )  [static]
```

Create a shape (Rectangle or Circle) annotation with the given parameters. An appearance for the new annotation will be created based on the parameters using createNormalAppearance().

Parameters

jawsMako	The JawsMako instance
rect	The annotations bounds. Must not be empty.
type	The type of annotation to create. Must be IAnnotation::eATSquare or IAnnotation::eATCircle
color	Optional; The desired annotation color. If NULL, no line will be visible.
border	Optional; The border to use. If not provided, a default will be used.
interior	Optional; the fill color to use. If NULL, no fill will be visible.
inset	Optional; The amount to inset the line from the annotation rectangle.

Returns

IShapeAnnotationPtr A smart pointer to the new shape annotation

getInteriorColor()

```
virtual IDOMColorPtr JawsMako::IShapeAnnotation::getInteriorColor ( ) const  [pure virtual]
```

Get the interior color of the fill used for the shape.

Returns

IDOMColorPtr The fill color. NULL is returned if there is no such color

getRectInset()

```
virtual CRectInset JawsMako::IShapeAnnotation::getRectInset ( ) const  [pure virtual]
```

Get the rect inset describing, relative to the annotation rect, the shape area within the annotation.

Returns

CRectInset The rect inset

setInteriorColor()

```
virtual void JawsMako::IShapeAnnotation::setInteriorColor (
            const IDOMColorPtr & color )  [pure virtual]
```

Set the interior color to be used to fill the shapes.

Parameters

color	The color definition, or pass NULL to remove the color

The documentation for this class was generated from the following file:

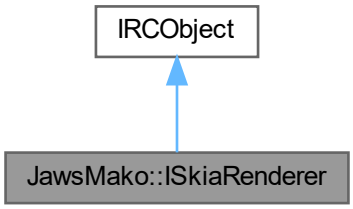
- interactive.h

7.410 JawsMako::ISkiaRenderer Class Reference

A renderer that can paint XPS compatible DOM into a Skia canvas using the Skia API.

`#include <skiarenderer.h>`

Inheritance diagram for JawsMako::ISkiaRenderer:



Public Member Functions

- virtual void drawNode (const IDOMNodePtr &node, SkCanvas *canvas)=0

  Render the given node into the Skia canvas.
- virtual void flushCaches ()=0

  Flush all caches used by the renderer.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API ISkiaRendererPtr create (const IJawsMakoPtr &jawsMako)

  Create a Skia Renderer Instance.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.410.1 Detailed Description

A renderer that can paint XPS compatible DOM into a Skia canvas using the Skia API.

7.410.2 Member Function Documentation

create()

```
static JAWSMAKO_API ISkiaRendererPtr JawsMako::ISkiaRenderer::create (
            const IJawsMakoPtr & jawsMako )  [static]
```

Create a Skia Renderer Instance.

```
As some data will be cached, it is best to reuse a single
instance when repeatedly rendering the same content or
content from the same document.
```

drawNode()

```
virtual void JawsMako::ISkiaRenderer::drawNode (
            const IDOMNodePtr & node,
            SkCanvas * canvas )  [pure virtual]
```

Render the given node into the Skia canvas.

```
It is safe for multiple threads to call this member at the
same time, but care should be taken to ensure that the
target canvas environment supports this.
```

flushCaches()

```
virtual void JawsMako::ISkiaRenderer::flushCaches ( )  [pure virtual]
```

Flush all caches used by the renderer.

```
This is required if the renderer has previously been used to
paint content from a file that no longer exists as some of the
cached results may point to content within such files. It is
advisable to invoke this function if such a file is deleted or
moved.

Secondarily, this may provide temporary memory relief as
cached objects are released.
```

The documentation for this class was generated from the following file:

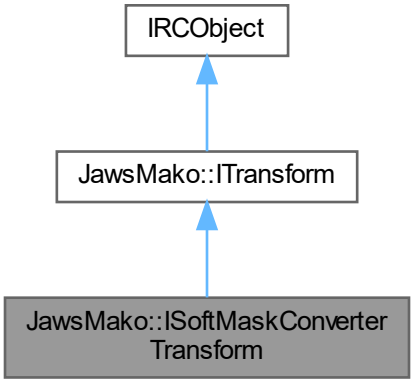
- skiarenderer.h

7.411 JawsMako::ISoftMaskConverterTransform Class Reference

A transform that converts PDF style soft masks to opacity masks, suitable for XPS.

`#include <transforms.h>`

Inheritance diagram for JawsMako::ISoftMaskConverterTransform:



Public Member Functions

- virtual void **setResolution** (uint32 resolution)=0

  Set the resolution to be used where rendering is required. The default is 150dpi.

Public Member Functions inherited from JawsMako::ITransform

- virtual IDOMBrushPtr transform (const IDOMBrushPtr &brush, eBrushUsage usage=eBUGeneral, const CTransformState &state=CTransformState())=0

  Apply the transform to the given brush, if applicable. These transforms are thread safe.
- virtual IDOMImagePtr transform (const IDOMImagePtr &image, const CTransformState &state=CTransformState())=0

  Apply the transform to the given image, if applicable. These transforms are thread safe.
- virtual IDOMColorPtr transform (const IDOMColorPtr &color, const CTransformState &state=CTransformState())=0

  Apply the transform to the given color, if applicable. These transforms are thread safe.
- virtual IDOMColorSpacePtr transform (const IDOMColorSpacePtr &colorSpace, const CTransformState &state=CTransformState())=0

  Apply the transform to the given colorspace, if applicable. These transforms are thread safe.
- virtual IDOMNodePtr transform (const IDOMNodePtr &node, bool &changed, bool transformChildren=true, const CTransformState &state=CTransformState())=0

  Apply the transform to the given node, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time.
- virtual void transformPage (const IPagePtr &page, bool transformContent=true, bool transform↩ Annotations=true)=0

  Apply the transform to the given page, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time. The transform will also apply to the annotations appearances.

- virtual void **flushCaches** ()=0

  Flush the caches used by the transform. Most transforms cache recently transformed results to improve the performance of repeated transformations of equivalent results. However, it is possible that some cached results may point to entities that no longer exist, such as content inside an XPS file that no longer exists. If you are deleting or replacing files where transforms have been used, it is advisable to invoke this routine to clear the caches.
- virtual void setProgressMonitor (const IProgressMonitorPtr &progressMonitor)=0

  Set the IProgressMonitor object for this transform to allow for monitoring the progress of the transform.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.411.1 Detailed Description

A transform that converts PDF style soft masks to opacity masks, suitable for XPS.

The documentation for this class was generated from the following file:

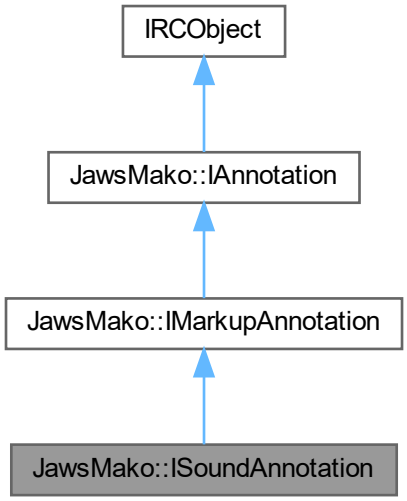
- transforms.h

7.412 JawsMako::ISoundAnnotation Class Reference

An interface class for a sound annotation. Allows access to the sound as a WAV stream if the stream is embedded. It is intended that future releases of JawsMako will extend this interface.

```
#include <interactive.h>
```

Inheritance diagram for JawsMako::ISoundAnnotation:



Public Member Functions

- virtual IInputStreamPtr getSoundAsWav () const =0

  Get the embedded sound (if present) as a WAV stream.

Public Member Functions inherited from JawsMako::IMarkupAnnotation

- virtual U8String getAuthor () const =0

  Get the Author of the markup annotation. This is the "T" entry in the annotation, which by convention is the author.
- virtual void setAuthor (const U8String &author)=0

  Set the Author of the markup annotation. This is the "T" entry in the annotation, which by convention is the author.
- virtual IEDLTimePtr getCreationTime () const =0

  Get the creation date and time of the annotation, if present.
- virtual void setCreationTime (const IEDLTimePtr &creationTime)=0

  Set the creation date and time of the annotation.
- virtual float getOpacity () const =0

  Get the opacity of the markup annotation.
- virtual void setOpacity (float opacity)=0

  Set the opacity of the markup annotation.
- virtual IAnnotationReferencePtr getPopupReference () const =0

  Get a reference to the popup, if present.
- virtual void setPopup (const IAnnotationReferencePtr &popupReference)=0

  Set or clear the popup, by reference.
- virtual void setPopup (const IPopupAnnotationPtr &popup)=0

  Set a reference to a popup, if present.
- virtual IAnnotationAppearancePtr createNormalAppearance () const =0

  Create a basic appearance given the current annotation's parameters. This can then be installed by using IAnnotation::addAppearance().

Public Member Functions inherited from JawsMako::IAnnotation

- virtual eAnnotationType getType () const =0

  Get the type of the annotation.
- virtual const FRect & getRect () const =0

  Get the rect in which the appearances should be displayed.
- virtual void setRect (const FRect &rect)=0

  Set the rect in which the appearances should be displayed.
- virtual U8String getContents () const =0

  Get the Contents entry in UTF-8.
- virtual void setContents (const U8String &contents)=0

  Set the Contents entry in UTF-8.
- virtual IDOMColorPtr getColor () const =0

  Get the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.
- virtual void setColor (const IDOMColorPtr &color)=0

  Set the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.
- virtual IEDLTimePtr getModificationTime () const =0

  Get the Modification date and time of the annotation, if present.
- virtual void setModificationTime (const IEDLTimePtr &modificationTime)=0

  Set the Modification date and time of the annotation.
- virtual CAnnotationBorder getBorder () const =0

  Get the annotation's border. See CAnnotationBorder for details.
- virtual void setBorder (const CAnnotationBorder &border)=0

  Set the annotation's border.
- virtual uint32 getFlags () const =0

  Get the annotation flags. To interpret these flags please see section 8.4.2 "Annotation Flags" in the PDF 1.7 specification.
- virtual void setFlags (uint32 flags)=0

  Set the annotation flags. Please see section 8.4.2 "Annotation Flags" for details about these flags.
- virtual void rotate (uint16 rotate, double pageWidth, double pageHeight)=0

  Rotate the annotation clockwise as if the page was rotated by the same amount.
- virtual CAnnotationAppearanceVect getAppearances () const =0

  Return all the annotation appearances in a vector.
- virtual void **removeAppearances** ()=0

  Remove all annotation appearances.
- virtual U8String getState () const =0

  Get the current annotation state.
- virtual void setState (const U8String &state)=0

  Set the current annotation state.
- virtual IAnnotationAppearancePtr getAppearance (eAppearanceUsage usage, const U8String &state=U8String()) const =0

  Fetch the annotation appearance that should be used for the given annotation usage and state according to the following rules:
- virtual void addAppearance (const IAnnotationAppearancePtr &appearance)=0

  Add or replace an appearance.
- virtual bool hasNormalAppearance () const =0

  Does the annotation have a normal appearance? Convenience utility function.
- virtual IAnnotationPtr clone () const =0

  Clone the annotation. This is a deep clone. The annotation reference will remain the same.
- virtual bool matchesReference (const IAnnotationReferencePtr &reference) const =0

  Does this annotation match the given IAnnotationReference?
- virtual IAnnotationReferencePtr getReference () const =0

  Get a reference that can be used to refer to this annotation.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Public Types inherited from JawsMako::IAnnotation

- enum eAnnotationType {
  eAT3D , eATCaret , eATCircle , eATFileAttachment ,
  eATFreeText , eATHighlight , eATInk , eATLine ,
  eATLink , eATMovie , eATPolygon , eATPolyLine ,
  eATPopup , eATPrinterMark , eATProjection , eATRedact ,
  eATRichMedia , eATScreen , eATSound , eATSquare ,
  eATSquiggly , eATStamp , eATStrikeOut , eATText ,
  eATTrapNet , eATUnderline , eATWatermark , eATWidget ,
  eATOther }

  Types of annotations, listed with the earliest version of PDF that supported them.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.412.1 Detailed Description

An interface class for a sound annotation. Allows access to the sound as a WAV stream if the stream is embedded. It is intended that future releases of JawsMako will extend this interface.

7.412.2 Member Function Documentation

getSoundAsWav()

```
virtual IInputStreamPtr JawsMako::ISoundAnnotation::getSoundAsWav ( ) const  [pure virtual]
```

Get the embedded sound (if present) as a WAV stream.

Returns

IInputStreamPtr The WAV stream, or NULL if there is no embedded stream

The documentation for this class was generated from the following file:

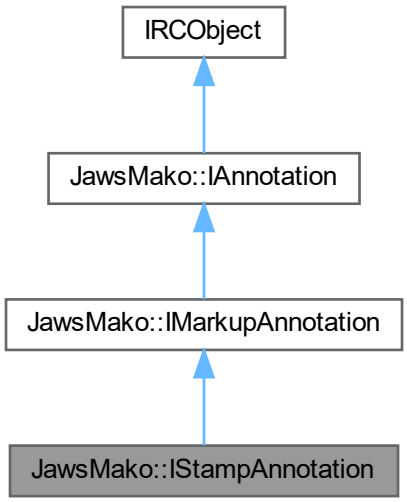
- interactive.h

7.413 JawsMako::IStampAnnotation Class Reference

A generic interface class for a stamp annotation.

`#include <interactive.h>`

Inheritance diagram for JawsMako::IStampAnnotation:



Public Member Functions

- virtual U8String getName () const =0

  Get the stamp annotation's name.
- virtual void setName (const U8String &name)=0

  Set the stamp annotation's name.
- virtual U8String getIntent () const =0

  Get the stamp annotation's intent.
- virtual void setIntent (const U8String &intent)=0

  Set the stamp annotation's intent. Note: This is a PDF 2.0 feature. If intent is set to anything other than Stamp (the default), then the name of the stamp annotation will be dropped.

Public Member Functions inherited from JawsMako::IMarkupAnnotation

- virtual U8String getAuthor () const =0

  Get the Author of the markup annotation. This is the "T" entry in the annotation, which by convention is the author.
- virtual void setAuthor (const U8String &author)=0

  Set the Author of the markup annotation. This is the "T" entry in the annotation, which by convention is the author.
- virtual IEDLTimePtr getCreationTime () const =0

  Get the creation date and time of the annotation, if present.
- virtual void setCreationTime (const IEDLTimePtr &creationTime)=0

Set the creation date and time of the annotation.

- virtual float getOpacity () const =0

  Get the opacity of the markup annotation.
- virtual void setOpacity (float opacity)=0

  Set the opacity of the markup annotation.
- virtual IAnnotationReferencePtr getPopupReference () const =0

  Get a reference to the popup, if present.
- virtual void setPopup (const IAnnotationReferencePtr &popupReference)=0

  Set or clear the popup, by reference.
- virtual void setPopup (const IPopupAnnotationPtr &popup)=0

  Set a reference to a popup, if present.
- virtual IAnnotationAppearancePtr createNormalAppearance () const =0

  Create a basic appearance given the current annotation's parameters. This can then be installed by using IAnnotation::addAppearance().

Public Member Functions inherited from JawsMako::IAnnotation

- virtual eAnnotationType getType () const =0

  Get the type of the annotation.
- virtual const FRect & getRect () const =0

  Get the rect in which the appearances should be displayed.
- virtual void setRect (const FRect &rect)=0

  Set the rect in which the appearances should be displayed.
- virtual U8String getContents () const =0

  Get the Contents entry in UTF-8.
- virtual void setContents (const U8String &contents)=0

  Set the Contents entry in UTF-8.
- virtual IDOMColorPtr getColor () const =0

  Get the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.
- virtual void setColor (const IDOMColorPtr &color)=0

  Set the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.
- virtual IEDLTimePtr getModificationTime () const =0

  Get the Modification date and time of the annotation, if present.
- virtual void setModificationTime (const IEDLTimePtr &modificationTime)=0

  Set the Modification date and time of the annotation.
- virtual CAnnotationBorder getBorder () const =0

  Get the annotation's border. See CAnnotationBorder for details.
- virtual void setBorder (const CAnnotationBorder &border)=0

  Set the annotation's border.
- virtual uint32 getFlags () const =0

  Get the annotation flags. To interpret these flags please see section 8.4.2 "Annotation Flags" in the PDF 1.7 specification.
- virtual void setFlags (uint32 flags)=0

  Set the annotation flags. Please see section 8.4.2 "Annotation Flags" for details about these flags.
- virtual void rotate (uint16 rotate, double pageWidth, double pageHeight)=0

  Rotate the annotation clockwise as if the page was rotated by the same amount.
- virtual CAnnotationAppearanceVect getAppearances () const =0

  Return all the annotation appearances in a vector.
- virtual void **removeAppearances** ()=0

  Remove all annotation appearances.
- virtual U8String getState () const =0

Get the current annotation state.

- virtual void setState (const U8String &state)=0

  Set the current annotation state.

- virtual IAnnotationAppearancePtr getAppearance (eAppearanceUsage usage, const U8String &state=U8String()) const =0

  Fetch the annotation appearance that should be used for the given annotation usage and state according to the following rules:

- virtual void addAppearance (const IAnnotationAppearancePtr &appearance)=0

  Add or replace an appearance.

- virtual bool hasNormalAppearance () const =0

  Does the annotation have a normal appearance? Convenience utility function.

- virtual IAnnotationPtr clone () const =0

  Clone the annotation. This is a deep clone. The annotation reference will remain the same.

- virtual bool matchesReference (const IAnnotationReferencePtr &reference) const =0

  Does this annotation match the given IAnnotationReference?

- virtual IAnnotationReferencePtr getReference () const =0

  Get a reference that can be used to refer to this annotation.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.

- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.

- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IStampAnnotationPtr create (const IJawsMakoPtr &jawsMako, const FRect &rect, const U8String &name=U8String("Draft"))

  Create a stamp annotation.

Additional Inherited Members

Public Types inherited from JawsMako::IAnnotation

- enum eAnnotationType {
  eAT3D , eATCaret , eATCircle , eATFileAttachment ,
  eATFreeText , eATHighlight , eATInk , eATLine ,
  eATLink , eATMovie , eATPolygon , eATPolyLine ,
  eATPopup , eATPrinterMark , eATProjection , eATRedact ,
  eATRichMedia , eATScreen , eATSound , eATSquare ,
  eATSquiggly , eATStamp , eATStrikeOut , eATText ,
  eATTrapNet , eATUnderline , eATWatermark , eATWidget ,
  eATOther }

  Types of annotations, listed with the earliest version of PDF that supported them.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()
  Virtual destructor.

7.413.1 Detailed Description

A generic interface class for a stamp annotation.

7.413.2 Member Function Documentation

create()

```
static JAWSMAKO_API IStampAnnotationPtr JawsMako::IStampAnnotation::create (
            const IJawsMakoPtr & jawsMako,
            const FRect & rect,
            const U8String & name = U8String("Draft") )  [static]
```

Create a stamp annotation.

Parameters

jawsMako	The JawsMako instance
rect	The annotations bounds. Must not be empty
name	The "name" of the stamp

Returns

IStampAnnotationPtr A smart pointer to the new stamp annotation

getIntent()

```
virtual U8String JawsMako::IStampAnnotation::getIntent ( ) const  [pure virtual]
```

Get the stamp annotation's intent.

Returns

U8String The stamp annotation's intent

See also

IStampAnnotation::getIntent

getName()

```
virtual U8String JawsMako::IStampAnnotation::getName ( ) const  [pure virtual]
```

Get the stamp annotation's name.

Returns

U8String The stamp annotation's name

setIntent()

```
virtual void JawsMako::IStampAnnotation::setIntent (
            const U8String & intent )  [pure virtual]
```

Set the stamp annotation's intent. Note: This is a PDF 2.0 feature. If intent is set to anything other than Stamp (the default), then the name of the stamp annotation will be dropped.

Parameters

intent	The intent

setName()

```
virtual void JawsMako::IStampAnnotation::setName (
            const U8String & name )  [pure virtual]
```

Set the stamp annotation's name.

Parameters

name	The desired name

The documentation for this class was generated from the following file:

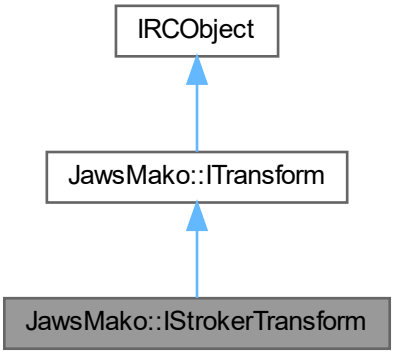
- interactive.h

7.414 JawsMako::IStrokerTransform Class Reference

A transform for converting some or all stroked paths into plain filled paths.

```
#include <transforms.h>
```

Inheritance diagram for JawsMako::IStrokerTransform:



Public Member Functions

- virtual void **setTargetResolution** (float resolution)=0

  Sets the target resolution (dpi) of the eventual consumer. In some situations the stroker will need to "flatten" sections of the path and this parameter will help the stroker decide how much error is allowable in such situations. The default is 300dpi.
- virtual void setupForXPSStyleOutput ()=0

  Sets up the stroker transform for output to an XPS style consumer.
- virtual void setupForPDFStyleOutput ()=0

  Sets up the stroker transform for output to a PDF style consumner.
- virtual void **setConvertAllStrokes** (bool convert)=0

  Sets whether or not the stroker should convert all stroked paths into fills. The default is true.
- virtual void **setConvertForDashedStrokes** (bool convert)=0

  Sets whether or not to convert paths with dashed strokes. The default is false but this is ignored if setConvertAllStrokes() has been set to true.
- virtual void **setConvertForNonTilingVisualOrImageStrokes** (bool convert)=0

  Sets whether or not to convert paths with non-tiling visual or image brushes. The default is false but this is ignored if setConvertAllStrokes() has been set to true. This results in simpler processing for many consumers.
- virtual void **setConvertForMiterTreatment** (IDOMPathNode::eStrokeMiterLimitTreatment treatment, bool convert=true)=0

  Sets whether or not the stroker should convert paths with the given miter treatment mode. Note that if this is set the stroker will not convert paths where the miters would not be seen, such as if none of the corners in the path are acute enough to generate miters longer than the threshold of the path. The default is false in all cases, but this is ignored if setConvertAllStrokes() has been set to true.
- virtual void **setMiterErrorThreshold** (float maxError)=0

  Sets the approximate allowable amount of error, in pixels at the target resolution allowed for miter joins. This does not affect the result of converting a stroke to a filled path, but does affect whether or not to convert if setConvertForMiterTreatment() has been set to true for the applicable miter treatment method. It is designed to allow thin paths where miters would not be easily discernable to be left as is. The default is 0.0.
- virtual void **setConvertForLineCap** (IDOMPathNode::eStrokeLineCap cap, bool convert=true)=0

  Sets whether or not strokes containing the given line cap type should be converted. In all cases the default is false, but this is ignored if setConvertAllStrokes() has been set to true.
- virtual void **setConvertForDashCap** (IDOMPathNode::eStrokeLineCap cap, bool convert=true)=0

Sets whether or not strokes containing the given dash cap type should be converted. In all cases the default is false, but this is ignored if setConvertAllStrokes() has been set to true.

- virtual void **setConvertForMismatchedCaps** (bool convert)=0

  Sets whether or not strokes where the start and end line cap do not match should be converted. The default is false, but ignored if setConvertAllStrokes() has been set to true.

- virtual void **setConvertForMismatchedDashCap** (bool convert)=0

  Sets whether or not strokes where the line is dashed and the dash dash cap does not match both of the line capse. The default is false but ignored if setConvertAllStreoks() has been set to true.

- virtual void **setConvertForJoin** (IDOMPathNode::eStrokeLineJoin join, bool convert=true)=0

  Sets whether or not strokes containing the given line join type should be converted. In all cases the default is false, but this is ignored if setConvertAllStrokes() has been set to true.

- virtual void **setConvertForNonStrokingSegments** (bool convert)=0

  Sets whether or not strokes with any segments marked non-stroking should be converted. The default is false, but this is ignored if setConvertAllStrokes() has been set to true.

Public Member Functions inherited from JawsMako::ITransform

- virtual IDOMBrushPtr transform (const IDOMBrushPtr &brush, eBrushUsage usage=eBUGeneral, const CTransformState &state=CTransformState())=0

  Apply the transform to the given brush, if applicable. These transforms are thread safe.

- virtual IDOMImagePtr transform (const IDOMImagePtr &image, const CTransformState &state=CTransformState())=0

  Apply the transform to the given image, if applicable. These transforms are thread safe.

- virtual IDOMColorPtr transform (const IDOMColorPtr &color, const CTransformState &state=CTransformState())=0

  Apply the transform to the given color, if applicable. These transforms are thread safe.

- virtual IDOMColorSpacePtr transform (const IDOMColorSpacePtr &colorSpace, const CTransformState &state=CTransformState())=0

  Apply the transform to the given colorspace, if applicable. These transforms are thread safe.

- virtual IDOMNodePtr transform (const IDOMNodePtr &node, bool &changed, bool transformChildren=true, const CTransformState &state=CTransformState())=0

  Apply the transform to the given node, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time.

- virtual void transformPage (const IPagePtr &page, bool transformContent=true, bool transform↩ Annotations=true)=0

  Apply the transform to the given page, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time. The transform will also apply to the annotations appearances.

- virtual void **flushCaches** ()=0

  Flush the caches used by the transform. Most transforms cache recently transformed results to improve the performance of repeated transformations of equivalent results. However, it is possible that some cached results may point to entities that no longer exist, such as content inside an XPS file that no longer exists. If you are deleting or replacing files where transforms have been used, it is advisable to invoke this routine to clear the caches.

- virtual void setProgressMonitor (const IProgressMonitorPtr &progressMonitor)=0

  Set the IProgressMonitor object for this transform to allow for monitoring the progress of the transform.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.

- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.

- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IStrokerTransformPtr create (const IJawsMakoPtr &jawsMako, const IAbortPtr &abort=IAbortPtr())

  Create the transform.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.414.1 Detailed Description

A transform for converting some or all stroked paths into plain filled paths.

Different rendering engines have different stroking abilities. XPS for example supports several stroking capabilities that, say, PDF or SVG support, such as triangular line caps, using a different start and end line caps, and the ability to mark sections of a path as non-stroking.

Further, XPS and most other renderers differ in how miters are treated, with XPS "clipping" miters to the miter limit where PDF and SVG renderers would simply bevel instead.

This filter provides a method for converting strokes to plain fills in all cases or only in certain circumstances.

For best results the transform entry in the state given to the transform() routines should be valid.

7.414.2 Member Function Documentation

create()

```
static JAWSMAKO_API IStrokerTransformPtr JawsMako::IStrokerTransform::create (
            const IJawsMakoPtr & jawsMako,
            const IAbortPtr & abort = IAbortPtr() )  [static]
```

Create the transform.

Parameters

jawsMako	The JawsMako instance.
abort	An abort callback handler.

Returns

The new instance.

setupForPDFStyleOutput()

```
virtual void JawsMako::IStrokerTransform::setupForPDFStyleOutput ( )  [pure virtual]
```

Sets up the stroker transform for output to a PDF style consumner.

```
Equivalent to (on a freshly instantiated transform):
   setConvertAllStrokes(false);
   setConvertForMiterTreatment(IDOMPathNode::eClipLongMiters);
   setConvertForLineCap(IDOMPath::eTriangleCap);
   setConvertForDashCap(IDOMPath::eTriangleCap);
   setConvertForMismatchedCaps(true);
   setConvertForMismatchedDashCap(true);
   setConvertForNonStrokingSegments(true);
   setConvertForNonTilingVisualOrImageStrokes(true);
```

setupForXPSStyleOutput()

```
virtual void JawsMako::IStrokerTransform::setupForXPSStyleOutput ( )  [pure virtual]
```

Sets up the stroker transform for output to an XPS style consumer.

```
Equivalent to (on a freshly instantiated transform):
   setConvertAllStrokes(false);
   setConvertForMiterTreatment(IDOMPathNode::eBevelLongMiters);
```

The documentation for this class was generated from the following file:

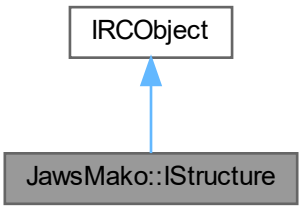
- transforms.h

7.415 JawsMako::IStructure Class Reference

Top level tracking structure describing the logical structure of the document.

```
#include <structure.h>
```

Inheritance diagram for JawsMako::IStructure:



Public Member Functions

- virtual IStructurePtr **clone** () const =0

  Perform a deep clone of the structure.
- virtual IDOMNodePtr tagNode (const IStructureElementPtr &element, const IPagePtr &page, const IDOMNodePtr &nodeToTag)=0

  Tag an IDOMNode from the given page with the given element.
- virtual void **appendElement** (const IStructureElementPtr &element)=0

  Add an element to the end of the children at the top level. An exception will be thrown if the element already exists in the structure tree.
- virtual void **insertElement** (const IStructureElementPtr &element, uint32 index)=0

  Insert an element into the children at the top level at the given index, where an index of 0 is the head of the list. An exception will be thrown if the element already exists in the structure tree.
- virtual void **removeElement** (uint32 index)=0

  Remove the element at the given index from the list of children. Note: This will also cause all child elements to be removed.
- virtual const CStructureElementVect & **getChildElements** () const =0

  Get the child structure elements.
- virtual IStructureElementPtr **findStructureElementWithReference** (const IStructureElementReferencePtr &reference) const =0

  Find the structure element matching the given reference, or NULL if it does not exist.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IStructurePtr **create** (const IJawsMakoPtr &jawsMako)

  Create a new Structure.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.415.1 Detailed Description

Top level tracking structure describing the logical structure of the document.

7.415.2 Member Function Documentation

tagNode()

```
virtual IDOMNodePtr JawsMako::IStructure::tagNode (
            const IStructureElementPtr & element,
            const IPagePtr & page,
            const IDOMNodePtr & nodeToTag )  [pure virtual]
```

Tag an IDOMNode from the given page with the given element.

```
If the node is an IDOMGroup (which isn't already optional content or
tagged with structure information) or IDOMForm, the node will have
it's marked content information set accordingly and the returned result
will be that node.
However, if the node is some other graphical node, it will be moved
inside an IDOMGroup with the optional content information set. If the
node is in a tree, the node will be replaced with the group. The
group will be returned.

The given element must be present in the structure.
```

The documentation for this class was generated from the following file:

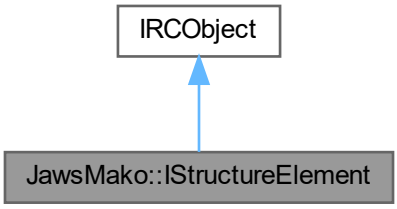
- structure.h

7.416 JawsMako::IStructureElement Class Reference

A structure element in the structure tree.

```
#include <structure.h>
```

Inheritance diagram for JawsMako::IStructureElement:



Public Member Functions

- virtual IStructureElementPtr **clone** () const =0

  Obtain a clone of the element.
- virtual U8String **getStructureType** () const =0

  Get the structure type.
- virtual void **setStructureType** (const U8String &structureType)=0

  Set the structure type. Must be non-zero length.
- virtual void **setTitle** (const U8String &title)=0

  Set the "Title" of the structure element, or an empty string to remove.
- virtual U8String **getLanguage** () const =0

  Get the language specifier for this element, or an empty string if none is specified.
- virtual void **setLanguage** (const U8String &language)=0

  Set the language specifier for this element, or an empty string to remove.
- virtual U8String **getAlternate** () const =0

  Get the alternate description of the structure element, in human readable form. Returns an empty string if none is provided.
- virtual void **setAlternate** (const U8String &alternateDescription)=0

  Set the alternate description of the structure element, or an empty string to remove.
- virtual U8String **getActualText** () const =0

  Get the actual text of the structure element. Returns an empty string if none is provided.
- virtual void **setActualText** (const U8String &actualText)=0

  Set the alternate description of the structure element, or an empty string to remove.
- virtual void setAttributes (const IPDFObjectPtr &attributes)=0

  Set the PDF object representing the attributes of this element. Please see the PDF specification for the format of these objects. Any far references should be retrievable using the object store (see getObjectStore() above)
- virtual IPDFObjectPtr getAttributes () const =0

  Obtain the PDF object representing the attributes of this element. Please see the PDF specification for the format of these objects. Any far references should be retrievable from the IDocument associated with the structure (see IDocument::getObjectStore() for details)
- virtual void **appendElement** (const IStructurePtr &structure, const IStructureElementPtr &element)=0

  Add an element to the end of the children. An exception will be thrown if the element already exists in the structure tree. The structure containing the target element must be provided.
- virtual void **insertElement** (const IStructurePtr &structure, const IStructureElementPtr &element, uint32 index)=0

  Insert an element at the given index, where 0 is the head. An exception will be thrown if an element is added that already exists in the structure tree.
- virtual void **appendMarkedContentReferenceChild** (const IStructureMarkedContentReferenceChildPtr &child)=0

  Add a marked content reference to the end of the children.
- virtual void **insertMarkedContentReferenceChild** (const IStructureMarkedContentReferenceChildPtr &child, uint32 index)=0

  Insert a marked content reference into the children at the given index, where an index of 0 is the head.
- virtual void **appendObjectReferenceChild** (const IPagePtr &page, const IDOMFormPtr &targetForm)=0

  Add an object reference targetting the given form to the end of the chidren. In some situations the form may have its properties edited. Pass in the page (required) that the form will appear on.
- virtual void **insertObjectReferenceChild** (const IPagePtr &page, const IDOMFormPtr &targetForm, uint32 index)=0

  Insert an object reference targetting the given form into the children at the given index, where 0 is the head. In some situations the form may have its properties edited. Pass in the page (required) that the form will appear on.
- virtual void **removeChild** (const IStructurePtr &structure, uint32 index)=0

  Remove the child at the given index from the list of children. Note: This will also cause all child elements of that element to be removed.

- virtual void createMarkedContentReferencePair (const IStructurePtr &structure, const IPagePtr &page, const IDOMNodePtr &node, IStructureMarkedContentReferenceChildPtr &resultReference, IMarkedContent↩ StructureDetailsPtr &resultDetails)=0

  Create a IStructureMarkedContentReferenceChild and an IMarkedContentStructureDetails pair for a node present on a given page. Cannot be used with form nodes. Neither the node nor the target element are edited, but in some situations the parents of the node may have their properties edited.
- virtual const CStructureElementChildVect & **getChildren** () const =0

  Get the child structure information.
- virtual IStructureElementReferencePtr **getStructureElementReference** () const =0

  Obtain a reference for the structure element.
- virtual IStructureElementPtr **findStructureElementWithReference** (const IStructureElementReferencePtr &reference) const =0

  Find the structure element matching the given reference, in the children. Returns NULL if it does not exist.
- virtual IPDFObjectStorePtr getObjectStore ()=0

  Obtain access to the store of objects associated with the structure element. Objects which are controlled by formal Mako APIs may not be reachable, and may be overridden at output.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IStructureElementPtr **create** (const IJawsMakoPtr &jawsMako, const U8String &structureType)

  Create a structure element.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.416.1 Detailed Description

A structure element in the structure tree.

7.416.2 Member Function Documentation

createMarkedContentReferencePair()

```
virtual void JawsMako::IStructureElement::createMarkedContentReferencePair (
            const IStructurePtr & structure,
            const IPagePtr & page,
            const IDOMNodePtr & node,
            IStructureMarkedContentReferenceChildPtr & resultReference,
            IMarkedContentStructureDetailsPtr & resultDetails )  [pure virtual]
```

Create a IStructureMarkedContentReferenceChild and an IMarkedContentStructureDetails pair for a node present on a given page. Cannot be used with form nodes. Neither the node nor the target element are edited, but in some situations the parents of the node may have their properties edited.

See also IStructure::tagNode() for a simplified method for applying structure to an individual node.

getAttributes()

```
virtual IPDFObjectPtr JawsMako::IStructureElement::getAttributes ( ) const  [pure virtual]
```

Obtain the PDF object representing the attributes of this element. Please see the PDF specification for the format of these objects. Any far references should be retrievable from the IDocument associated with the structure (see IDocument::getObjectStore() for details)

Returns

IPDFObjectPtr The attributes, or NULL if not available. If non-null this can be edited freely, but any edits will not take effect unless setAttributes() is used.

getObjectStore()

```
virtual IPDFObjectStorePtr JawsMako::IStructureElement::getObjectStore ( )  [pure virtual]
```

Obtain access to the store of objects associated with the structure element. Objects which are controlled by formal Mako APIs may not be reachable, and may be overridden at output.

Returns

IPDFObjectStorePtr The object store for the document.

setAttributes()

```
virtual void JawsMako::IStructureElement::setAttributes (
            const IPDFObjectPtr & attributes )  [pure virtual]
```

Set the PDF object representing the attributes of this element. Please see the PDF specification for the format of these objects. Any far references should be retrievable using the object store (see getObjectStore() above)

Parameters

attributes	Either a Dictionary, Array or NULL to clear the attributes.

The documentation for this class was generated from the following file:

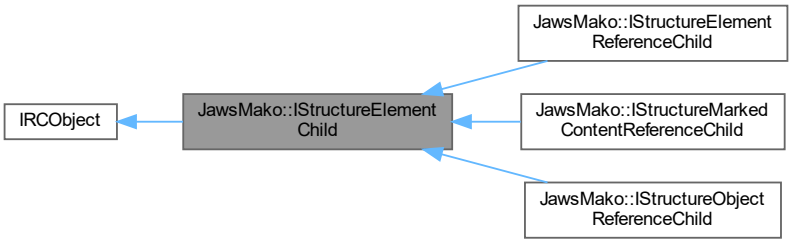
- structure.h

7.417 JawsMako::IStructureElementChild Class Reference

A child of a structure element. Either points to actual marked content, or another structure element.

`#include <structure.h>`

Inheritance diagram for JawsMako::IStructureElementChild:



Public Member Functions

- virtual eStructureChildType **getType** () const =0

  Get the type of this child.
- virtual IStructureElementChildPtr **clone** () const =0

  Clone the element child.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.417.1 Detailed Description

A child of a structure element. Either points to actual marked content, or another structure element.

The documentation for this class was generated from the following file:

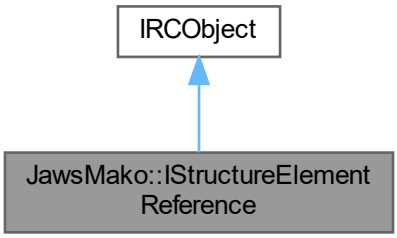
- structure.h

7.418 JawsMako::IStructureElementReference Class Reference

A token-like class encapsulating a reference to a structure element.

`#include <structure.h>`

Inheritance diagram for JawsMako::IStructureElementReference:



Public Member Functions

- virtual bool **equals** (const IStructureElementReferencePtr &other) const =0

  Determine if another structure element reference refers to the same element.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.418.1 Detailed Description

A token-like class encapsulating a reference to a structure element.

The documentation for this class was generated from the following file:

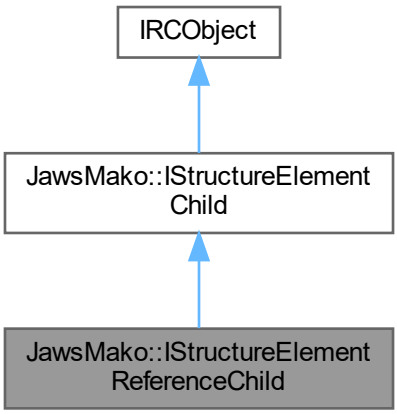
- structure.h

7.419 JawsMako::IStructureElementReferenceChild Class Reference

A child of a structure element that points to another structure element.

`#include <structure.h>`

Inheritance diagram for JawsMako::IStructureElementReferenceChild:



Public Member Functions

- virtual IStructureElementPtr **getElement** () const =0

  Obtain a reference to this structure element.

Public Member Functions inherited from JawsMako::IStructureElementChild

- virtual eStructureChildType **getType** () const =0

  Get the type of this child.
- virtual IStructureElementChildPtr **clone** () const =0

  Clone the element child.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.419.1 Detailed Description

A child of a structure element that points to another structure element.

The documentation for this class was generated from the following file:

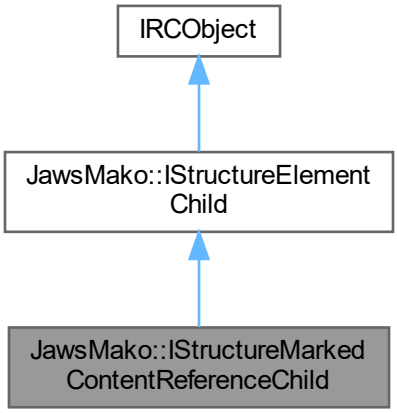
- structure.h

7.420 JawsMako::IStructureMarkedContentReferenceChild Class Reference

A child of a structure element that points to a piece of marked content. Note; to create these, please see IStructureElement::createMarkedContentReferencePair()

`#include <structure.h>`

Inheritance diagram for JawsMako::IStructureMarkedContentReferenceChild:



Public Member Functions

- virtual bool **matches** (const IStructureElementPtr &parentElement, const IMarkedContentStructureDetailsPtr &details) const =0

  Check to see if the given IMarkedContentStructureDetails object matches this reference. That is, does this child refer to a group with these details. Pass in the parent structure element; it is needed to check that the values match.

Public Member Functions inherited from JawsMako::IStructureElementChild

- virtual eStructureChildType **getType** () const =0

  Get the type of this child.
- virtual IStructureElementChildPtr **clone** () const =0

  Clone the element child.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.420.1 Detailed Description

A child of a structure element that points to a piece of marked content. Note; to create these, please see IStructureElement::createMarkedContentReferencePair()

The documentation for this class was generated from the following file:

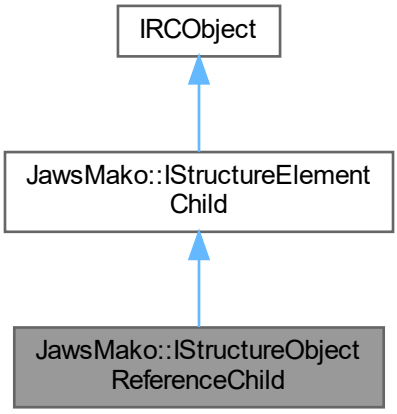
- structure.h

7.421 JawsMako::IStructureObjectReferenceChild Class Reference

A child of a structure element that points to a piece of marked content. These cannot be created directly. Instead use IStructureElement::appendObjectReferenceChild() or IStructureElement::insertObjectReferenceChild()

`#include <structure.h>`

Inheritance diagram for JawsMako::IStructureObjectReferenceChild:



Public Member Functions

- virtual bool **matches** (const IDOMFormPtr &form) const =0

  Check to see if the given IDOMForm is pointed to by this object reference.

Public Member Functions inherited from JawsMako::IStructureElementChild

- virtual eStructureChildType **getType** () const =0

  Get the type of this child.
- virtual IStructureElementChildPtr **clone** () const =0

  Clone the element child.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.421.1 Detailed Description

A child of a structure element that points to a piece of marked content. These cannot be created directly. Instead use IStructureElement::appendObjectReferenceChild() or IStructureElement::insertObjectReferenceChild()

The documentation for this class was generated from the following file:

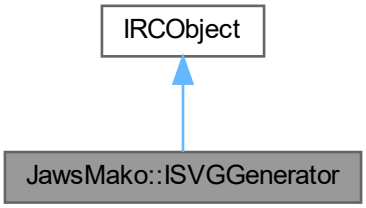
- structure.h

7.422 JawsMako::ISVGGenerator Class Reference

An SVG generator for JawsMako, allowing simple generation of SVG fragments for individual DOM nodes or entire pages.

```
#include <svggenerator.h>
```

Inheritance diagram for JawsMako::ISVGGenerator:



Classes

- class CResourceEntry

  Resource entry.

Public Types

- enum SVGVersion

  The SVG available versions for which the output is generated.

Public Member Functions

- virtual IRAInputStreamPtr generateSVG (const IDOMNodePtr &node)=0

  Generate SVG for the given DOM Node, returning the result in a stream.
- virtual void generateSVG (const IDOMNodePtr &node, const IOutputStreamPtr &outputStream)=0

  Alternate form of generateSVG() when an existing stream should be used.
- virtual IInputStreamPtr getResource (const U8String &name)=0

  Get the stream for a named resource. An exception will be thrown if the resource cannot be found.
- virtual void getResources (CSVGResourceVect &resources)=0

  Get all the resources in a vector.
- virtual void **setVersion** (SVGVersion version)=0

  Set the output version of the generated SVG. The default is 1.1.
- virtual void **setResolution** (float resolution)=0

  Set the target resolution for display. The default is 96dpi.
- virtual void setEnableImageDownsampling (bool downsample)=0

  Enable or disable image downsampling.
- virtual void setOptionalContent (const IOptionalContentPtr &optionalContent)=0

  Set the optional content for the document.
- virtual void setOptionalContentUsage (eOptionalContentEvent usage)=0

  Set the usage of the optional content items that should be retained.
- virtual void setTargetResolutionCallback (void *priv, UrlCallbackForTarget callback)=0

  Set a callback to provide the Url (absolute or relative) of the object where a target may be found. Return either the URL of the containing object, or an empty string if no such target exists.
- virtual void setPageResolutionCallback (void *priv, UrlCallbackForPage callback)=0

  Set a callback to provide the Url (absolute or relative) of the given page number. Return either the URL of the page, or an empty string if the page will not be reachable.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API ISVGGeneratorPtr create (const IJawsMakoPtr &jawsMako, const U8String &resourcePrefix=U8String(), const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create a SVG generator instance.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.422.1 Detailed Description

An SVG generator for JawsMako, allowing simple generation of SVG fragments for individual DOM nodes or entire pages.

The SVG is provided in a stream, with resources used by the SVG presented as streams tracked by instances of this object, which can be requested via getResource(). Resources are reused where possible for multiple SVG fragments.

7.422.2 Member Function Documentation

create()

```
static JAWSMAKO_API ISVGGeneratorPtr JawsMako::ISVGGenerator::create (
            const IJawsMakoPtr & jawsMako,
            const U8String & resourcePrefix = U8String(),
            const IProgressMonitorPtr & progressMonitor = IProgressMonitorPtr() )  [static]
```

Create a SVG generator instance.

Parameters

jawsMako	The JawsMako instance.
resourcePrefix	A path fragment to prepend to resource names when generating SVG. Useful for cases where the resources are to be written to disk somewhere else than alongside the SVG fragment.
progressMonitor	The progress monitor which allows aborting an operation or registering a progress callback.

Returns

ISVGGeneratorPtr The new instance.

generateSVG() [1/2]

```
virtual IRAInputStreamPtr JawsMako::ISVGGenerator::generateSVG (
            const IDOMNodePtr & node )  [pure virtual]
```

Generate SVG for the given DOM Node, returning the result in a stream.

Parameters

node	The DOM node to be represented as SVG.

Returns

IRAInputStreamPtr The SVG stream.

generateSVG() [2/2]

```
virtual void JawsMako::ISVGGenerator::generateSVG (
            const IDOMNodePtr & node,
            const IOutputStreamPtr & outputStream )  [pure virtual]
```

Alternate form of generateSVG() when an existing stream should be used.

Parameters

node	The DOM node to be represented as SVG.
outputStream	The destination stream for the SVG.

getResource()

```
virtual IInputStreamPtr JawsMako::ISVGGenerator::getResource (
            const U8String & name )  [pure virtual]
```

Get the stream for a named resource. An exception will be thrown if the resource cannot be found.

Parameters

name	The name of the resource.

Returns

IInputStreamPtr The resource stream.

getResources()

```
virtual void JawsMako::ISVGGenerator::getResources (
            CSVGResourceVect & resources )  [pure virtual]
```

Get all the resources in a vector.

Parameters

resources	A reference to a vector to receive the entries.

setEnableImageDownsampling()

```
virtual void JawsMako::ISVGGenerator::setEnableImageDownsampling (
            bool downsample )  [pure virtual]
```

Enable or disable image downsampling.

If true, images whose resolution is 50% higher than the target resolution (see setResolution()) will be downsampled to the target resolution.

If false, images will not be downsampled.

The default is true.

setOptionalContent()

```
virtual void JawsMako::ISVGGenerator::setOptionalContent (
            const IOptionalContentPtr & optionalContent )  [pure virtual]
```

Set the optional content for the document.

If provided, the generator will apply optional content rules to the content during generation. If NULL, all content will be written to the output stream.

setOptionalContentUsage()

```
virtual void JawsMako::ISVGGenerator::setOptionalContentUsage (
            eOptionalContentEvent usage )  [pure virtual]
```

Set the usage of the optional content items that should be retained.

Setting the usage to eOCEUnknown will retain all items if no optional content is set with setOptionalContent(), otherwise it will throw an IError of code EDL_ERR_BAD_ARGUMENTS.

The default is eOCEView.

setPageResolutionCallback()

```
virtual void JawsMako::ISVGGenerator::setPageResolutionCallback (
            void * priv,
            UrlCallbackForPage callback )  [pure virtual]
```

Set a callback to provide the Url (absolute or relative) of the given page number. Return either the URL of the page, or an empty string if the page will not be reachable.

The priv argument will be provided to the callback at each invocation.

See svggenerator.cpp for an example of its use.

setTargetResolutionCallback()

```
virtual void JawsMako::ISVGGenerator::setTargetResolutionCallback (
            void * priv,
            UrlCallbackForTarget callback )  [pure virtual]
```

Set a callback to provide the Url (absolute or relative) of the object where a target may be found. Return either the URL of the containing object, or an empty string if no such target exists.

The priv argument will be provided to the callback at each invocation.

See svggenerator.cpp for an example of its use.

The documentation for this class was generated from the following file:

- svggenerator.h

7.423 Iterator Class Reference

A convenience interator-like class for iterating through the contents of a dictionary.

`#include <pdfobjects.h>`

7.423.1 Detailed Description

A convenience interator-like class for iterating through the contents of a dictionary.

The documentation for this class was generated from the following file:

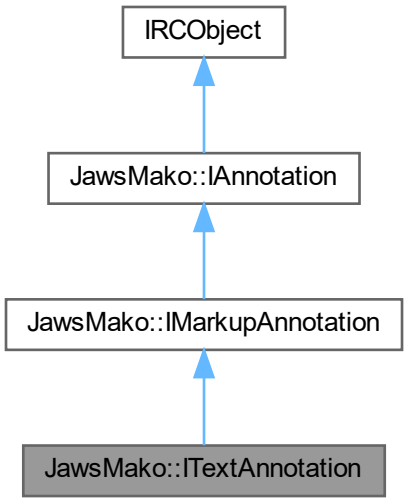
- pdfobjects.h

7.424 JawsMako::ITextAnnotation Class Reference

A generic interface class for a text (sticky note) annotation.

`#include <interactive.h>`

Inheritance diagram for JawsMako::ITextAnnotation:



Public Member Functions

- virtual bool getOpen () const =0

  Get the annotation's open status.
- virtual void setOpen (bool open)=0

  Set the annotation's open status.
- virtual U8String getIconName () const =0

  Get the annotation's icon name. Must not be empty.
- virtual void setIconName (const U8String &iconName)=0

  Set the annotation's icon name.

Public Member Functions inherited from JawsMako::IMarkupAnnotation

- virtual U8String getAuthor () const =0

  Get the Author of the markup annotation. This is the "T" entry in the annotation, which by convention is the author.
- virtual void setAuthor (const U8String &author)=0

  Set the Author of the markup annotation. This is the "T" entry in the annotation, which by convention is the author.
- virtual IEDLTimePtr getCreationTime () const =0

  Get the creation date and time of the annotation, if present.
- virtual void setCreationTime (const IEDLTimePtr &creationTime)=0

  Set the creation date and time of the annotation.
- virtual float getOpacity () const =0

  Get the opacity of the markup annotation.
- virtual void setOpacity (float opacity)=0

  Set the opacity of the markup annotation.
- virtual IAnnotationReferencePtr getPopupReference () const =0

  Get a reference to the popup, if present.
- virtual void setPopup (const IAnnotationReferencePtr &popupReference)=0

  Set or clear the popup, by reference.
- virtual void setPopup (const IPopupAnnotationPtr &popup)=0

  Set a reference to a popup, if present.
- virtual IAnnotationAppearancePtr createNormalAppearance () const =0

  Create a basic appearance given the current annotation's parameters. This can then be installed by using IAnnotation::addAppearance().

Public Member Functions inherited from JawsMako::IAnnotation

- virtual eAnnotationType getType () const =0

  Get the type of the annotation.
- virtual const FRect & getRect () const =0

  Get the rect in which the appearances should be displayed.
- virtual void setRect (const FRect &rect)=0

  Set the rect in which the appearances should be displayed.
- virtual U8String getContents () const =0

  Get the Contents entry in UTF-8.
- virtual void setContents (const U8String &contents)=0

  Set the Contents entry in UTF-8.
- virtual IDOMColorPtr getColor () const =0

  Get the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.
- virtual void setColor (const IDOMColorPtr &color)=0

Set the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.

- virtual IEDLTimePtr getModificationTime () const =0

  Get the Modification date and time of the annotation, if present.
- virtual void setModificationTime (const IEDLTimePtr &modificationTime)=0

  Set the Modification date and time of the annotation.
- virtual CAnnotationBorder getBorder () const =0

  Get the annotation's border. See CAnnotationBorder for details.
- virtual void setBorder (const CAnnotationBorder &border)=0

  Set the annotation's border.
- virtual uint32 getFlags () const =0

  Get the annotation flags. To interpret these flags please see section 8.4.2 "Annotation Flags" in the PDF 1.7 specification.
- virtual void setFlags (uint32 flags)=0

  Set the annotation flags. Please see section 8.4.2 "Annotation Flags" for details about these flags.
- virtual void rotate (uint16 rotate, double pageWidth, double pageHeight)=0

  Rotate the annotation clockwise as if the page was rotated by the same amount.
- virtual CAnnotationAppearanceVect getAppearances () const =0

  Return all the annotation appearances in a vector.
- virtual void **removeAppearances** ()=0

  Remove all annotation appearances.
- virtual U8String getState () const =0

  Get the current annotation state.
- virtual void setState (const U8String &state)=0

  Set the current annotation state.
- virtual IAnnotationAppearancePtr getAppearance (eAppearanceUsage usage, const U8String &state=U8String()) const =0

  Fetch the annotation appearance that should be used for the given annotation usage and state according to the following rules:
- virtual void addAppearance (const IAnnotationAppearancePtr &appearance)=0

  Add or replace an appearance.
- virtual bool hasNormalAppearance () const =0

  Does the annotation have a normal appearance? Convenience utility function.
- virtual IAnnotationPtr clone () const =0

  Clone the annotation. This is a deep clone. The annotation reference will remain the same.
- virtual bool matchesReference (const IAnnotationReferencePtr &reference) const =0

  Does this annotation match the given IAnnotationReference?
- virtual IAnnotationReferencePtr getReference () const =0

  Get a reference that can be used to refer to this annotation.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API ITextAnnotationPtr create (const IJawsMakoPtr &jawsMako, const FRect &rect, const U8String &contents, bool open=false, const U8String &iconName=U8String("Note"), const IPopupAnnotationPtr &popup=IPopupAnnotationPtr())

  Create a text annotation.

Additional Inherited Members

Public Types inherited from JawsMako::IAnnotation

- enum eAnnotationType {
  eAT3D , eATCaret , eATCircle , eATFileAttachment ,
  eATFreeText , eATHighlight , eATInk , eATLine ,
  eATLink , eATMovie , eATPolygon , eATPolyLine ,
  eATPopup , eATPrinterMark , eATProjection , eATRedact ,
  eATRichMedia , eATScreen , eATSound , eATSquare ,
  eATSquiggly , eATStamp , eATStrikeOut , eATText ,
  eATTrapNet , eATUnderline , eATWatermark , eATWidget ,
  eATOther }

  Types of annotations, listed with the earliest version of PDF that supported them.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.424.1 Detailed Description

A generic interface class for a text (sticky note) annotation.

7.424.2 Member Function Documentation

create()

```
static JAWSMAKO_API ITextAnnotationPtr JawsMako::ITextAnnotation::create (
            const IJawsMakoPtr & jawsMako,
            const FRect & rect,
            const U8String & contents,
            bool open = false,
            const U8String & iconName = U8String("Note"),
            const IPopupAnnotationPtr & popup = IPopupAnnotationPtr() )  [static]
```

Create a text annotation.

Parameters

jawsMako	The JawsMako instance
rect	The annotations bounds. Must not be empty.
contents	The textual contents of the text annotation.
open	Optional; whether or not the note is open. Default is false.
iconName	Optional; the name of the icon to use for displaying the annotation. The default is "Note". Must not be empty.
popup	The associated popup. The associated popup must be added to the annotations of the same page.

Returns

ITextAnnotationPtr A smart pointer to the new text annotation

getIconName()

```
virtual U8String JawsMako::ITextAnnotation::getIconName ( ) const  [pure virtual]
```

Get the annotation's icon name. Must not be empty.

Returns

U8String The annotation's icon name

getOpen()

```
virtual bool JawsMako::ITextAnnotation::getOpen ( ) const  [pure virtual]
```

Get the annotation's open status.

Returns

bool True if open, false if closed

setIconName()

```
virtual void JawsMako::ITextAnnotation::setIconName (
            const U8String & iconName )  [pure virtual]
```

Set the annotation's icon name.

Parameters

iconName	the annotation's icon name. Cannot be empty

setOpen()

```
virtual void JawsMako::ITextAnnotation::setOpen (
            bool open )  [pure virtual]
```

Set the annotation's open status.

Parameters

open	Set to true for open, false for closed

The documentation for this class was generated from the following file:

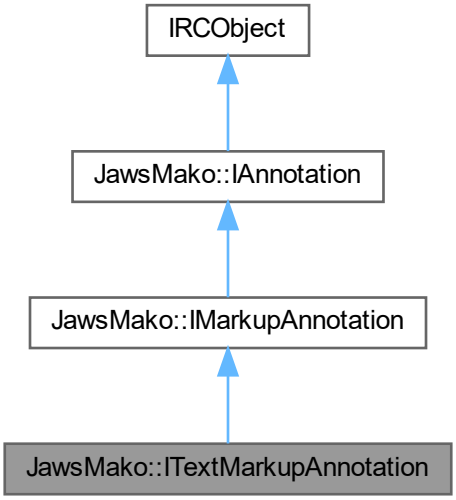
- interactive.h

7.425 JawsMako::ITextMarkupAnnotation Class Reference

A generic interface class for a text markup annotation It is intended that future releases of JawsMako will extend this interface.

`#include <interactive.h>`

Inheritance diagram for JawsMako::ITextMarkupAnnotation:



Public Member Functions

- virtual CQuadPointVect getQuadPoints () const =0

  Get the markup annotation's quad points if present.
- virtual void setQuadPoints (const CQuadPointVect &quadPoints)=0

  Set the markup annotation's quad points.

Public Member Functions inherited from JawsMako::IMarkupAnnotation

- virtual U8String getAuthor () const =0

  Get the Author of the markup annotation. This is the "T" entry in the annotation, which by convention is the author.
- virtual void setAuthor (const U8String &author)=0

  Set the Author of the markup annotation. This is the "T" entry in the annotation, which by convention is the author.
- virtual IEDLTimePtr getCreationTime () const =0

  Get the creation date and time of the annotation, if present.

- virtual void setCreationTime (const IEDLTimePtr &creationTime)=0

  Set the creation date and time of the annotation.
- virtual float getOpacity () const =0

  Get the opacity of the markup annotation.
- virtual void setOpacity (float opacity)=0

  Set the opacity of the markup annotation.
- virtual IAnnotationReferencePtr getPopupReference () const =0

  Get a reference to the popup, if present.
- virtual void setPopup (const IAnnotationReferencePtr &popupReference)=0

  Set or clear the popup, by reference.
- virtual void setPopup (const IPopupAnnotationPtr &popup)=0

  Set a reference to a popup, if present.
- virtual IAnnotationAppearancePtr createNormalAppearance () const =0

  Create a basic appearance given the current annotation's parameters. This can then be installed by using IAnnotation::addAppearance().

Public Member Functions inherited from JawsMako::IAnnotation

- virtual eAnnotationType getType () const =0

  Get the type of the annotation.
- virtual const FRect & getRect () const =0

  Get the rect in which the appearances should be displayed.
- virtual void setRect (const FRect &rect)=0

  Set the rect in which the appearances should be displayed.
- virtual U8String getContents () const =0

  Get the Contents entry in UTF-8.
- virtual void setContents (const U8String &contents)=0

  Set the Contents entry in UTF-8.
- virtual IDOMColorPtr getColor () const =0

  Get the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.
- virtual void setColor (const IDOMColorPtr &color)=0

  Set the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.
- virtual IEDLTimePtr getModificationTime () const =0

  Get the Modification date and time of the annotation, if present.
- virtual void setModificationTime (const IEDLTimePtr &modificationTime)=0

  Set the Modification date and time of the annotation.
- virtual CAnnotationBorder getBorder () const =0

  Get the annotation's border. See CAnnotationBorder for details.
- virtual void setBorder (const CAnnotationBorder &border)=0

  Set the annotation's border.
- virtual uint32 getFlags () const =0

  Get the annotation flags. To interpret these flags please see section 8.4.2 "Annotation Flags" in the PDF 1.7 specification.
- virtual void setFlags (uint32 flags)=0

  Set the annotation flags. Please see section 8.4.2 "Annotation Flags" for details about these flags.
- virtual void rotate (uint16 rotate, double pageWidth, double pageHeight)=0

  Rotate the annotation clockwise as if the page was rotated by the same amount.
- virtual CAnnotationAppearanceVect getAppearances () const =0

  Return all the annotation appearances in a vector.
- virtual void **removeAppearances** ()=0

Remove all annotation appearances.

- virtual U8String getState () const =0

  Get the current annotation state.
- virtual void setState (const U8String &state)=0

  Set the current annotation state.
- virtual IAnnotationAppearancePtr getAppearance (eAppearanceUsage usage, const U8String &state=U8String()) const =0

  Fetch the annotation appearance that should be used for the given annotation usage and state according to the following rules:
- virtual void addAppearance (const IAnnotationAppearancePtr &appearance)=0

  Add or replace an appearance.
- virtual bool hasNormalAppearance () const =0

  Does the annotation have a normal appearance? Convenience utility function.
- virtual IAnnotationPtr clone () const =0

  Clone the annotation. This is a deep clone. The annotation reference will remain the same.
- virtual bool matchesReference (const IAnnotationReferencePtr &reference) const =0

  Does this annotation match the given IAnnotationReference?
- virtual IAnnotationReferencePtr getReference () const =0

  Get a reference that can be used to refer to this annotation.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Public Types inherited from JawsMako::IAnnotation

- enum eAnnotationType {
  eAT3D , eATCaret , eATCircle , eATFileAttachment ,
  eATFreeText , eATHighlight , eATInk , eATLine ,
  eATLink , eATMovie , eATPolygon , eATPolyLine ,
  eATPopup , eATPrinterMark , eATProjection , eATRedact ,
  eATRichMedia , eATScreen , eATSound , eATSquare ,
  eATSquiggly , eATStamp , eATStrikeOut , eATText ,
  eATTrapNet , eATUnderline , eATWatermark , eATWidget ,
  eATOther }

  Types of annotations, listed with the earliest version of PDF that supported them.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.425.1 Detailed Description

A generic interface class for a text markup annotation It is intended that future releases of JawsMako will extend this interface.

7.425.2 Member Function Documentation

getQuadPoints()

```
virtual CQuadPointVect JawsMako::ITextMarkupAnnotation::getQuadPoints ( ) const  [pure virtual]
```

Get the markup annotation's quad points if present.

Returns

CQuadPointVect The markup annotation's quad points

setQuadPoints()

```
virtual void JawsMako::ITextMarkupAnnotation::setQuadPoints (
            const CQuadPointVect & quadPoints )  [pure virtual]
```

Set the markup annotation's quad points.

Parameters

quadPoints	The markup annotation's quad points

The documentation for this class was generated from the following file:

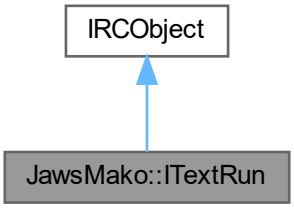
- interactive.h

7.426 JawsMako::ITextRun Class Reference

A run of text, containing unicode information, the position, transformation and bounds of the text.

```
#include <text.h>
```

Inheritance diagram for JawsMako::ITextRun:



Public Member Functions

- virtual const String & **getUnicode** () const =0

  Get the Unicode string for the run as available.
- virtual U8String **getUTF8** () const =0

  Get the Unicode string for the run as available, in UTF-8.
- virtual FMatrix **getTransform** () const =0

  Get the transformation active where the glyph run is positioned, relative to the page. The transformation includes any transformation present in the IDOMGlyphs node where the text is represented.
- virtual FPoint **getLocalOrigin** () const =0

  Get the origin of the first character, in glyph-local coordinates.
- FPoint **getOriginOnPage** () const

  Get the origin of the first character, in page coordinates. Convenience.
- virtual FRect getLocalBounds (bool tight=true) const =0

  Get the bounds of the run, in glyph-local coordinates.
- FRect getBoundsOnPage (bool tight=true) const

  Get the bounds of the run, in page coordinates. Convenience.
- virtual void **getCornersOnPage** (FPoint &p1, FPoint &p2, FPoint &p3, FPoint &p4, bool tight) const =0

  Get the corner points of the run (in the clockwise order), in page coordinates.
- virtual CTextRunVect **split** ()=0

  Split the run into the smallest possible units, returning the split runs. If the run cannot be split, the result will be a vector containing this run.
- virtual double **getSpaceWidth** () const =0

  Determine the width of a space in the font used for this run. If the font does not have a space character then an estimate will be returned. The returned value will be in ems.
- virtual double **getSpaceWidthOnPage** () const =0

  Determine the width of a space in the font used for this run. If the font does not have a space character then an estimate will be returned. The returned value will be in page units.
- virtual double **getFontHeightOnPage** () const =0

  Determine the height of the font used for this run. The returned value will be in page units.
- virtual IDOMGlyphsPtr **getGlyphs** () const =0

  Get the IDOMGlyphs node that represents this run.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.426.1 Detailed Description

A run of text, containing unicode information, the position, transformation and bounds of the text.

7.426.2 Member Function Documentation

getBoundsOnPage()

```
FRect JawsMako::ITextRun::getBoundsOnPage (
            bool tight = true ) const  [inline]
```

Get the bounds of the run, in page coordinates. Convenience.

Parameters

tight	Whether the bounding box should be 'tight' around the actual glyph markup, or expanded to include the full em height.

getLocalBounds()

```
virtual FRect JawsMako::ITextRun::getLocalBounds (
            bool tight = true ) const  [pure virtual]
```

Get the bounds of the run, in glyph-local coordinates.

Parameters

tight	Whether the bounding box should be 'tight' around the actual glyph markup, or expanded to include the full em height.

The documentation for this class was generated from the following file:

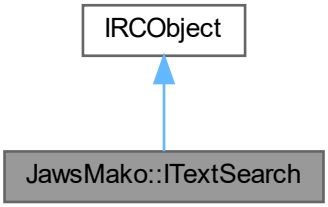
- text.h

7.427 JawsMako::ITextSearch Class Reference

Perform text searching using the page information obtained from an IPageLayout.

`#include <text.h>`

Inheritance diagram for JawsMako::ITextSearch:



Public Member Functions

- virtual CFPointVectVect **search** (const String &targetText, CEDLStringVect &found, bool caseSensitive, bool ignoreSpaces) const =0

  Return a collection of quadpoint data covering all found text. Each entry in the CFPointVectVect represents a unique search 'hit'.
- virtual CFPointVectVect **search** (const String &targetText, CEDLSysStringVect &found, bool caseSensitive, bool ignoreSpaces) const =0

  Return a collection of quadpoint data covering all found text. Each entry in the CFPointVectVect represents a unique search 'hit'.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API ITextSearchPtr create (IEDLClassFactory *factory, const IPageLayoutPtr &page↩Layout)

  Creation function for ITextSearch that performs text searching using page information obtained from an IPageLayout. Throws an IEDLError on failure.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.427.1 Detailed Description

Perform text searching using the page information obtained from an IPageLayout.

7.427.2 Member Function Documentation

create()

```
static JAWSMAKO_API ITextSearchPtr JawsMako::ITextSearch::create (
            IEDLClassFactory * factory,
            const IPageLayoutPtr & pageLayout )  [static]
```

Creation function for ITextSearch that performs text searching using page information obtained from an IPageLayout. Throws an IEDLError on failure.

Parameters

factory	The factory to use
pageLayout	The page layout to use

Returns

ITextSearchPtr The new text searcher

The documentation for this class was generated from the following file:

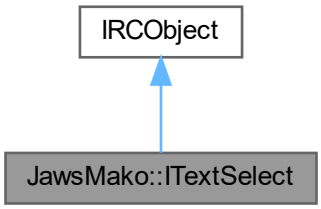
- text.h

7.428 JawsMako::ITextSelect Class Reference

Perform text selection using the page information obtained from an IPageLayout.

```
#include <text.h>
```

Inheritance diagram for JawsMako::ITextSelect:



Public Member Functions

- virtual String **getTextAtRect** (const FRect &pageArea, eLanguageType language=eLTArabic) const =0

  Return the unicode string located within the bounds of 'pageArea', using the specified 'language' unicode helper. Result will be empty if no string is found.
- virtual CFPointVect **selectArea** (const FRect &pageArea, String *selectedText) const =0

  Return quadpoint data covering all text within the specified page area, optionally returning the actual text as well.
- virtual CFPointVect **selectLines** (const FPoint &startPoint, const FPoint &endPoint, String *selectedText) const =0

  Return quadpoint data covering all text within the specified page start/end points, optionally returning the actual text as well.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API ITextSelectPtr create (IEDLClassFactory *factory, const IPageLayoutPtr &page↩ Layout)

  Creation function for an ITextSelect that performs text selection using page information obtained from an IPageLayout. Throws an IEDLError on failure.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.428.1 Detailed Description

Perform text selection using the page information obtained from an IPageLayout.

7.428.2 Member Function Documentation

create()

```
static JAWSMAKO_API ITextSelectPtr JawsMako::ITextSelect::create (
            IEDLClassFactory * factory,
            const IPageLayoutPtr & pageLayout )  [static]
```

Creation function for an ITextSelect that performs text selection using page information obtained from an IPageLayout. Throws an IEDLError on failure.

Parameters

factory	The factory to use
pageLayout	The page layout to use

Returns

ITextSelectPtr The new text selector

The documentation for this class was generated from the following file:

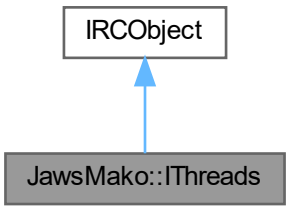
- text.h

7.429 JawsMako::IThreads Class Reference

An interface class for document threads, Currently a stub interface.

```
#include <interactive.h>
```

Inheritance diagram for JawsMako::IThreads:



Additional Inherited Members

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.429.1 Detailed Description

An interface class for document threads, Currently a stub interface.

The documentation for this class was generated from the following file:

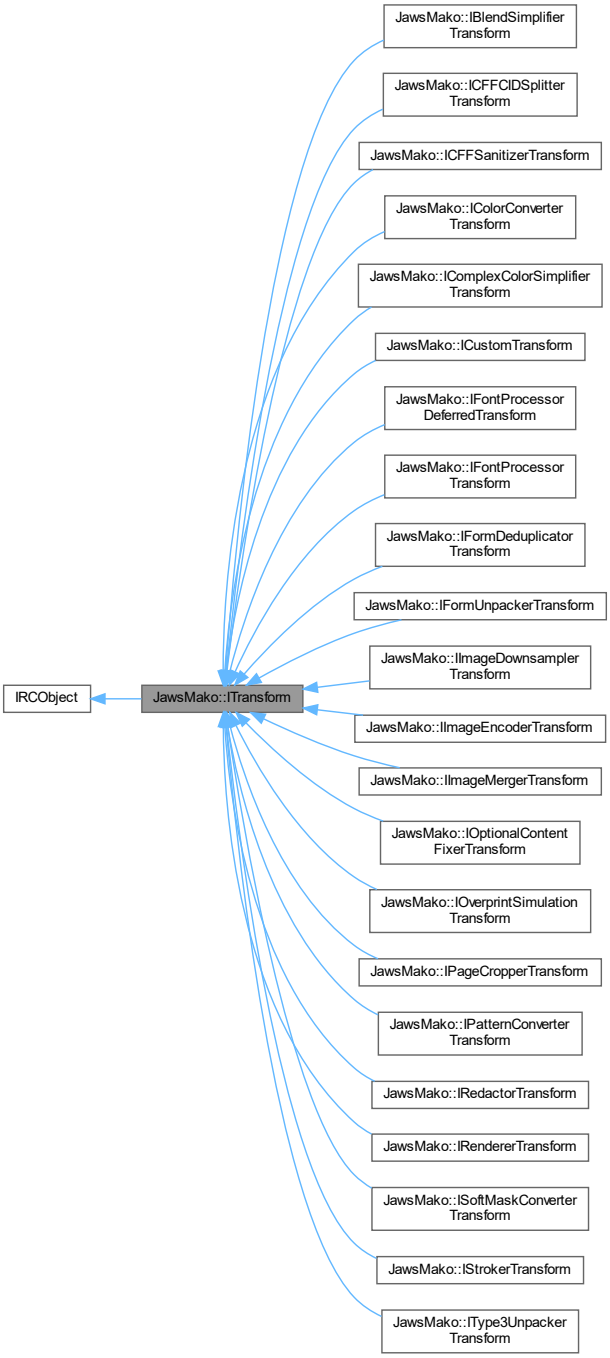
- interactive.h

7.430 JawsMako::ITransform Class Reference

ITransforms provide a method of applying common operations on DOM objects such as brushes, nodes, colors, colorspaces or entire trees. Not all transforms will operate on all kinds of objects, as noted in their descriptions.

```
#include <transforms.h>
```

Inheritance diagram for JawsMako::ITransform:



Public Member Functions

- virtual IDOMBrushPtr transform (const IDOMBrushPtr &brush, eBrushUsage usage=eBUGeneral, const CTransformState &state=CTransformState())=0

  Apply the transform to the given brush, if applicable. These transforms are thread safe.
- virtual IDOMImagePtr transform (const IDOMImagePtr &image, const CTransformState &state=CTransformState())=0

  Apply the transform to the given image, if applicable. These transforms are thread safe.

- virtual IDOMColorPtr transform (const IDOMColorPtr &color, const CTransformState &state=CTransformState())=0

  Apply the transform to the given color, if applicable. These transforms are thread safe.
- virtual IDOMColorSpacePtr transform (const IDOMColorSpacePtr &colorSpace, const CTransformState &state=CTransformState())=0

  Apply the transform to the given colorspace, if applicable. These transforms are thread safe.
- virtual IDOMNodePtr transform (const IDOMNodePtr &node, bool &changed, bool transformChildren=true, const CTransformState &state=CTransformState())=0

  Apply the transform to the given node, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time.
- virtual void transformPage (const IPagePtr &page, bool transformContent=true, bool transform↩ Annotations=true)=0

  Apply the transform to the given page, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time. The transform will also apply to the annotations appearances.
- virtual void **flushCaches** ()=0

  Flush the caches used by the transform. Most transforms cache recently transformed results to improve the performance of repeated transformations of equivalent results. However, it is possible that some cached results may point to entities that no longer exist, such as content inside an XPS file that no longer exists. If you are deleting or replacing files where transforms have been used, it is advisable to invoke this routine to clear the caches.
- virtual void setProgressMonitor (const IProgressMonitorPtr &progressMonitor)=0

  Set the IProgressMonitor object for this transform to allow for monitoring the progress of the transform.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.430.1 Detailed Description

ITransforms provide a method of applying common operations on DOM objects such as brushes, nodes, colors, colorspaces or entire trees. Not all transforms will operate on all kinds of objects, as noted in their descriptions.

ITransform instances attempt to return the same object for cases where identical transforms have been applied. For example, if an image has been previously transformed, the same object will be returned as the previous call if the results would be identical.

7.430.2 Member Function Documentation

setProgressMonitor()

```
virtual void JawsMako::ITransform::setProgressMonitor (
            const IProgressMonitorPtr & progressMonitor )  [pure virtual]
```

Set the IProgressMonitor object for this transform to allow for monitoring the progress of the transform.

Parameters

progressMonitor	The progress monitor which allows aborting an operation or registering a progress callback.

transform() [1/5]

```
virtual IDOMBrushPtr JawsMako::ITransform::transform (
            const IDOMBrushPtr & brush,
            eBrushUsage usage = eBUGeneral,
            const CTransformState & state = CTransformState() )  [pure virtual]
```

Apply the transform to the given brush, if applicable. These transforms are thread safe.

Parameters

brush	The brush to be processed.
usage	What the brush is to be used for.
state	The active CTransformState when this brush is encountered. Not all transforms require this information.

Returns

IDOMBrushPtr The result. If modifications have been applied the result will be a new brush. If NULL, this indicates that the brush should be dropped entirely.

transform() [2/5]

```
virtual IDOMColorPtr JawsMako::ITransform::transform (
            const IDOMColorPtr & color,
            const CTransformState & state = CTransformState() )  [pure virtual]
```

Apply the transform to the given color, if applicable. These transforms are thread safe.

Parameters

color	The color to be processed.
state	The active CTransformState when this color is encountered. Not all transforms require this information.

Returns

IDOMColorPtr The result. If modifications have been applied the result will be a new color. If NULL, this indicates that the color should be dropped entirely.

transform() [3/5]

```
virtual IDOMColorSpacePtr JawsMako::ITransform::transform (
            const IDOMColorSpacePtr & colorSpace,
            const CTransformState & state = CTransformState() )  [pure virtual]
```

Apply the transform to the given colorspace, if applicable. These transforms are thread safe.

Parameters

colorSpace	The colorSpace to be processed.
state	The active CTransformState when this color space is encountered. Not all transforms require this information.

Returns

IDOMColorSpacePtr The result. If modifications have been applied the result will be a new color space.

transform() [4/5]

```
virtual IDOMImagePtr JawsMako::ITransform::transform (
            const IDOMImagePtr & image,
            const CTransformState & state = CTransformState() )  [pure virtual]
```

Apply the transform to the given image, if applicable. These transforms are thread safe.

Parameters

image	The image to be processed.
state	The active CTransformState when this image is encountered. Not all transforms require this information.

Returns

IDOMImagehPtr The result. If modifications have been applied the result will be a new image. If NULL, this indicates that the image should be dropped entirely.

transform() [5/5]

```
virtual IDOMNodePtr JawsMako::ITransform::transform (
            const IDOMNodePtr & node,
            bool & changed,
            bool transformChildren = true,
            const CTransformState & state = CTransformState() )  [pure virtual]
```

Apply the transform to the given node, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time.

Parameters

node	The node to be processed.
changed	Will be set to true on return if any changes have been made.
transformChildren	If true, the transform will also apply to any children of the node. For some transforms, this must be set; see the description of the individual transforms for details.
state	The active CTransformState when this node is encountered. Not all transforms require this information.

Returns

IDOMNodePtr The result. It may be a new node, a modified version, or NULL indicating that the node is to be dropped entirely.

transformPage()

```
virtual void JawsMako::ITransform::transformPage (
            const IPagePtr & page,
            bool transformContent = true,
            bool transformAnnotations = true )  [pure virtual]
```

Apply the transform to the given page, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time. The transform will also apply to the annotations appearances.

Parameters

page	The page to be processed.
transformContent	If true, apply the transform to the page content
transformAnnotations	If true, apply the transform to the annotations

The documentation for this class was generated from the following file:

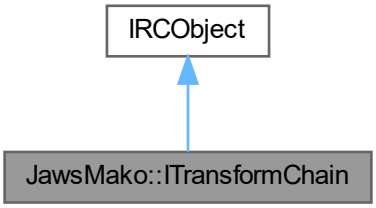
- transforms.h

7.431 JawsMako::ITransformChain Class Reference

ITransformChain represents a change of ITransforms, and provides a method of applying a range of transforms to an entire DOM tree. Instances of this type attempt to ensure that shared resources are modified once only.

```
#include <transforms.h>
```

Inheritance diagram for JawsMako::ITransformChain:



Public Member Functions

- virtual void removeTransform (uint32 index)=0

  Remove the ITransform at the specified index from the ITransformChain.
- virtual void pushTransform (const ITransformPtr &transform)=0

  Push an ITransform onto the end of the ITransformChain. Should not be called if another thread is currently using the transform.
- virtual void pushTransformFront (const ITransformPtr &transform)=0

  Push an ITransform onto the front of the ITransformChain. Should not be called if another thread is currently using the transform.
- virtual CTransformVect getTransforms () const =0

  Get the transforms in a vector.
- virtual IDOMNodePtr transform (const IDOMNodePtr &node)=0

  Apply the transform chain to the given node and its children, returning the resulting node. This is thread safe.
- virtual IDOMNodePtr transform (const IDOMNodePtr &node, bool &changed)=0

  Apply the transform chain to the given node and its children, returning the resulting node, and providing an indication if any changes were actually made.
- virtual void transformPage (const IPagePtr &page, bool transformContent=true, bool transform↩ Annotations=true)=0

  Apply the transform chain to the given page, if applicable. The transform will also apply to the annotations appearances.
- virtual void **flushCaches** ()=0

  Flush the caches used by the transforms in the chain. Most transforms cache recently transformed results to improve the performance of repeated transformations of equivalent results. However, it is possible that some cached results may point to entities that no longer exist, such as content inside an XPS file that no longer exists. If you are deleting or replacing files where transforms have been used, it is advisable to invoke this routine to clear the caches.
- virtual ITransformChainPtr **clone** ()=0

  Clones this ITransformChain into a new object that can be manipulated independently of the original.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API ITransformChainPtr create (const IJawsMakoPtr &jawsMako, const IProgress↩ MonitorPtr &progressMonitor=IProgressMonitorPtr(), const CTransformVect &transforms=CTransformVect())

  Create a new transform chain.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.431.1 Detailed Description

ITransformChain represents a change of ITransforms, and provides a method of applying a range of transforms to an entire DOM tree. Instances of this type attempt to ensure that shared resources are modified once only.

7.431.2 Member Function Documentation

create()

```
static JAWSMAKO_API ITransformChainPtr JawsMako::ITransformChain::create (
            const IJawsMakoPtr & jawsMako,
            const IProgressMonitorPtr & progressMonitor = IProgressMonitorPtr(),
            const CTransformVect & transforms = CTransformVect() )  [static]
```

Create a new transform chain.

Parameters

jawsMako	The JawsMako instance.
progressMonitor	The progress monitor which allows aborting an operation or registering a progress callback.
transforms	The initial vector of ITransforms for the chain.

Returns

ITransformChainPtr The new instance.

getTransforms()

```
virtual CTransformVect JawsMako::ITransformChain::getTransforms ( ) const  [pure virtual]
```

Get the transforms in a vector.

Returns

CTransformVect The transforms.

pushTransform()

```
virtual void JawsMako::ITransformChain::pushTransform (
            const ITransformPtr & transform )  [pure virtual]
```

Push an ITransform onto the end of the ITransformChain. Should not be called if another thread is currently using the transform.

Parameters

transform	The transform to push.

pushTransformFront()

```
virtual void JawsMako::ITransformChain::pushTransformFront (
            const ITransformPtr & transform )  [pure virtual]
```

Push an ITransform onto the front of the ITransformChain. Should not be called if another thread is currently using the transform.

Parameters

transform	The transform to push.

removeTransform()

```
virtual void JawsMako::ITransformChain::removeTransform (
            uint32 index )  [pure virtual]
```

Remove the ITransform at the specified index from the ITransformChain.

Parameters

index	The index of the ITransform to remove

transform() [1/2]

```
virtual IDOMNodePtr JawsMako::ITransformChain::transform (
            const IDOMNodePtr & node )  [pure virtual]
```

Apply the transform chain to the given node and its children, returning the resulting node. This is thread safe.

Parameters

node	The node to transform.

Returns

IDOMNodePtr the modified node.

transform() [2/2]

```
virtual IDOMNodePtr JawsMako::ITransformChain::transform (
            const IDOMNodePtr & node,
            bool & changed )  [pure virtual]
```

Apply the transform chain to the given node and its children, returning the resulting node, and providing an indication if any changes were actually made.

Parameters

node	The node to transform.
changed	On exit changes will be set to true if any modifications were made, false otherwise.

Returns

IDOMNodePtr the modified node.

transformPage()

```
virtual void JawsMako::ITransformChain::transformPage (
            const IPagePtr & page,
            bool transformContent = true,
            bool transformAnnotations = true )  [pure virtual]
```

Apply the transform chain to the given page, if applicable. The transform will also apply to the annotations appearances.

Parameters

page	The page to be processed.
transformContent	If true, apply the transform to the page content
transformAnnotations	If true, apply the transform to the annotations

The documentation for this class was generated from the following file:

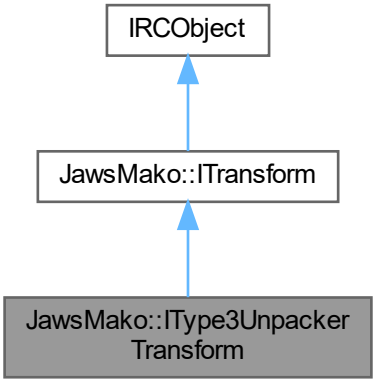
- transforms.h

7.432 JawsMako::IType3UnpackerTransform Class Reference

A transform for unpacking glyphs using a Type 3 font into regular DOM.

```
#include <transforms.h>
```

Inheritance diagram for JawsMako::IType3UnpackerTransform:



Static Public Member Functions

- static JAWSMAKO_API IType3UnpackerTransformPtr create (const IJawsMakoPtr &jawsMako, const IAbortPtr &abort=IAbortPtr())

  Create the transform.

Additional Inherited Members

Public Member Functions inherited from JawsMako::ITransform

- virtual IDOMBrushPtr transform (const IDOMBrushPtr &brush, eBrushUsage usage=eBUGeneral, const CTransformState &state=CTransformState())=0

  Apply the transform to the given brush, if applicable. These transforms are thread safe.
- virtual IDOMImagePtr transform (const IDOMImagePtr &image, const CTransformState &state=CTransformState())=0

  Apply the transform to the given image, if applicable. These transforms are thread safe.
- virtual IDOMColorPtr transform (const IDOMColorPtr &color, const CTransformState &state=CTransformState())=0

  Apply the transform to the given color, if applicable. These transforms are thread safe.
- virtual IDOMColorSpacePtr transform (const IDOMColorSpacePtr &colorSpace, const CTransformState &state=CTransformState())=0

  Apply the transform to the given colorspace, if applicable. These transforms are thread safe.
- virtual IDOMNodePtr transform (const IDOMNodePtr &node, bool &changed, bool transformChildren=true, const CTransformState &state=CTransformState())=0

  Apply the transform to the given node, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time.
- virtual void transformPage (const IPagePtr &page, bool transformContent=true, bool transform↩ Annotations=true)=0

  Apply the transform to the given page, if applicable. These transforms are thread safe, providing no other transforms are being applied to the same nodes at the same time. The transform will also apply to the annotations appearances.
- virtual void **flushCaches** ()=0

  Flush the caches used by the transform. Most transforms cache recently transformed results to improve the performance of repeated transformations of equivalent results. However, it is possible that some cached results may point to entities that no longer exist, such as content inside an XPS file that no longer exists. If you are deleting or replacing files where transforms have been used, it is advisable to invoke this routine to clear the caches.
- virtual void setProgressMonitor (const IProgressMonitorPtr &progressMonitor)=0

  Set the IProgressMonitor object for this transform to allow for monitoring the progress of the transform.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.432.1 Detailed Description

A transform for unpacking glyphs using a Type 3 font into regular DOM.

```
Glyphs are recolored if a plain solid color brush was used in the glyphs
fill. Otherwise, opacity masking is used to apply the glyphs brush to the
glyphs.

Useful for consumers that cannot directly handle Type 3 fonts.
```

7.432.2 Member Function Documentation

create()

```
static JAWSMAKO_API IType3UnpackerTransformPtr JawsMako::IType3UnpackerTransform::create (
            const IJawsMakoPtr & jawsMako,
            const IAbortPtr & abort = IAbortPtr() )  [static]
```

Create the transform.

Parameters

jawsMako	The JawsMako instance.
abort	An abort callback handler.

Returns

The new instance.

The documentation for this class was generated from the following file:

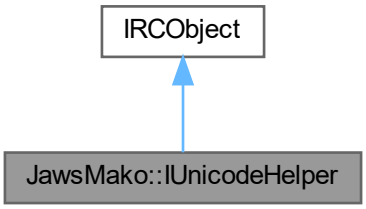
- transforms.h

7.433 JawsMako::IUnicodeHelper Class Reference

An interface into language specific unicode helpers.

`#include <text.h>`

Inheritance diagram for JawsMako::IUnicodeHelper:



Public Member Functions

- virtual bool **isCombiningCharacter** (const wchar_t wchar) const =0

  Returns true is wchar is within the languages combining character ranges.
- virtual wchar_t **getNonContextualCharacter** (const wchar_t ch) const =0

  Performs a reverse mapping on a contextual character and returns the original.
- virtual bool **hasContextualForms** () const =0

  Returns true if the language has contextual forms.
- virtual String **constructStringFromRuns** (CTextRunVect &runs) const =0

  Creates a string from a CTextRunVect by applying diacritic fixes and contextual substitution, if required. The supplied runs collection must be contain unicode strings of length one. Basically the glyphs need to be in their single character form.
- virtual void **applyDiacriticFixes** (CTextRunVect &runs) const =0

  Checks and attempts to fix diacritic characters that are incorrectly ordered.
- virtual void **applyContextualSubstitution** (String &input) const =0

  Applys the languages contextual substitution rules, in place.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API bool **isSpaceCharacter** (wchar_t ch)

  Returns true if ch is a unicode space character.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.433.1 Detailed Description

An interface into language specific unicode helpers.

The documentation for this class was generated from the following file:

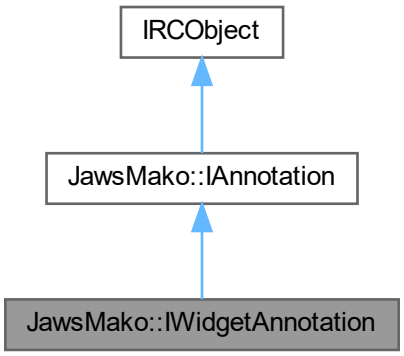
- text.h

7.434 JawsMako::IWidgetAnnotation Class Reference

An interface class for a widget annotation It is intended that future releases of JawsMako will extend this interface.

`#include <interactive.h>`

Inheritance diagram for JawsMako::IWidgetAnnotation:



Public Types

- enum eHighlightMode

  The highlight mode for a Widget annotation.
- enum eFontEncoding

  Encoding scheme used for form fonts.

Public Types inherited from JawsMako::IAnnotation

- enum eAnnotationType {
  eAT3D , eATCaret , eATCircle , eATFileAttachment ,
  eATFreeText , eATHighlight , eATInk , eATLine ,
  eATLink , eATMovie , eATPolygon , eATPolyLine ,
  eATPopup , eATPrinterMark , eATProjection , eATRedact ,
  eATRichMedia , eATScreen , eATSound , eATSquare ,
  eATSquiggly , eATStamp , eATStrikeOut , eATText ,
  eATTrapNet , eATUnderline , eATWatermark , eATWidget ,
  eATOther }

  Types of annotations, listed with the earliest version of PDF that supported them.

Public Member Functions

- virtual void createBasicAppearances (const IFormPtr &form, const U8String &onStateName=U8String(), const IDOMFontPtr &font=IDOMFontPtr(), uint32 fontIndex=0, float captionSize=10.0f)=0

  Create a set of very basic appearances given the current annotation's parameters. Replaces any already present. Not yet supported for all widget types. An IError with error code JM_ERR_UNSUPPORTED will be thrown for such appearances.
- virtual eFieldType getFieldType () const =0

  Get the field type.
- virtual U8String getPartialName () const =0

  Get the partial name of the widget.
- virtual void setPartialName (const U8String &name)=0

  Set the partial name of the widget.
- virtual eHighlightMode getHighlightMode () const =0

  Get the highlight mode.
- virtual void setHighlightMode (eHighlightMode highlightMode)=0

  Set the hightlight mode.
- virtual bool getFieldFlags (uint32 &flags) const =0

  Get the field flags. Please see the PDF specification for the definition of the flags. See also eFieldFlags. Note that the significance of of the flags changes depending on type, and that the value may be inheritable.
- virtual void setFieldFlags (uint32 flags)=0

  Set the field flags. Refer to the PDF specification 1.7 table 8.70 for details.
- virtual IWidgetAppearanceCharacteristicsPtr getAppearanceCharacteristics () const =0

  Get the appearance characteristics of this Widget.
- virtual void setAppearanceCharacteristics (const IWidgetAppearanceCharacteristicsPtr &appearance↩ Characteristics)=0

  Set the appearance characteristics.
- virtual bool getValue (CU8StringVect &value) const =0

  Get the value of the field, as an array of strings.
- virtual void setValue (const CU8StringVect &value)=0

  Set the value of the field, as an array of strings. For fields that can take multiple values, eg radio buttons, there will be multiple entries. For fields requiring only a single value, a single entry will be returned. If the field has no value an empty array will be returned.
- virtual void setValue (const U8String &value)

  Convenience form for fields that only need a single value.
- virtual bool getDefaultValue (CU8StringVect &value) const =0

  Get the default value of the field, as an array of strings.
- virtual void setDefaultValue (const CU8StringVect &value)=0

  Set the value of the field, as an array of strings.
- virtual void setDefaultValue (const U8String &value)

Convenience form for fields that only need a single value.

- virtual bool getDefaultAppearanceString (U8String &defaultAppearanceString) const =0

  Get the default appearance string for variable text, if set.
- virtual void setDefaultAppearanceString (const U8String &defaultAppearanceString)=0

  Set the default appearance string for variable text.
- virtual eQuadding getQuadding () const =0

  Get the Quadding (Justification) for variable text.
- virtual void setQuadding (eQuadding quadding)=0

  Set the quadding (Justification) for variable text.
- virtual bool getOptions (CFieldOptionVect &options) const =0

  Get the options for this field.
- virtual void setOptions (const CFieldOptionVect &options)=0

  Set the options for this field.
- virtual uint32 getFirstVisibleOption () const =0

  Get the index of the first item visible in the list from the options.
- virtual void setFirstVisibleOption (uint32 option)=0

  Set the index of the first item visible in the list from the options.
- virtual CUInt32Vect getSelectedOptions () const =0

  Get a vector of 0-indexed indexes into the options for this field that represent the currently selected options.
- virtual void setSelectedOptions (const CUInt32Vect &selectedOptions)=0

  Set the currently selected options by 0-indexed indexes into the options.
- virtual IPDFDictionaryConstPtr getActionsDictionary () const =0

  Get the actions dictionary (if present) associated with this widget.
- virtual void setActionsDictionary (const IPDFDictionaryPtr &actions)=0

  Set the actions dictionary associated with this widget.
- virtual IPDFDictionaryConstPtr getAdditionalActionsDictionary () const =0

  Get the additional actions dictionary (if present) associated with this widget.
- virtual void setAdditionalActionsDictionary (const IPDFDictionaryPtr &actions)=0

  Set the additional actions dictionary associated with this widget.

Public Member Functions inherited from JawsMako::IAnnotation

- virtual eAnnotationType getType () const =0

  Get the type of the annotation.
- virtual const FRect & getRect () const =0

  Get the rect in which the appearances should be displayed.
- virtual void setRect (const FRect &rect)=0

  Set the rect in which the appearances should be displayed.
- virtual U8String getContents () const =0

  Get the Contents entry in UTF-8.
- virtual void setContents (const U8String &contents)=0

  Set the Contents entry in UTF-8.
- virtual IDOMColorPtr getColor () const =0

  Get the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.
- virtual void setColor (const IDOMColorPtr &color)=0

  Set the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.
- virtual IEDLTimePtr getModificationTime () const =0

  Get the Modification date and time of the annotation, if present.
- virtual void setModificationTime (const IEDLTimePtr &modificationTime)=0

  Set the Modification date and time of the annotation.

- virtual CAnnotationBorder getBorder () const =0

  Get the annotation's border. See CAnnotationBorder for details.
- virtual void setBorder (const CAnnotationBorder &border)=0

  Set the annotation's border.
- virtual uint32 getFlags () const =0

  Get the annotation flags. To interpret these flags please see section 8.4.2 "Annotation Flags" in the PDF 1.7 specification.
- virtual void setFlags (uint32 flags)=0

  Set the annotation flags. Please see section 8.4.2 "Annotation Flags" for details about these flags.
- virtual void rotate (uint16 rotate, double pageWidth, double pageHeight)=0

  Rotate the annotation clockwise as if the page was rotated by the same amount.
- virtual CAnnotationAppearanceVect getAppearances () const =0

  Return all the annotation appearances in a vector.
- virtual void **removeAppearances** ()=0

  Remove all annotation appearances.
- virtual U8String getState () const =0

  Get the current annotation state.
- virtual void setState (const U8String &state)=0

  Set the current annotation state.
- virtual IAnnotationAppearancePtr getAppearance (eAppearanceUsage usage, const U8String &state=U8String()) const =0

  Fetch the annotation appearance that should be used for the given annotation usage and state according to the following rules:
- virtual void addAppearance (const IAnnotationAppearancePtr &appearance)=0

  Add or replace an appearance.
- virtual bool hasNormalAppearance () const =0

  Does the annotation have a normal appearance? Convenience utility function.
- virtual IAnnotationPtr clone () const =0

  Clone the annotation. This is a deep clone. The annotation reference will remain the same.
- virtual bool matchesReference (const IAnnotationReferencePtr &reference) const =0

  Does this annotation match the given IAnnotationReference?
- virtual IAnnotationReferencePtr getReference () const =0

  Get a reference that can be used to refer to this annotation.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IWidgetAnnotationPtr createTextField (const IJawsMakoPtr &jawsMako, const IFormPtr &form, const IPagePtr &page, const FRect &rect, const U8String &partialName, const U8String &text, float textSize, const IDOMFontOpenTypePtr &font, uint32 fontIndex=0, eFontEncoding encoding↩Hint=eFELatin, const IDOMColorPtr &color=IDOMColorPtr(), bool multiLine=false, eQuadding quadding=e↩QLeftJustified)

  Creator for a simple text annotation, including a basic appearance stream, inserts it into the form, and places it on the given page.
- static JAWSMAKO_API IWidgetAnnotationPtr createCheckButton (const IJawsMakoPtr &jawsMako, const IFormPtr &form, const IPagePtr &page, const FRect &rect, const U8String &partialName, const U8String &onValue, bool on=false)

  Creator for a check button annotation, including basic appearance streams, inserts it into the form, and places it on the given page.
- static JAWSMAKO_API IFormFieldPtr createRadioButtons (const IJawsMakoPtr &jawsMako, const IFormPtr &form, const IPagePtr &page, const U8String &partialName, const U8String &value, const CRadioButton↩InfoVect &buttonDetails)

  Creator for a set of radio buttons, including basic appearance streams, inserts them into the form inside a new subfield, and places them on the given page.
- static JAWSMAKO_API IWidgetAnnotationPtr createChoiceField (const IJawsMakoPtr &jawsMako, const IFormPtr &form, const IPagePtr &page, const FRect &rect, const U8String &partialName, bool combo, bool editable, bool multiSelect, const CFieldOptionVect &options, const CU8StringVect &selectedOptions, float textSize, const IDOMFontOpenTypePtr &font, uint32 fontIndex=0, eFontEncoding encodingHint=eFELatin, const IDOMColorPtr &color=IDOMColorPtr())

  Creator for a choice annotation, including basic appearance streams, inserts it into the form, and places it on the given page. Can create either menus, combo boxes, or selection boxes.
- static JAWSMAKO_API IWidgetAnnotationPtr createButton (const IJawsMakoPtr &jawsMako, const IFormPtr &form, const IPagePtr &page, const FRect &rect, const U8String &partialName, const U8String &caption, float textSize, const IDOMFontOpenTypePtr &font, uint32 fontIndex=0, eFontEncoding encodingHint=eFELatin, const IDOMColorPtr &color=IDOMColorPtr(), const IDOMColorPtr &background↩Color=IDOMColorPtr(), const IPDFDictionaryPtr &actions=IPDFDictionaryPtr())

  Convenience creator for a push button annotation, including basic appearance streams, inserts it into the form, and places it on the given page. Makes use of the above APIs to do its work. Can create either menus or selection boxes.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.434.1 Detailed Description

An interface class for a widget annotation It is intended that future releases of JawsMako will extend this interface.

7.434.2 Member Function Documentation

createBasicAppearances()

```
virtual void JawsMako::IWidgetAnnotation::createBasicAppearances (
            const IFormPtr & form,
            const U8String & onStateName = U8String(),
            const IDOMFontPtr & font = IDOMFontPtr(),
            uint32 fontIndex = 0,
            float captionSize = 10.0f )  [pure virtual]
```

Create a set of very basic appearances given the current annotation's parameters. Replaces any already present. Not yet supported for all widget types. An IError with error code JM_ERR_UNSUPPORTED will be thrown for such appearances.

Supported for:

- Variable text
- Choice fields
- Check boxes
- Radio Buttons
- Push Buttons (no icons)

Parameters

form	The form in which this widget appears. Necessary if field information is inherited from its parent in the tree. If form is NULL and the widget does not know its own details, an exception will result. If in doubt, always pass the form.
onStateName	The name of the appearance state to use for the "on" appearance. Required for button fields. Ignored for other field types. The name "Off" is used for all off appearance states.
font	The font to use when we need to create a caption or variable text appearance. If not provided and it is needed, an exception will result.
fontIndex	The index of the font to use.
captionSize	The size of the font to use when we need to create a caption.

createButton()

```
static JAWSMAKO_API IWidgetAnnotationPtr JawsMako::IWidgetAnnotation::createButton (
            const IJawsMakoPtr & jawsMako,
            const IFormPtr & form,
            const IPagePtr & page,
            const FRect & rect,
            const U8String & partialName,
            const U8String & caption,
            float textSize,
            const IDOMFontOpenTypePtr & font,
            uint32 fontIndex = 0,
            eFontEncoding encodingHint = eFELatin,
            const IDOMColorPtr & color = IDOMColorPtr(),
```

```
            const IDOMColorPtr & backgroundColor = IDOMColorPtr(),
            const IPDFDictionaryPtr & actions = IPDFDictionaryPtr() )  [static]
```

Convenience creator for a push button annotation, including basic appearance streams, inserts it into the form, and places it on the given page. Makes use of the above APIs to do its work. Can create either menus or selection boxes.

Parameters

jawsMako	The Mako instance
form	The form to receive the button field
page	The page to attach the annotation to
rect	The annotation area
partialName	The desired partial name of the field. If an empty string is passed, no partial name will be set.
caption	The string to use as the caption
textSize	The text size for the caption
font	The OpenType font to use for the field text
fontIndex	If font is a TrueType collection, the index of the font within the collection
encodingHint	A hint of the encoding to use for the font, if an encoding cannot be unambigously detected from the font itself. Not usually critical for buttons. Must be consistent with other form users of this font.
color	The color to use for the caption. Must use a DeviceGray, DeviceRGB or DeviceCMYK colorspace. Default is black.
backgroundColor	The background color for the button
actions	The action(s) to be performed when the button is clicked

Returns

IWidgetAnnotationPtr The widget. May be edited after creation for further customisation.

createCheckButton()

```
static JAWSMAKO_API IWidgetAnnotationPtr JawsMako::IWidgetAnnotation::createCheckButton (
            const IJawsMakoPtr & jawsMako,
            const IFormPtr & form,
            const IPagePtr & page,
            const FRect & rect,
            const U8String & partialName,
            const U8String & onValue,
            bool on = false )  [static]
```

Creator for a check button annotation, including basic appearance streams, inserts it into the form, and places it on the given page.

Parameters

jawsMako	The Mako instance
form	The form to receive the button field
page	The page to attach the annotation to
rect	The annotation area
partialName	The desired partial name of the field. If an empty string is passed, no partial name will be set.
onValue	The name used by the form to specify an on value. Must be supplied.
on	If true, the button will be checked by default

Returns

IWidgetAnnotationPtr The widget. May be edited after creation for further customisation.

createChoiceField()

```
static JAWSMAKO_API IWidgetAnnotationPtr JawsMako::IWidgetAnnotation::createChoiceField (
            const IJawsMakoPtr & jawsMako,
            const IFormPtr & form,
            const IPagePtr & page,
            const FRect & rect,
            const U8String & partialName,
            bool combo,
            bool editable,
            bool multiSelect,
            const CFieldOptionVect & options,
            const CU8StringVect & selectedOptions,
            float textSize,
            const IDOMFontOpenTypePtr & font,
            uint32 fontIndex = 0,
            eFontEncoding encodingHint = eFELatin,
            const IDOMColorPtr & color = IDOMColorPtr() )  [static]
```

Creator for a choice annotation, including basic appearance streams, inserts it into the form, and places it on the given page. Can create either menus, combo boxes, or selection boxes.

Parameters

jawsMako	The Mako instance
form	The form to receive the button field
page	The page to attach the annotation to
rect	The annotation area
partialName	The desired partial name of the field. If an empty string is passed, no partial name will be set.
combo	If true, a combo box/popup will be created. If false, a selectable field will be created.
editable	If true, and combo is set, the combo box text will be editable
multiSelect	If true, and combo is not set, the choice field will allow multiple selections
options	The options present in the choice field
selectedOptions	The options that are initially selected
textSize	The text size
font	The OpenType font to use for the field text
fontIndex	If font is a TrueType collection, the index of the font within the collection
encodingHint	A hint of the encoding to use for the font, if an encoding cannot be unambigously detected from the font itself. Must be consistent with other form users of this font.
color	The color to use for the text. Must use a DeviceGray, DeviceRGB or DeviceCMYK colorspace. Default is black.

Returns

IWidgetAnnotationPtr The widget. May be edited after creation for further customisation.

createRadioButtons()

```
static JAWSMAKO_API IFormFieldPtr JawsMako::IWidgetAnnotation::createRadioButtons (
            const IJawsMakoPtr & jawsMako,
            const IFormPtr & form,
            const IPagePtr & page,
            const U8String & partialName,
            const U8String & value,
            const CRadioButtonInfoVect & buttonDetails )  [static]
```

Creator for a set of radio buttons, including basic appearance streams, inserts them into the form inside a new subfield, and places them on the given page.

Parameters

jawsMako	The Mako instance
form	The form to receive the fields
page	The page to attach the annotation to
partialName	The desired partial name of the field that represents the group of radio buttons. If an empty string is passed, no partial name will be set.
value	The value of the field; that is, the name of the radio button that is checked. Pass an empty string if none are selected.
buttonDetails	A vector containing the details of the radio buttons

Returns

IFormFieldPtr The field that contains the widgets

createTextField()

```
static JAWSMAKO_API IWidgetAnnotationPtr JawsMako::IWidgetAnnotation::createTextField (
            const IJawsMakoPtr & jawsMako,
            const IFormPtr & form,
            const IPagePtr & page,
            const FRect & rect,
            const U8String & partialName,
            const U8String & text,
            float textSize,
            const IDOMFontOpenTypePtr & font,
            uint32 fontIndex = 0,
            eFontEncoding encodingHint = eFELatin,
            const IDOMColorPtr & color = IDOMColorPtr(),
            bool multiLine = false,
            eQuadding quadding = eQLeftJustified )  [static]
```

Creator for a simple text annotation, including a basic appearance stream, inserts it into the form, and places it on the given page.

Parameters

jawsMako	The Mako instance
form	The form to receive the text field
page	The page to attach the annotation to

Parameters

rect	The annotation area
partialName	The desired partial name of the field. If an empty string is passed, no partial name will be set.
text	The text present in the field (the field's value)
textSize	The text size
font	The OpenType font to use for the field text
fontIndex	If font is a TrueType collection, the index of the font within the collection
encodingHint	A hint of the encoding to use for the font, if an encoding cannot be unambigously detected from the font itself. Must be consistent with other form users of this font.
color	The color to use for the text. Must use a DeviceGray, DeviceRGB or DeviceCMYK colorspace
multiLine	True if the text should wrap within the field, false otherwise
quadding	The desired justification mode for the text within the field

Returns

IWidgetAnnotationPtr The widget. May be edited after creation for further customisation.

getActionsDictionary()

```
virtual IPDFDictionaryConstPtr JawsMako::IWidgetAnnotation::getActionsDictionary ( ) const
[pure virtual]
```

Get the actions dictionary (if present) associated with this widget.

Returns

IPDFDictionaryConstPtr the actions dictionary, or NULL if not present.

getAdditionalActionsDictionary()

```
virtual IPDFDictionaryConstPtr JawsMako::IWidgetAnnotation::getAdditionalActionsDictionary ( )
const  [pure virtual]
```

Get the additional actions dictionary (if present) associated with this widget.

Returns

IPDFDictionaryConstPtr the actions dictionary, or NULL if not present.

getAppearanceCharacteristics()

```
virtual IWidgetAppearanceCharacteristicsPtr JawsMako::IWidgetAnnotation::getAppearanceCharacteristics
( ) const  [pure virtual]
```

Get the appearance characteristics of this Widget.

Returns

IAppearanceCharacteristicsPtr The appearance characteristics, or NULL if appearance characteristics are not present. Note that if you edit the returned object, the edits will not take effect in the widget until setAppearanceCharacteristics() is called.

getDefaultAppearanceString()

```
virtual bool JawsMako::IWidgetAnnotation::getDefaultAppearanceString (
            U8String & defaultAppearanceString ) const  [pure virtual]
```

Get the default appearance string for variable text, if set.

Parameters

defaultAppearanceString	A reference to receive the default appearance string

Returns

bool true if the field defines a default appearance. If false, the value may be inherited from a parent field

getDefaultValue()

```
virtual bool JawsMako::IWidgetAnnotation::getDefaultValue (
            CU8StringVect & value ) const  [pure virtual]
```

Get the default value of the field, as an array of strings.

Parameters

value	A vector to receive the value. For fields that can take multiple values, such as radio buttons or choices, there may be multiple entries. For fields requiring only a single entry.

Returns

bool True if the field has a value. If false, the value may be stored in a parent field

getFieldFlags()

```
virtual bool JawsMako::IWidgetAnnotation::getFieldFlags (
            uint32 & flags ) const  [pure virtual]
```

Get the field flags. Please see the PDF specification for the definition of the flags. See also eFieldFlags. Note that the significance of of the flags changes depending on type, and that the value may be inheritable.

Parameters

flags	Reference to receive the field flags

Returns

bool False if the flags have not been set for this widget, otherwise true

getFieldType()

```
virtual eFieldType JawsMako::IWidgetAnnotation::getFieldType ( ) const  [pure virtual]
```

Get the field type.

Returns

eFieldType The field type

getFirstVisibleOption()

```
virtual uint32 JawsMako::IWidgetAnnotation::getFirstVisibleOption ( ) const  [pure virtual]
```

Get the index of the first item visible in the list from the options.

Returns

uint32 The index of the first option visible in a list

getHighlightMode()

```
virtual eHighlightMode JawsMako::IWidgetAnnotation::getHighlightMode ( ) const  [pure virtual]
```

Get the highlight mode.

Returns

eHightlightMode The highlight mode

getOptions()

```
virtual bool JawsMako::IWidgetAnnotation::getOptions (
            CFieldOptionVect & options ) const  [pure virtual]
```

Get the options for this field.

Parameters

options	Reference to receive the options

Returns

bool true if the option is present. If false, the options may be inherited and the parent field should be consulted.

getPartialName()

```
virtual U8String JawsMako::IWidgetAnnotation::getPartialName ( ) const  [pure virtual]
```

Get the partial name of the widget.

Returns

U8String The partial name. If no partial name is present, an empty string is returned

getQuadding()

```
virtual eQuadding JawsMako::IWidgetAnnotation::getQuadding ( ) const  [pure virtual]
```

Get the Quadding (Justification) for variable text.

Returns

eQuadding The quadding to use, or eQInherited if it is inherited from a parent field

getSelectedOptions()

```
virtual CUInt32Vect JawsMako::IWidgetAnnotation::getSelectedOptions ( ) const  [pure virtual]
```

Get a vector of 0-indexed indexes into the options for this field that represent the currently selected options.

Returns

CUint32Vect The selected option indexes

getValue()

```
virtual bool JawsMako::IWidgetAnnotation::getValue (
            CU8StringVect & value ) const  [pure virtual]
```

Get the value of the field, as an array of strings.

Parameters

value	A vector to receive the value. For fields that can take multiple values, such as radio buttons or choices, there may be multiple entries. For fields requiring only a single entry.

Returns

bool True if the field has a value. If false, the value may be stored in a parent field

setActionsDictionary()

```
virtual void JawsMako::IWidgetAnnotation::setActionsDictionary (
            const IPDFDictionaryPtr & actions )  [pure virtual]
```

Set the actions dictionary associated with this widget.

Parameters

actions	Either an actions dictionary or NULL to clear

setAdditionalActionsDictionary()

```
virtual void JawsMako::IWidgetAnnotation::setAdditionalActionsDictionary (
            const IPDFDictionaryPtr & actions )  [pure virtual]
```

Set the additional actions dictionary associated with this widget.

Parameters

actions	Either an actions dictionary or NULL to clear

setAppearanceCharacteristics()

```
virtual void JawsMako::IWidgetAnnotation::setAppearanceCharacteristics (
            const IWidgetAppearanceCharacteristicsPtr & appearanceCharacteristics )  [pure
virtual]
```

Set the appearance characteristics.

Parameters

appearanceCharacteristics	The desired appearance characteristics, or NULL to clear.

setDefaultAppearanceString()

```
virtual void JawsMako::IWidgetAnnotation::setDefaultAppearanceString (
            const U8String & defaultAppearanceString )  [pure virtual]
```

Set the default appearance string for variable text.

Parameters

defaultAppearanceString	The desired default appearance string

setDefaultValue() [1/2]

```
virtual void JawsMako::IWidgetAnnotation::setDefaultValue (
            const CU8StringVect & value )  [pure virtual]
```

Set the value of the field, as an array of strings.

For fields that can take multiple values, eg radio buttons, there will be multiple entries. For fields requiring only a single value, a single entry must be specified. Pass an empty vector to clear the value.

Parameters

value	The value.

setDefaultValue() [2/2]

```
virtual void JawsMako::IWidgetAnnotation::setDefaultValue (
            const U8String & value )  [inline], [virtual]
```

Convenience form for fields that only need a single value.

Parameters

value	The string value for the field.

setFieldFlags()

```
virtual void JawsMako::IWidgetAnnotation::setFieldFlags (
            uint32 flags )  [pure virtual]
```

Set the field flags. Refer to the PDF specification 1.7 table 8.70 for details.

Parameters

flags	The desired flags.

setFirstVisibleOption()

```
virtual void JawsMako::IWidgetAnnotation::setFirstVisibleOption (
            uint32 option )  [pure virtual]
```

Set the index of the first item visible in the list from the options.

Parameters

option	The desired index of the first option visible in a list

setHighlightMode()

```
virtual void JawsMako::IWidgetAnnotation::setHighlightMode (
            eHighlightMode highlightMode )  [pure virtual]
```

Set the hightlight mode.

Parameters

highlightMode	The highlight mode to set.

setOptions()

```
virtual void JawsMako::IWidgetAnnotation::setOptions (
            const CFieldOptionVect & options )  [pure virtual]
```

Set the options for this field.

Parameters

options The desired options

setPartialName()

```
virtual void JawsMako::IWidgetAnnotation::setPartialName (
            const U8String & name )  [pure virtual]
```

Set the partial name of the widget.

Parameters

name	The partial name to be set

setQuadding()

```
virtual void JawsMako::IWidgetAnnotation::setQuadding (
            eQuadding quadding )  [pure virtual]
```

Set the quadding (Justification) for variable text.

Parameters

quadding	The desired quadding

setSelectedOptions()

```
virtual void JawsMako::IWidgetAnnotation::setSelectedOptions (
            const CUInt32Vect & selectedOptions )  [pure virtual]
```

Set the currently selected options by 0-indexed indexes into the options.

Parameters

selectedOptions	The selected option indexes

setValue() [1/2]

```
virtual void JawsMako::IWidgetAnnotation::setValue (
            const CU8StringVect & value )  [pure virtual]
```

Set the value of the field, as an array of strings. For fields that can take multiple values, eg radio buttons, there will be multiple entries. For fields requiring only a single value, a single entry will be returned. If the field has no value an empty array will be returned.

Parameters

value	The value.

setValue() [2/2]

```
virtual void JawsMako::IWidgetAnnotation::setValue (
            const U8String & value )  [inline], [virtual]
```

Convenience form for fields that only need a single value.

Parameters

value	The string value for the field.

The documentation for this class was generated from the following file:

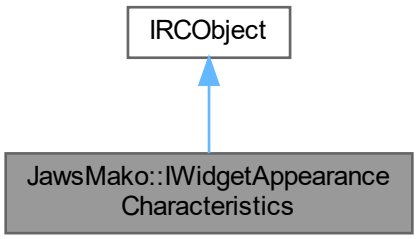
- interactive.h

7.435 JawsMako::IWidgetAppearanceCharacteristics Class Reference

Appearance information for a widget annotation.

```
#include <interactive.h>
```

Inheritance diagram for JawsMako::IWidgetAppearanceCharacteristics:



Public Member Functions

- virtual int32 getRotation () const =0

  Get the rotation of the widget appearance.
- virtual void setRotation (int32 rotation)=0

  Set the rotation of the widget appearance.
- virtual IDOMColorPtr getBorderColor () const =0

  Get the border color of the widget.
- virtual void setBorderColor (const IDOMColorPtr &color)=0

  Set the border color of the widget.
- virtual IDOMColorPtr getBackgroundColor () const =0

  Get the background color of the widget.
- virtual void setBackgroundColor (const IDOMColorPtr &color)=0

  Set the background color of the widget.
- virtual U8String getCaption () const =0

  Get the caption for the widget.
- virtual void setCaption (const U8String &caption)=0

  Set the caption for the widget.
- virtual U8String getRolloverCaption () const =0

  Get the rollover caption for the widget.
- virtual void setRolloverCaption (const U8String &caption)=0

  Set the rollover caption for the widget.
- virtual U8String getAlternateCaption () const =0

  Get the alternate caption for the widget.
- virtual void setAlternateCaption (const U8String &caption)=0

  Set the alternate caption for the widget.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IWidgetAppearanceCharacteristicsPtr create (const IJawsMakoPtr &jawsMako, int32 rotation=0, const IDOMColorPtr &borderColor=IDOMColorPtr(), const IDOMColorPtr &backgroundColor=IDOMColorPtr(), const U8String &caption=U8String(), const U8String &rolloverCaption=U8String(), const U8String &alternateCaption=U8String())

  Create a new widget appearance characteristics.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.435.1 Detailed Description

Appearance information for a widget annotation.

7.435.2 Member Function Documentation

create()

```
static JAWSMAKO_API IWidgetAppearanceCharacteristicsPtr JawsMako::IWidgetAppearanceCharacteristics↩
::create (
            const IJawsMakoPtr & jawsMako,
            int32 rotation = 0,
            const IDOMColorPtr & borderColor = IDOMColorPtr(),
            const IDOMColorPtr & backgroundColor = IDOMColorPtr(),
            const U8String & caption = U8String(),
            const U8String & rolloverCaption = U8String(),
            const U8String & alternateCaption = U8String() )  [static]
```

Create a new widget appearance characteristics.

Parameters

jawsMako	The Mako instance
rotation	The rotation of the widget in degrees. Must be a multiple of 90.
borderColor	The color to use for the border. Must use a DeviceGray, DeviceRGB or DeviceCMYK colorspace.
backgroundColor	The color to use for the background. Must use a DeviceGray, DeviceRGB or DeviceCMYK colorspace.
caption	The widget's normal caption.
rolloverCaption	The widget's rollover caption. Used only for push button widgets.
alternateCaption	The widgets's alternate caption. Used only for push button widgets.

getAlternateCaption()

```
virtual U8String JawsMako::IWidgetAppearanceCharacteristics::getAlternateCaption ( ) const
[pure virtual]
```

Get the alternate caption for the widget.

Returns

U8String The caption, or an empty string if not present.

getBackgroundColor()

```
virtual IDOMColorPtr JawsMako::IWidgetAppearanceCharacteristics::getBackgroundColor ( ) const
[pure virtual]
```

Get the background color of the widget.

Returns

IDOMColorPtr The color, or NULL if not present

getBorderColor()

```
virtual IDOMColorPtr JawsMako::IWidgetAppearanceCharacteristics::getBorderColor ( ) const
[pure virtual]
```

Get the border color of the widget.

Returns

IDOMColorPtr The color, or NULL if not present

getCaption()

```
virtual U8String JawsMako::IWidgetAppearanceCharacteristics::getCaption ( ) const  [pure virtual]
```

Get the caption for the widget.

Returns

U8String The caption, or an empty string if not present.

getRolloverCaption()

```
virtual U8String JawsMako::IWidgetAppearanceCharacteristics::getRolloverCaption ( ) const
[pure virtual]
```

Get the rollover caption for the widget.

Returns

U8String The caption, or an empty string if not present.

getRotation()

```
virtual int32 JawsMako::IWidgetAppearanceCharacteristics::getRotation ( ) const  [pure virtual]
```

Get the rotation of the widget appearance.

Returns

int32 The rotation, in degrees

setAlternateCaption()

```
virtual void JawsMako::IWidgetAppearanceCharacteristics::setAlternateCaption (
            const U8String & caption )  [pure virtual]
```

Set the alternate caption for the widget.

Parameters

caption	The caption, or an empty string to clear

setBackgroundColor()

```
virtual void JawsMako::IWidgetAppearanceCharacteristics::setBackgroundColor (
            const IDOMColorPtr & color )  [pure virtual]
```

Set the background color of the widget.

Parameters

color	The color, or NULL to clear. Must use either DeviceGray, DeviceRGB, or DeviceCMYK.

setBorderColor()

```
virtual void JawsMako::IWidgetAppearanceCharacteristics::setBorderColor (
```

```
            const IDOMColorPtr & color )  [pure virtual]
```

Set the border color of the widget.

Parameters

color	The color, or NULL to clear. Must use either DeviceGray, DeviceRGB, or DeviceCMYK.

setCaption()

```
virtual void JawsMako::IWidgetAppearanceCharacteristics::setCaption (
            const U8String & caption )  [pure virtual]
```

Set the caption for the widget.

Parameters

caption	The caption, or an empty string to clear

setRolloverCaption()

```
virtual void JawsMako::IWidgetAppearanceCharacteristics::setRolloverCaption (
            const U8String & caption )  [pure virtual]
```

Set the rollover caption for the widget.

Parameters

caption	The caption, or an empty string to clear

setRotation()

```
virtual void JawsMako::IWidgetAppearanceCharacteristics::setRotation (
            int32 rotation )  [pure virtual]
```

Set the rotation of the widget appearance.

Parameters

rotation	The rotation in degrees. Must be a multiple of 90 degrees.

The documentation for this class was generated from the following file:

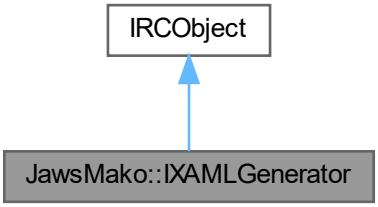
- interactive.h

7.436 JawsMako::IXAMLGenerator Class Reference

A XAML generator for Mako, allowing simple generation of XAML fragments for individual DOM nodes or entire pages.

`#include <xamlgenerator.h>`

Inheritance diagram for JawsMako::IXAMLGenerator:



Classes

- class CAnnotationXAML

  Class for receiving XAML generated for annotation appearances in a bulk fashion.
- class CResourceEntry

  Resource entry.

Public Member Functions

- virtual IRAInputStreamPtr generateXAML (const IDOMNodePtr &node, const IDOMCatalogPtr &catalog=IDOMCatalog↩ Ptr())=0

  Generate XAML for the given DOM Node, returning the result in a stream.
- virtual void generateXAML (const IDOMNodePtr &node, const IOutputStreamPtr &outputStream, const IDOMCatalogPtr &catalog=IDOMCatalogPtr())=0

  Alternate form of generateXAML() when an existing stream should be used.
- virtual IRAInputStreamPtr generateXAMLForPageAndAnnotationAppearances (const IPagePtr &page, CAnnotationXAMLVect &appearanceXAMLs)=0

  Generate XAML for the given IPage and all the annotation appearances present on the page. The XAML for the page will be in the returned stream, and appearanceXAMLs will be populated with the XAML for all annotation appearances. This will usually be faster than using generateXAML() on the page contents and then separately invoking generateXAMLForPageAnnotationAppearances().
- virtual IRAInputStreamPtr generateXAMLForAppearance (const IAnnotationAppearancePtr &appearance, const IAnnotationPtr &annotation, const IPagePtr &page)=0

  Generate XAML for a given annotation appearance. Provide the annotation and page from which the appearance comes. The page is required as some annotations may need to be composited against the page backdrop. This is slower than invoking generateXAMLForPageAnnotationAppearances() which is in turn slower than using generate↩ XAMLForPageAndAnnotations().
- virtual void generateXAMLForAppearance (const IAnnotationAppearancePtr &appearance, const IAnnotationPtr &annotation, const IPagePtr &page, const IOutputStreamPtr &outputStream)=0

Alternate form of generateXAMLForAppearance() where an existing stream should be used. Provide the annotation and page from which the appearance comes. The page is required as some annotations may need to be composited against the page backdrop. This is slower than invoking generateXAMLForPageAnnotationAppearances() which is in turn slower than using generateXAMLForPageAndAnnotations().

- virtual void generateXAMLForPageAnnotationAppearances (const IPagePtr &page, CAnnotationXAMLVect &appearanceXAMLs)=0

  Generate XAML for all the annotation appearances on the given page. This is generally more efficient than using generateXAMLForAppearance() repeatedly for every page, but not as fast as using generateXAMLForPageAnd↩ Annotations() to generate XAML for the page and the annotations in one bulk operation.

- virtual IInputStreamPtr getResource (const U8String &name)=0

  Get the stream for a named resource. An exception will be thrown if the resource cannot be found.

- virtual void getResources (CEDLVector< CResourceEntry > &resources)=0

  Get all the resources in a vector.

- virtual void setSubsetFonts (bool subset)=0

  Set whether fonts should be subset in the output.

- virtual void setMergeFonts (bool merge)=0

  Set whether or not an attempt will be made to merge disparate subsets of a font into a single font.

- virtual void **setMergeImages** (bool merge=true)=0

  Set if the XAML writer should attempt to merge adjacent images. The default is true.

- virtual void setColorImageMaxResolution (float resolution, float threshold=0.0f, IDOMImageDownsamplerFilter::eDownsampling method=IDOMImageDownsamplerFilter::eBicubic)=0

  Set the desired maximum resolution, threshold and downsampling method for colour images for XAML output.

- virtual void setGrayImageMaxResolution (float resolution, float threshold=0.0f, IDOMImageDownsamplerFilter::eDownsampling method=IDOMImageDownsamplerFilter::eBicubic)=0

  Set the desired maximum resolution, threshold and downsampling method for gray images.

- virtual void setMonoImageMaxResolution (float resolution, float threshold=0.0f, IDOMImageDownsamplerFilter::eDownsampling method=IDOMImageDownsamplerFilter::eSubsample)=0

  Set the desired maximum resolution, threshold and downsampling method for monochrome images.

- virtual void setTargetColorSpace (const IDOMColorSpacePtr &targetSpace)=0

  Set the target color space for the output. The default behaviour is to, where possible, leave the color space of objects unchanged.

- virtual void setTargetProfile (const IDOMICCProfilePtr &profile)=0

  Set the target color space for the output using an ICC profile. The default behaviour is to, where possible, leave the color space of objects unchanged.

- virtual void **applyColorConverterTransform** (const IColorConverterTransformPtr &transform)=0

  Apply the given color converter transform to the contents before writing XAML. This supersedes the target color space parameters described above. This allows for more advanced configuration of the color spaces of the output.

- virtual void setRenderResolution (uint32 resolution)=0

  Set the resolution to use if page content requires rendering in order to be output as XAML. The default is 150dpi. This is affected also by the maximum image resolution parameters.

- virtual void **applyRendererTransform** (const IRendererTransformPtr &transform)=0

  Apply the given renderer transform to the contents before writing XAML. This supersedes the target color space parameters described above. This allows for more advanced configuration of rendering.

- virtual void setPreferredColorImageFormat (IImageEncoderTransform::eEncodeFormat format)=0

  Set the desired image format for color images that need to be reencoded for XAML output. The default is eEFAuto. Note: this is advisory only and may not be honoured in all cases if the image cannot be represented in XAML in the desired form.

- virtual void setPreferredGrayImageFormat (IImageEncoderTransform::eEncodeFormat format)=0

  Set the desired image format for gray images that need to be reencoded for XAML output. The default is eEFAuto. Note: this is advisory only and may not be honoured in all cases if the image cannot be represented in XPS in the desired form.

- virtual void setPreferredMonoImageFormat (IImageEncoderTransform::eEncodeFormat format)=0

  Set the desired image format for monochrome images that need to be reencoded for XAML output. The default is eEFAuto. Note: this is advisory only and may not be honoured in all cases if the image cannot be represented in XPS in the desired form.

- virtual void setJPEGQuality (uint8 quality)=0

  Set the JPEG quality to use when encoding images in JPEG format. Equivalent to calling setParameter() with the parameter name "JPEGQuality" and the value being the required quality.
- virtual void **applyEncoderTransform** (const IImageEncoderTransformPtr &transform)=0

  Apply the given image encoder transform to the contents before writing to XAML. This supersedes the image encoding parameters described above. This allows for more advanced configuration of image encoding.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IXAMLGeneratorPtr create (const IJawsMakoPtr &jawsMako, const U8String &resourcePrefix=U8String(), const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create a XAML generator instance.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.436.1 Detailed Description

A XAML generator for Mako, allowing simple generation of XAML fragments for individual DOM nodes or entire pages.

The XAML is provided in a stream, with resources used by the XAML presented as streams tracked by instances of this object, which can be requested via getResource(). Resources are reused where possible for multiple XAML fragments.

7.436.2 Member Function Documentation

create()

```
static JAWSMAKO_API IXAMLGeneratorPtr JawsMako::IXAMLGenerator::create (
            const IJawsMakoPtr & jawsMako,
            const U8String & resourcePrefix = U8String(),
            const IProgressMonitorPtr & progressMonitor = IProgressMonitorPtr() )  [static]
```

Create a XAML generator instance.

Parameters

jawsMako	The JawsMako instance.
resourcePrefix	A path fragment to prepend to resource names when generating XAML. Useful for cases where the resources are to be written to disk somewhere else than alongside the XAML fragment.
progressMonitor	The progress monitor which allows aborting an operation or registering a progress callback.

Returns

IXAMLGeneratorPtr The new instance.

generateXAML() [1/2]

```
virtual IRAInputStreamPtr JawsMako::IXAMLGenerator::generateXAML (
            const IDOMNodePtr & node,
            const IDOMCatalogPtr & catalog = IDOMCatalogPtr() )  [pure virtual]
```

Generate XAML for the given DOM Node, returning the result in a stream.

Parameters

node	The DOM node to be represented as XAML.
catalog	The catalog associated with the node. Optional, but required if the names of named elements are to be present in the generated XAML.

Returns

IRAInputStreamPtr The XAML stream.

generateXAML() [2/2]

```
virtual void JawsMako::IXAMLGenerator::generateXAML (
            const IDOMNodePtr & node,
            const IOutputStreamPtr & outputStream,
            const IDOMCatalogPtr & catalog = IDOMCatalogPtr() )  [pure virtual]
```

Alternate form of generateXAML() when an existing stream should be used.

Parameters

node	The DOM node to be represented as XAML.
outputStream	The destination stream for the XAML.
catalog	The catalog associated with the node. Optional, but required if the names of named elements are to be present in the generated XAML.

generateXAMLForAppearance() [1/2]

```
virtual IRAInputStreamPtr JawsMako::IXAMLGenerator::generateXAMLForAppearance (
            const IAnnotationAppearancePtr & appearance,
            const IAnnotationPtr & annotation,
            const IPagePtr & page )  [pure virtual]
```

Generate XAML for a given annotation appearance. Provide the annotation and page from which the appearance comes. The page is required as some annotations may need to be composited against the page backdrop. This is slower than invoking generateXAMLForPageAnnotationAppearances() which is in turn slower than using generate↩ XAMLForPageAndAnnotations().

Parameters

appearance	The appearance to be presented as XAML.
annotation	The annotation that contains the appearance. Required in order to position the annotation against the page background correctly.
page	The page the appearance appears on.

generateXAMLForAppearance() [2/2]

```
virtual void JawsMako::IXAMLGenerator::generateXAMLForAppearance (
            const IAnnotationAppearancePtr & appearance,
            const IAnnotationPtr & annotation,
            const IPagePtr & page,
            const IOutputStreamPtr & outputStream )  [pure virtual]
```

Alternate form of generateXAMLForAppearance() where an existing stream should be used. Provide the annotation and page from which the appearance comes. The page is required as some annotations may need to be composited against the page backdrop. This is slower than invoking generateXAMLForPageAnnotationAppearances() which is in turn slower than using generateXAMLForPageAndAnnotations().

Parameters

appearance	The appearance to be presented as XAML.
annotation	The annotation that contains the appearance. Required in order to position the annotation against the page background correctly.
page	The page the appearance appears on.

generateXAMLForPageAndAnnotationAppearances()

```
virtual IRAInputStreamPtr JawsMako::IXAMLGenerator::generateXAMLForPageAndAnnotationAppearances
(
            const IPagePtr & page,
            CAnnotationXAMLVect & appearanceXAMLs )  [pure virtual]
```

Generate XAML for the given IPage and all the annotation appearances present on the page. The XAML for the page will be in the returned stream, and appearanceXAMLs will be populated with the XAML for all annotation appearances. This will usually be faster than using generateXAML() on the page contents and then separately invoking generateXAMLForPageAnnotationAppearances().

Parameters

page	The DOM node to be produced as XAML
appearanceXAMLs	A vector to receive the XAML for the annotation appearances.

Returns

IRAInputStreamPtr The XAML stream.

generateXAMLForPageAnnotationAppearances()

```
virtual void JawsMako::IXAMLGenerator::generateXAMLForPageAnnotationAppearances (
            const IPagePtr & page,
            CAnnotationXAMLVect & appearanceXAMLs )  [pure virtual]
```

Generate XAML for all the annotation appearances on the given page. This is generally more efficient than using generateXAMLForAppearance() repeatedly for every page, but not as fast as using generateXAMLForPageAnd↩ Annotations() to generate XAML for the page and the annotations in one bulk operation.

Parameters

page	The page whose annotations should be represented as XAML.
appearanceXAMLs	A vector to receive the XAML for the annotation appearances.

getResource()

```
virtual IInputStreamPtr JawsMako::IXAMLGenerator::getResource (
            const U8String & name )  [pure virtual]
```

Get the stream for a named resource. An exception will be thrown if the resource cannot be found.

Parameters

name	The name of the resource.

Returns

IInputStreamPtr The resource stream.

getResources()

```
virtual void JawsMako::IXAMLGenerator::getResources (
            CEDLVector< CResourceEntry > & resources )  [pure virtual]
```

Get all the resources in a vector.

Parameters

resources	A reference to a vector to receive the entries.

setColorImageMaxResolution()

```
virtual void JawsMako::IXAMLGenerator::setColorImageMaxResolution (
            float resolution,
            float threshold = 0.0f,
            IDOMImageDownsamplerFilter::eDownsamplingMethod method = IDOMImageDownsamplerFilter↩
::eBicubic )  [pure virtual]
```

Set the desired maximum resolution, threshold and downsampling method for colour images for XAML output.

The default behaviour is leave the image resolution unchanged.

Parameters

resolution	The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default.
threshold	The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution.
method	The method to use when downsampling. The default is bicubic for colour images.

setGrayImageMaxResolution()

```
virtual void JawsMako::IXAMLGenerator::setGrayImageMaxResolution (
            float resolution,
            float threshold = 0.0f,
            IDOMImageDownsamplerFilter::eDownsamplingMethod method = IDOMImageDownsamplerFilter↩
::eBicubic )  [pure virtual]
```

Set the desired maximum resolution, threshold and downsampling method for gray images.

The default behaviour is leave the image resolution unchanged.

Parameters

resolution	The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default.
threshold	The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution.
method	The method to use when downsampling. The default is bicubic for gray images.

setJPEGQuality()

```
virtual void JawsMako::IXAMLGenerator::setJPEGQuality (
            uint8 quality )  [pure virtual]
```

Set the JPEG quality to use when encoding images in JPEG format. Equivalent to calling setParameter() with the parameter name "JPEGQuality" and the value being the required quality.

Parameters

quality	The desired quality level, with 1 being lowest quality and 5 being highest quality.

setMergeFonts()

```
virtual void JawsMako::IXAMLGenerator::setMergeFonts (
            bool merge )  [pure virtual]
```

Set whether or not an attempt will be made to merge disparate subsets of a font into a single font.

The default is false. Equivalent to calling setParameter with "MergeFonts" as the parameter name.

Some formats such as PostScript and XPS tend to include many font subsets and for output it is often avantageous to attempt to merge these fonts into a single font where possible.

Note that it is possible to enable both font subsetting and merging at the same time. In this case merging happens first, followed by subsetting of the merged results.

setMonoImageMaxResolution()

```
virtual void JawsMako::IXAMLGenerator::setMonoImageMaxResolution (
            float resolution,
            float threshold = 0.0f,
            IDOMImageDownsamplerFilter::eDownsamplingMethod method = IDOMImageDownsamplerFilter↩
::eSubsample )  [pure virtual]
```

Set the desired maximum resolution, threshold and downsampling method for monochrome images.

The default behaviour is leave the image resolution unchanged.

Parameters

resolution	The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default.
threshold	The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution.
method	The method to use when downsampling. The default is subsample for monochrome images; using any other method will result in grayscale output.

setPreferredColorImageFormat()

```
virtual void JawsMako::IXAMLGenerator::setPreferredColorImageFormat (
            IImageEncoderTransform::eEncodeFormat format )  [pure virtual]
```

Set the desired image format for color images that need to be reencoded for XAML output. The default is eEFAuto. Note: this is advisory only and may not be honoured in all cases if the image cannot be represented in XAML in the desired form.

Parameters

format	The desired format.

setPreferredGrayImageFormat()

```
virtual void JawsMako::IXAMLGenerator::setPreferredGrayImageFormat (
            IImageEncoderTransform::eEncodeFormat format ) [pure virtual]
```

Set the desired image format for gray images that need to be reencoded for XAML output. The default is eEFAuto. Note: this is advisory only and may not be honoured in all cases if the image cannot be represented in XPS in the desired form.

Parameters

format	The desired format.

setPreferredMonoImageFormat()

```
virtual void JawsMako::IXAMLGenerator::setPreferredMonoImageFormat (
            IImageEncoderTransform::eEncodeFormat format ) [pure virtual]
```

Set the desired image format for monochrome images that need to be reencoded for XAML output. The default is eEFAuto. Note: this is advisory only and may not be honoured in all cases if the image cannot be represented in XPS in the desired form.

Parameters

format	The desired format.

setRenderResolution()

```
virtual void JawsMako::IXAMLGenerator::setRenderResolution (
            uint32 resolution ) [pure virtual]
```

Set the resolution to use if page content requires rendering in order to be output as XAML. The default is 150dpi. This is affected also by the maximum image resolution parameters.

Parameters

resolution	The desired resolution in dpi.

setSubsetFonts()

```
virtual void JawsMako::IXAMLGenerator::setSubsetFonts (
            bool subset ) [pure virtual]
```

Set whether fonts should be subset in the output.

```
The default is false. Equivalent to calling setParameter with
"SubsetFonts" as the parameter name.

Note that all fonts may not be subsetted; for some subsetting may be
forbidden, and for others there may be little gain in subsetting.
```

setTargetColorSpace()

```
virtual void JawsMako::IXAMLGenerator::setTargetColorSpace (
            const IDOMColorSpacePtr & targetSpace )  [pure virtual]
```

Set the target color space for the output. The default behaviour is to, where possible, leave the color space of objects unchanged.

Parameters

targetSpace	The desired color space. Must be a simple or ICC space.

setTargetProfile()

```
virtual void JawsMako::IXAMLGenerator::setTargetProfile (
            const IDOMICCProfilePtr & profile )  [pure virtual]
```

Set the target color space for the output using an ICC profile. The default behaviour is to, where possible, leave the color space of objects unchanged.

Parameters

profile	The desired profile.

The documentation for this class was generated from the following file:

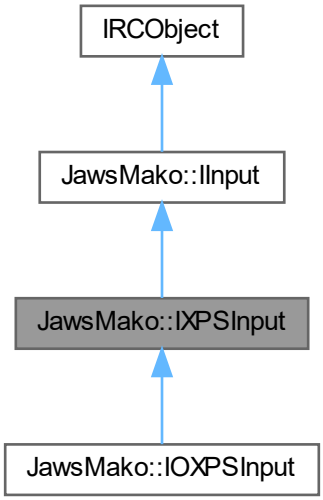
- xamlgenerator.h

7.437 JawsMako::IXPSInput Class Reference

An instance of the JawsMako XPS input class.

```
#include <xpsinput.h>
```

Inheritance diagram for JawsMako::IXPSInput:



Public Member Functions

- virtual IDocumentAssemblyPtr openStreaming (const IInputStreamPtr &inputStream)=0

  Open a stream, returning the IDocumentAssembly representing the contents, in streaming mode.

Public Member Functions inherited from JawsMako::IInput

- virtual IDocumentAssemblyPtr open (const U8String &pathToFile)=0

  Open a file on disk, returning the IDocumentAssembly representing the contents.
- virtual IDocumentAssemblyPtr open (const String &pathToFile)=0

  Open a file on disk, returning the IDocumentAssembly representing the contents. Takes a wide character string.
- virtual IDocumentAssemblyPtr open (const IInputStreamPtr &inputStream)=0

  Open a stream, returning the IDocumentAssembly representing the contents.
- virtual void setSequentialMode (bool sequential)=0

  Set/unset sequential mode on this input.
- virtual void setParameter (const U8String &param, const U8String &value)=0

  Apply a key value pair output parameter with a string value. The parameter name is case insensitive. Please refer to the supplied documentation for the details of the available parameters and their ranges.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IXPSInputPtr create (const IJawsMakoPtr &jawsMako, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create an input for reading source documents in XPS format.

Static Public Member Functions inherited from JawsMako::IInput

- static JAWSMAKO_API IInputPtr create (const IJawsMakoPtr &jawsMako, eFileFormat format, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create an input for reading source documents in the given format. The following formats are currently supported:

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.437.1 Detailed Description

An instance of the JawsMako XPS input class.

7.437.2 Member Function Documentation

create()

```
static JAWSMAKO_API IXPSInputPtr JawsMako::IXPSInput::create (
            const IJawsMakoPtr & jawsMako,
            const IProgressMonitorPtr & progressMonitor = IProgressMonitorPtr() )  [static]
```

Create an input for reading source documents in XPS format.

Parameters

jawsMako	The IJawsMako object
progressMonitor	A smart pointer to an IProgressMonitor object which can be NULL if no such object was passed in.

Returns

IXPSInputPtr the XPS input

openStreaming()

```
virtual IDocumentAssemblyPtr JawsMako::IXPSInput::openStreaming (
            const IInputStreamPtr & inputStream )  [pure virtual]
```

Open a stream, returning the IDocumentAssembly representing the contents, in streaming mode.

This is designed to support cases where we wish to begin working with the XPS before the entire stream has been received, such as within a Windows XPS print driver or if the XPS is being downloaded from a remote server.

The assembly returned by this function is opened once the most basic parts of the XPS stream have become available; for a properly interleaved XPS stream this should not required much data to be received.

Access to the assembly can still be in a random fashion, but a request for a document or page will block until that document becomes available in the incoming stream.

Note that there are several operations that will cause the a wait until the entire XPS input has been received. These include:

- Document manipulation (any attempt to insert or remove a document)
- Page manipulation (any attempt to insert or remove a page)
- Requesting the number of pages or documents.
- Searching for a target in a document or assembly.

Instead of requesting the number of pages or documents, it is reccommended that the documents and pages are requested sequentially, and if a page or document is not available an IError exception with error code JM↩ _ERR_PAGE_NOT_FOUND or JM_ERR_DOCUMENT_NOT_FOUND will be thrown. These can be caught and appropriately handled.

Returns

IDocumentAssemblyPtr the document assembly.

The documentation for this class was generated from the following file:

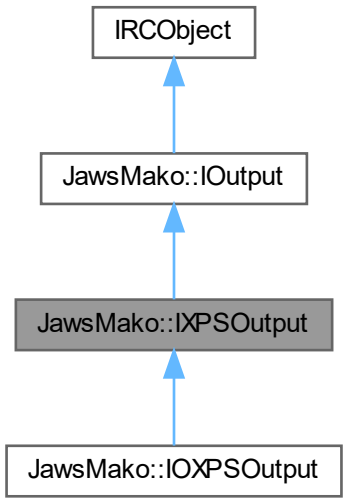
- xpsinput.h

7.438 JawsMako::IXPSOutput Class Reference

Interface for the XPS IOutput class.

`#include <xpsoutput.h>`

Inheritance diagram for JawsMako::IXPSOutput:



Public Member Functions

- virtual void **setCopyExistingXPSPartsWherePossible** (bool copy)=0

  Set whether or not entire XPS parts should be copied from the original assembly source when writing that assembly if the part remains unchanged. This is almost always faster. This defaults to true, and is set to true when using the "Preserve" preset. However, any change to configuration will cause this to be set to false in order to force reprocessing for the configuration to take effect. However this setting may be set to true after other configuration has been changed.
- virtual void setSubsetFonts (bool subset)=0

  Set whether fonts should be subset in the output.
- virtual void setMergeFonts (bool merge)=0

  Set whether or not an attempt will be made to merge disparate subsets of a font into a single font.
- virtual void **setMergeImages** (bool merge=true)=0

  Set if the XPS output should attempt to merge adjacent images. Equivalent to calling setParameter with "MergeAdjacentImages" as the parameter name. The default is true.
- virtual void setRenameFonts (bool rename=true)=0

  Set if the XPS output should rename fonts to aid with problematic print environments.
- virtual void setColorImageMaxResolution (float resolution, float threshold=0.0f, IDOMImageDownsamplerFilter::eDownsampling method=IDOMImageDownsamplerFilter::eBicubic)=0

  Set the desired maximum resolution, threshold and downsampling method for colour images.
- virtual void setGrayImageMaxResolution (float resolution, float threshold=0.0f, IDOMImageDownsamplerFilter::eDownsampling method=IDOMImageDownsamplerFilter::eBicubic)=0

  Set the desired maximum resolution, threshold and downsampling method for gray images.
- virtual void setMonoImageMaxResolution (float resolution, float threshold=0.0f, IDOMImageDownsamplerFilter::eDownsampling method=IDOMImageDownsamplerFilter::eSubsample)=0

  Set the desired maximum resolution, threshold and downsampling method for monochrome images.
- virtual void setTargetColorSpace (const IDOMColorSpacePtr &targetSpace)=0

  Set the target color space for the output. The default behaviour is to, where possible, leave the color space of objects unchanged. Equivalent to calling setParameter with the param name "TargetColorSpace" with appropriate values (please refer to documentation).

- virtual void setTargetProfile (const IDOMICCProfilePtr &profile)=0

  Set the target color space for the output using an ICC profile. The default behaviour is to, where possible, leave the color space of objects unchanged. Equivalent to calling setParameter() with the param name "TargetProfile" with the value as the path to the profile.
- virtual void **applyColorConverterTransform** (const IColorConverterTransformPtr &transform)=0

  Apply the given color converter transform to the contents before writing to XPS. This supersedes the target color space parameters described above. This allows for more advanced configuration of the color spaces of the output.
- virtual void setRenderResolution (uint32 resolution)=0

  Set the resolution to use if page content requires rendering in order to be output as XPS. The default is 150dpi. This is affected also by the maximum image resolution parameters. Equivalent to calling setParameter() with param name "RenderResolution" with the value as the desired resolution as value.
- virtual void **applyRendererTransform** (const IRendererTransformPtr &transform)=0

  Apply the given renderer transform to the contents before writing to XPS. This supersedes the target color space parameters described above. This allows for more advanced configuration of rendering.
- virtual void setPreferredColorImageFormat (IImageEncoderTransform::eEncodeFormat format)=0

  Set the desired image format for color images that need to be reencoded for XPS output. The default is eEFAuto. Note: this is advisory only and may not be honoured in all cases if the image cannot be represented in XPS in the desired form. Equivalent to calling setParameter() with the parameter name "ColorImageFormat" with appropriate values (please refer to documentation).
- virtual void setPreferredGrayImageFormat (IImageEncoderTransform::eEncodeFormat format)=0

  Set the desired image format for gray images that need to be reencoded for XPS output. The default is eEFAuto. Note: this is advisory only and may not be honoured in all cases if the image cannot be represented in XPS in the desired form. Equivalent to calling setParameter() with the parameter name "GrayImageFormat" with appropriate values (please refer to documentation).
- virtual void setPreferredMonoImageFormat (IImageEncoderTransform::eEncodeFormat format)=0

  Set the desired image format for monochrome images that need to be reencoded for XPS output. The default is eEFAuto. Note: this is advisory only and may not be honoured in all cases if the image cannot be represented in XPS in the desired form. Equivalent to calling setParameter() with the parameter name "MonoImageFormat" with appropriate values (please refer to documentation).
- virtual void setJPEGQuality (uint8 quality)=0

  Set the JPEG quality to use when encoding images in JPEG format. Equivalent to calling setParameter() with the parameter name "JPEGQuality" and the value being the required quality.
- virtual void **applyEncoderTransform** (const IImageEncoderTransformPtr &transform)=0

  Apply the given image encoder transform to the contents before writing to XPS. This supersedes the image encoding parameters described above. This allows for more advanced configuration of image encoding.

Public Member Functions inherited from JawsMako::IOutput

- virtual void **setPreset** (const U8String &preset)=0

  Configure the output according to a general preset. Please see the supplied documentation for details of these presets. The default is "Preserve" which will attempt to produce output as close to the input as possible for the output format. A string value can be used for any parameter and will be converted as necessary.
- virtual void **setParameter** (const U8String &param, const U8String &value)=0

  Apply a key value pair output parameter with a string value. The parameter name is case insensitive. Please refer to the supplied documentation for the details of the available parameters and their ranges.
- virtual void setAllowedPermissionsFlags (uint32 allowedPermissions)=0

  Control whether or not assemblies with certain security permission flags are allowed to be written by this output.
- virtual void writeAssembly (const IDocumentAssemblyPtr &assembly, const U8String &pathToFile)=0

  Write the given document assembly to a file on disk.
- virtual void writeAssembly (const IDocumentAssemblyPtr &assembly, const String &pathToFile)=0

  Write the given document assembly to a file on disk, specified by a wide character string.
- virtual void writeAssembly (const IDocumentAssemblyPtr &assembly, const IOutputStreamPtr &stream)=0

  Write the given document assembly to a stream.
- virtual IOutputWriterPtr openWriter (const IDocumentAssemblyPtr &assembly, const U8String &pathToFile)=0

Create an output writer for the given assembly, targeting a file on disk. This is designed to allow streaming output, or to deal with situations where an operation would require too much memory to hold an entire edited assembly in memory at once.

- virtual IOutputWriterPtr openWriter (const IDocumentAssemblyPtr &assembly, const String &pathToFile)=0

  Create an output writer for the given assembly, targeting a file on disk. As above, but with the file specified in a wide character string.

- virtual IOutputWriterPtr openWriter (const IDocumentAssemblyPtr &assembly, const IOutputStreamPtr &stream)=0

  Create an output writer for the given assembly, targeting a stream.

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.

- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.

- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

- static JAWSMAKO_API IXPSOutputPtr create (const IJawsMakoPtr &jawsMako, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create an XPS Output instance.

Static Public Member Functions inherited from JawsMako::IOutput

- static JAWSMAKO_API IOutputPtr create (const IJawsMakoPtr &jawsMako, eFileFormat format, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

  Create an output for writing source in the given format.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()

  Virtual destructor.

7.438.1 Detailed Description

Interface for the XPS IOutput class.

7.438.2 Member Function Documentation

create()

```
static JAWSMAKO_API IXPSOutputPtr JawsMako::IXPSOutput::create (
            const IJawsMakoPtr & jawsMako,
            const IProgressMonitorPtr & progressMonitor = IProgressMonitorPtr() )  [static]
```

Create an XPS Output instance.

Parameters

jawsMako	The IJawsMako object.
progressMonitor	A progress monitor to allow aborting an operation or to register a progress callback.

Returns

IXPSOutputPtr The XPS output.

setColorImageMaxResolution()

```
virtual void JawsMako::IXPSOutput::setColorImageMaxResolution (
            float resolution,
            float threshold = 0.0f,
            IDOMImageDownsamplerFilter::eDownsamplingMethod method = IDOMImageDownsamplerFilter↩
::eBicubic )  [pure virtual]
```

Set the desired maximum resolution, threshold and downsampling method for colour images.

The default behaviour is leave the image resolution unchanged. Equivalent to calling setParameter() with the param names "ColorImageDownsamplingResolution", "ColorImageDownsamplingThreshold" and "ColorImage↩ DownsamplingMethod" with the respective values.

Parameters

resolution	The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default.
threshold	The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution.
method	The method to use when downsampling. The default is bicubic for colour images.

setGrayImageMaxResolution()

```
virtual void JawsMako::IXPSOutput::setGrayImageMaxResolution (
            float resolution,
            float threshold = 0.0f,
            IDOMImageDownsamplerFilter::eDownsamplingMethod method = IDOMImageDownsamplerFilter↩
::eBicubic )  [pure virtual]
```

Set the desired maximum resolution, threshold and downsampling method for gray images.

The default behaviour is leave the image resolution unchanged. Equivalent to calling setParameter() with the param names "GrayImageDownsamplingResolution", "GrayImageDownsamplingThreshold" and "GrayImage↩ DownsamplingMethod" with the respective values.

Parameters

resolution	The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default.
threshold	The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution.
method	The method to use when downsampling. The default is bicubic for gray images.

setJPEGQuality()

```
virtual void JawsMako::IXPSOutput::setJPEGQuality (
            uint8 quality )  [pure virtual]
```

Set the JPEG quality to use when encoding images in JPEG format. Equivalent to calling setParameter() with the parameter name "JPEGQuality" and the value being the required quality.

Parameters

quality	The desired quality level, with 1 being lowest quality and 5 being highest quality.

setMergeFonts()

```
virtual void JawsMako::IXPSOutput::setMergeFonts (
            bool merge )  [pure virtual]
```

Set whether or not an attempt will be made to merge disparate subsets of a font into a single font.

The default is false. Equivalent to calling setParameter with "MergeFonts" as the parameter name.

Some formats such as PostScript and XPS tend to include many font subsets and for output it is often avantageous to attempt to merge these fonts into a single font where possible.

Note that it is possible to enable both font subsetting and merging at the same time. In this case merging happens first, followed by subsetting of the merged results.

setMonoImageMaxResolution()

```
virtual void JawsMako::IXPSOutput::setMonoImageMaxResolution (
            float resolution,
            float threshold = 0.0f,
            IDOMImageDownsamplerFilter::eDownsamplingMethod method = IDOMImageDownsamplerFilter↩
::eSubsample )  [pure virtual]
```

Set the desired maximum resolution, threshold and downsampling method for monochrome images.

The default behaviour is leave the image resolution unchanged. Equivalent to calling setParameter() with the param names "MonoImageDownsamplingResolution", "MonoImageDownsamplingThreshold" and "MonoImage↩ DownsamplingMethod" with the respective values.

Parameters

resolution	The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default.
threshold	The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution.
method	The method to use when downsampling. The default is subsample for monochrome images; using any other method will result in grayscale output.

setPreferredColorImageFormat()

```
virtual void JawsMako::IXPSOutput::setPreferredColorImageFormat (
            IImageEncoderTransform::eEncodeFormat format )  [pure virtual]
```

Set the desired image format for color images that need to be reencoded for XPS output. The default is eEFAuto. Note: this is advisory only and may not be honoured in all cases if the image cannot be represented in XPS in the desired form. Equivalent to calling setParameter() with the parameter name "ColorImageFormat" with appropriate values (please refer to documentation).

Parameters

format	The desired format.

setPreferredGrayImageFormat()

```
virtual void JawsMako::IXPSOutput::setPreferredGrayImageFormat (
            IImageEncoderTransform::eEncodeFormat format )  [pure virtual]
```

Set the desired image format for gray images that need to be reencoded for XPS output. The default is eEFAuto. Note: this is advisory only and may not be honoured in all cases if the image cannot be represented in XPS in the desired form. Equivalent to calling setParameter() with the parameter name "GrayImageFormat" with appropriate values (please refer to documentation).

Parameters

format	The desired format.

setPreferredMonoImageFormat()

```
virtual void JawsMako::IXPSOutput::setPreferredMonoImageFormat (
            IImageEncoderTransform::eEncodeFormat format )  [pure virtual]
```

Set the desired image format for monochrome images that need to be reencoded for XPS output. The default is eEFAuto. Note: this is advisory only and may not be honoured in all cases if the image cannot be represented in XPS in the desired form. Equivalent to calling setParameter() with the parameter name "MonoImageFormat" with appropriate values (please refer to documentation).

Parameters

format	The desired format.

setRenameFonts()

```
virtual void JawsMako::IXPSOutput::setRenameFonts (
            bool rename = true )  [pure virtual]
```

Set if the XPS output should rename fonts to aid with problematic print environments.

This is useful when the produced XPS is being sent to a print pipeline; some XPS drivers may have problems dealing with multiple fonts with the same name. If true, every font will be renamed and all chosen names will be unique. A PDF-style tag will be prefixed, and the font name limited to 63 characters.

Equivalent to calling setParameter with "RenameFonts" as the parameter name.

This is not required for regular XPS output and as a result the default is false.

setRenderResolution()

```
virtual void JawsMako::IXPSOutput::setRenderResolution (
            uint32 resolution )  [pure virtual]
```

Set the resolution to use if page content requires rendering in order to be output as XPS. The default is 150dpi. This is affected also by the maximum image resolution parameters. Equivalent to calling setParameter() with param name "RenderResolution" with the value as the desired resolution as value.

Parameters

resolution	The desired resolution in dpi.

setSubsetFonts()

```
virtual void JawsMako::IXPSOutput::setSubsetFonts (
            bool subset )  [pure virtual]
```

Set whether fonts should be subset in the output.

```
The default is false. Equivalent to calling setParameter with
"SubsetFonts" as the parameter name.

Note that all fonts may not be subsetted; for some subsetting may be
forbidden, and for others there may be little gain in subsetting.
```

setTargetColorSpace()

```
virtual void JawsMako::IXPSOutput::setTargetColorSpace (
            const IDOMColorSpacePtr & targetSpace )  [pure virtual]
```

Set the target color space for the output. The default behaviour is to, where possible, leave the color space of objects unchanged. Equivalent to calling setParameter with the param name "TargetColorSpace" with appropriate values (please refer to documentation).

Parameters

targetSpace	The desired color space. Must be a simple or ICC space.

setTargetProfile()

```
virtual void JawsMako::IXPSOutput::setTargetProfile (
            const IDOMICCProfilePtr & profile )  [pure virtual]
```

Set the target color space for the output using an ICC profile. The default behaviour is to, where possible, leave the color space of objects unchanged. Equivalent to calling setParameter() with the param name "TargetProfile" with the value as the path to the profile.

Parameters

profile	The desired profile.

The documentation for this class was generated from the following file:

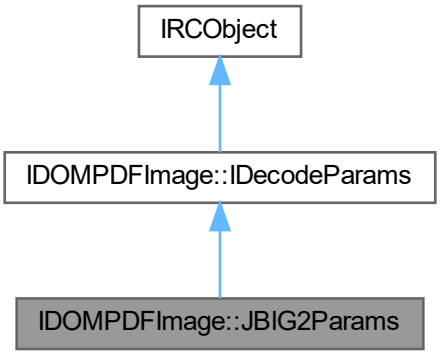
- xpsoutput.h

7.439 IDOMPDFImage::JBIG2Params Class Reference

Class to hold filter parameters for JBIG2-compressed image data. Please see the PDF specification for the meaning of these parameters.

`#include <idomimageresource.h>`

Inheritance diagram for IDOMPDFImage::JBIG2Params:



Additional Inherited Members

Public Member Functions inherited from IRCObject

- virtual void **addRef** () const =0

  Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool decRef () const =0

  Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
- virtual int32 getRefCount () const =0

  Retrieve the current reference count of the actual object pointed to.

Protected Member Functions inherited from IRCObject

- virtual ~**IRCObject** ()
  Virtual destructor.

7.439.1 Detailed Description

Class to hold filter parameters for JBIG2-compressed image data. Please see the PDF specification for the meaning of these parameters.

The documentation for this class was generated from the following file:

- idomimageresource.h

7.440 PointTmpl< PointType > Class Template Reference

Geometry primitives including: point, rectangle and matrix types supporting both integer and floating point values within.

`#include <edlgeom.h>`

Public Member Functions

- PointTmpl< PointType > **getMidPoint** (const PointTmpl< PointType > &pt)
  Find the mid-point between this and another point.
- double **getDistance** (const PointTmpl< PointType > &pt) const
  Find the distance to another point.

7.440.1 Detailed Description

template<typename PointType>
class PointTmpl< PointType >

Geometry primitives including: point, rectangle and matrix types supporting both integer and floating point values within.

Template for a point specified with X and Y coordinates

The documentation for this class was generated from the following file:

- edlgeom.h

7.441 PValue Class Reference

Stores a "property" value that is tagged with an enumeration value that indicates the underlying type.

`#include <edlproperty.h>`

7.441.1 Detailed Description

Stores a "property" value that is tagged with an enumeration value that indicates the underlying type.

The documentation for this class was generated from the following file:

- edlproperty.h

7.442 SignatureID Class Reference

Opentype table signatures.

`#include <idomfont.h>`

7.442.1 Detailed Description

Opentype table signatures.

The documentation for this class was generated from the following file:

- idomfont.h

8 File Documentation

Index

GLOBAL GRAPHICS® software

Global Graphics Software Ltd
Building 2030
Cambourne Business Park
Cambourne, Cambridge
CB23 6DW UK
Tel: +44 (0)1954 283100

Global Graphics Software Inc
5996 Clark Center Avenue
Sarasota, FL 34238
United States of America
Tel: +1(941)925-1303

Global Graphics KK
610 AIOS Nagatacho Bldg
2-17-17 Nagatacho, Chiyoda-ku
Tokyo 100-0014
Japan
Tel: +81-3-6273-3198

www.globalgraphics.com