



MAKO™

API Reference

Version 7.4.0

1 Version 7.4.0 API Documentation	1
1.1 Introduction	1
1.2 Contents	2
2 Topic Index	3
3 Hierarchical Index	3
3.1 Class Hierarchy	3
4 Class Index	17
4.1 Class List	17
5 File Index	43
5.1 File List	43
6 Topic Documentation	45
6.1 JawsMako API	45
6.1.1 Detailed Description	46
6.1.2 JawsMako Library	46
6.1.3 Error handling	49
6.1.4 DOM Objects	53
6.1.5 Input/Output	81
6.1.6 Interactive features	83
6.1.7 Utility	87
6.1.8 Rendering	98
6.1.9 Layout	100
6.1.10 Colors	101
6.1.11 Validate	104
6.1.12 Text	104
6.1.13 Types	105
6.2 Separator	105
6.2.1 Detailed Description	105
7 Class Documentation	106
7.1 BoxTpl< PointType > Class Template Reference	106
7.1.1 Detailed Description	106
7.2 JawsMako::CAnnotationBorder Class Reference	106
7.2.1 Detailed Description	107
7.2.2 Member Enumeration Documentation	107
7.2.3 Constructor & Destructor Documentation	107
7.2.4 Member Data Documentation	107
7.3 JawsMako::IXAMLGenerator::CAnnotationXAML Class Reference	108
7.3.1 Detailed Description	108
7.4 IDOMFontOpenType::CCIDMap Class Reference	108

7.4.1 Detailed Description	109
7.4.2 Member Data Documentation	109
7.5 IDOMPdfImage::CCITTFaxParams Class Reference	110
7.5.1 Detailed Description	111
7.6 CClassID Class Reference	111
7.6.1 Detailed Description	111
7.6.2 Constructor & Destructor Documentation	111
7.6.3 Member Function Documentation	112
7.7 CClassParams Class Reference	113
7.7.1 Detailed Description	114
7.8 JawsMako::IJawsRenderer::CColorSpotHalftone Class Reference	115
7.8.1 Detailed Description	115
7.9 JawsMako::IJawsRenderer::CColorThresholdArrayHalftone Class Reference	115
7.9.1 Detailed Description	116
7.10 JawsMako::IPDFValidator::CContentError Class Reference	116
7.10.1 Detailed Description	116
7.11 JawsMako::IJawsRenderer::CEDSHalftone Class Reference	116
7.11.1 Detailed Description	117
7.12 JawsMako::CFieldOption Class Reference	117
7.12.1 Detailed Description	118
7.13 JawsMako::IJawsRenderer::CFrameBufferInfo Class Reference	118
7.13.1 Detailed Description	118
7.13.2 Member Data Documentation	118
7.14 JawsMako::IPDFValidator::CGeneralError Class Reference	118
7.14.1 Detailed Description	119
7.15 CGlyphsCluster Class Reference	119
7.15.1 Detailed Description	119
7.15.2 Member Function Documentation	119
7.16 CGlyphsClusters Class Reference	120
7.16.1 Detailed Description	120
7.17 CIndicesGlyph Class Reference	120
7.17.1 Detailed Description	121
7.17.2 Member Function Documentation	121
7.18 JawsMako::ILayoutFont::CLayoutFontItem Class Reference	122
7.18.1 Detailed Description	122
7.19 IDOMShadingPatternType4567Brush::CMeshEntry Class Reference	122
7.19.1 Detailed Description	122
7.19.2 Member Data Documentation	122
7.20 JawsMako::IOptionalContentConfiguration::COrderEntry Class Reference	123
7.20.1 Detailed Description	124
7.21 JawsMako::IPDFValidator::CPageErrors Class Reference	124
7.21.1 Detailed Description	124

7.22 JawsMako::CPDFFarReference Class Reference	124
7.22.1 Detailed Description	125
7.22.2 Constructor & Destructor Documentation	125
7.22.3 Member Function Documentation	125
7.23 JawsMako::IPDFInput::CPdfFontInfo Class Reference	126
7.23.1 Detailed Description	127
7.23.2 Member Data Documentation	127
7.24 JawsMako::CPDFReference Class Reference	127
7.24.1 Detailed Description	128
7.24.2 Constructor & Destructor Documentation	128
7.24.3 Member Function Documentation	128
7.25 JawsMako::IPDFInput::CPdfScannedInk Class Reference	129
7.25.1 Detailed Description	129
7.25.2 Member Data Documentation	129
7.26 JawsMako::IPJLParser::CPjlAttributeValue Class Reference	129
7.26.1 Detailed Description	129
7.26.2 Member Data Documentation	130
7.27 JawsMako::CQuadPoint Class Reference	130
7.27.1 Detailed Description	130
7.28 JawsMako::CRectInset Class Reference	130
7.28.1 Detailed Description	131
7.29 JawsMako::ISVGGenerator::CResourceEntry Class Reference	131
7.29.1 Detailed Description	131
7.30 JawsMako::IXAMLGenerator::CResourceEntry Class Reference	131
7.30.1 Detailed Description	131
7.31 IDOMShape::CShapeDetails Class Reference	131
7.31.1 Detailed Description	132
7.32 IDOMShape::CShapeDetails::Cspan Struct Reference	132
7.32.1 Detailed Description	132
7.33 JawsMako::IJawsRenderer::CSpotHalftone Class Reference	133
7.33.1 Detailed Description	133
7.34 JawsMako::CTemporaryStoreParameters Class Reference	133
7.34.1 Detailed Description	133
7.34.2 Constructor & Destructor Documentation	133
7.35 JawsMako::IJawsRenderer::CThresholdArrayHalftone Class Reference	134
7.35.1 Detailed Description	134
7.36 JawsMako::IJawsRenderer::CThresholdHalftone Class Reference	135
7.36.1 Detailed Description	135
7.37 CTransformMatrix< TItem > Class Template Reference	135
7.37.1 Detailed Description	137
7.37.2 Constructor & Destructor Documentation	137
7.37.3 Member Function Documentation	138

7.38 JawsMako::CTransformState Class Reference	148
7.38.1 Detailed Description	149
7.38.2 Member Function Documentation	149
7.38.3 Member Data Documentation	150
7.39 JawsMako::CXFAPacket Class Reference	150
7.39.1 Detailed Description	150
7.40 JawsMako::IAnnotationUtils::CXMLResource Class Reference	150
7.40.1 Detailed Description	151
7.41 IDOMArcSegment::Data Class Reference	151
7.41.1 Detailed Description	151
7.42 IDOMAudioFile::Data Class Reference	151
7.42.1 Detailed Description	152
7.43 IDOMCachedImage::Data Class Reference	152
7.43.1 Detailed Description	152
7.44 IDOMCanvas::Data Class Reference	153
7.44.1 Detailed Description	153
7.45 IDOMCharPathGroup::Data Class Reference	153
7.45.1 Detailed Description	154
7.46 IDOMColorSpaceDeviceN::Data Class Reference	154
7.46.1 Detailed Description	154
7.47 IDOMColorSpaceICCBased::Data Class Reference	155
7.47.1 Detailed Description	155
7.48 IDOMColorSpaceIndexed::Data Class Reference	155
7.48.1 Detailed Description	156
7.49 IDOMColorSpaceLAB::Data Class Reference	156
7.49.1 Detailed Description	156
7.50 IDOMCompositImage::Data Class Reference	156
7.50.1 Detailed Description	157
7.51 IDOMDePremultiplierFilter::Data Class Reference	157
7.51.1 Detailed Description	158
7.52 IDOMDeviceNColorant::Data Class Reference	158
7.52.1 Detailed Description	158
7.53 IDOMExponentialFunction::Data Class Reference	159
7.53.1 Detailed Description	159
7.54 IDOMFilteredImage::Data Class Reference	159
7.54.1 Detailed Description	160
7.55 IDOMFixedPage::Data Class Reference	160
7.55.1 Detailed Description	160
7.56 IDOMFont::Data Class Reference	160
7.56.1 Detailed Description	161
7.57 IDOMFontOpenType::Data Class Reference	161
7.57.1 Detailed Description	162

7.58 IDOMFontOpenTypeTT::Data Class Reference	162
7.58.1 Detailed Description	163
7.59 IDOMFontPCL5::Data Class Reference	163
7.59.1 Detailed Description	164
7.60 IDOMFontPCLXL::Data Class Reference	164
7.60.1 Detailed Description	165
7.61 IDOMFontSource::Data Class Reference	165
7.61.1 Detailed Description	166
7.62 IDOMFontSourceFromStream::Data Class Reference	166
7.62.1 Detailed Description	167
7.63 IDOMFontSourceObfuscationConverter::Data Class Reference	167
7.63.1 Detailed Description	168
7.64 IDOMFontSourceStreamFilter::Data Class Reference	168
7.64.1 Detailed Description	169
7.65 IDOMForm::Data Class Reference	169
7.65.1 Detailed Description	170
7.66 IDOMFormInstance::Data Class Reference	170
7.66.1 Detailed Description	170
7.67 IDOMGlyph::Data Class Reference	171
7.67.1 Detailed Description	171
7.68 IDOMGlyphs::Data Class Reference	171
7.68.1 Detailed Description	172
7.69 IDOMGradientStop::Data Class Reference	172
7.69.1 Detailed Description	172
7.70 IDOMGroup::Data Class Reference	172
7.70.1 Detailed Description	173
7.71 IDOMGroupingFunction::Data Class Reference	173
7.71.1 Detailed Description	173
7.72 IDOMICCProfile::Data Class Reference	174
7.72.1 Detailed Description	174
7.73 IDOMImage::Data Class Reference	174
7.73.1 Detailed Description	175
7.74 IDOMImageBitScalerFilter::Data Class Reference	175
7.74.1 Detailed Description	176
7.75 IDOMImageBleederFilter::Data Class Reference	176
7.75.1 Detailed Description	176
7.76 IDOMImageBrush::Data Class Reference	177
7.76.1 Detailed Description	177
7.77 IDOMImageChannelSelectorFilter::Data Class Reference	177
7.77.1 Detailed Description	178
7.78 IDOMImageColorConverterFilter::Data Class Reference	178
7.78.1 Detailed Description	178

7.79 IDOMImageColorKeyFilter::Data Class Reference	179
7.79.1 Detailed Description	179
7.80 IDOMImageColorSpaceSubstitutionFilter::Data Class Reference	179
7.80.1 Detailed Description	180
7.81 IDOMImageDecodeFilter::Data Class Reference	180
7.81.1 Detailed Description	180
7.82 IDOMImageDeindexerFilter::Data Class Reference	181
7.82.1 Detailed Description	181
7.83 IDOMImageDeviceNToBaseFilter::Data Class Reference	181
7.83.1 Detailed Description	182
7.84 IDOMImageDownsamplerFilter::Data Class Reference	182
7.84.1 Detailed Description	182
7.85 IDOMImageInverterFilter::Data Class Reference	183
7.85.1 Detailed Description	183
7.86 IDOMImageMaskExpanderFilter::Data Class Reference	183
7.86.1 Detailed Description	184
7.87 IDOMImageMatteRemoverFilter::Data Class Reference	184
7.87.1 Detailed Description	184
7.88 IDOMImageSpotMergerFilter::Data Class Reference	185
7.88.1 Detailed Description	185
7.89 IDOMJobTkContent::Data Class Reference	185
7.89.1 Detailed Description	186
7.90 IDOMJobTkGenericCharacterData::Data Class Reference	186
7.90.1 Detailed Description	186
7.91 IDOMJobTkGenericNode::Data Class Reference	187
7.91.1 Detailed Description	187
7.92 IDOMJobTkNode::Data Class Reference	187
7.92.1 Detailed Description	188
7.93 IDOMJobTkValue::Data Class Reference	188
7.93.1 Detailed Description	188
7.94 IDOMLinearGradientBrush::Data Class Reference	189
7.94.1 Detailed Description	189
7.95 IDOMMaskedBrush::Data Class Reference	189
7.95.1 Detailed Description	190
7.96 IDOMMatrix::Data Class Reference	190
7.96.1 Detailed Description	190
7.97 IDOMNullBrush::Data Class Reference	190
7.97.1 Detailed Description	191
7.98 IDOMOutline::Data Class Reference	191
7.98.1 Detailed Description	191
7.99 IDOMOutlineEntry::Data Class Reference	191
7.99.1 Detailed Description	192

7.100 IDOMPathFigure::Data Class Reference	192
7.100.1 Detailed Description	192
7.101 IDOMPathGeometry::Data Class Reference	192
7.101.1 Detailed Description	193
7.102 IDOMPathNode::Data Class Reference	193
7.102.1 Detailed Description	193
7.103 IDOMPCLImage::Data Class Reference	193
7.103.1 Detailed Description	194
7.104 IDOMPDFAlternateImage::Data Class Reference	194
7.104.1 Detailed Description	195
7.105 IDOMPDFImage::Data Class Reference	195
7.105.1 Detailed Description	195
7.106 IDOMPolyBezierSegment::Data Class Reference	196
7.106.1 Detailed Description	196
7.107 IDOMPolyLineSegment::Data Class Reference	196
7.107.1 Detailed Description	197
7.108 IDOMPolyQuadraticBezierSegment::Data Class Reference	197
7.108.1 Detailed Description	197
7.109 IDOMPostScriptCalculatorFunction::Data Class Reference	198
7.109.1 Detailed Description	198
7.110 IDOMPrintTicket::Data Class Reference	198
7.110.1 Detailed Description	199
7.111 IDOMPSDImage::Data Class Reference	199
7.111.1 Detailed Description	199
7.112 IDOMRadialGradientBrush::Data Class Reference	200
7.112.1 Detailed Description	200
7.113 IDOMRawDataFile::Data Class Reference	200
7.113.1 Detailed Description	201
7.114 IDOMRawImage::Data Class Reference	201
7.114.1 Detailed Description	201
7.115 IDOMRecombineAlphaImage::Data Class Reference	202
7.115.1 Detailed Description	202
7.116 IDOMRecombineImage::Data Class Reference	202
7.116.1 Detailed Description	203
7.117 IDOMResourceDictionary::Data Class Reference	203
7.117.1 Detailed Description	204
7.118 IDOMSampledFunction::Data Class Reference	204
7.118.1 Detailed Description	204
7.119 IDOMShadingPatternType1Brush::Data Class Reference	205
7.119.1 Detailed Description	205
7.120 IDOMShadingPatternType2Brush::Data Class Reference	205
7.120.1 Detailed Description	206

7.121 IDOMShadingPatternType3Brush::Data Class Reference	206
7.121.1 Detailed Description	206
7.122 IDOMShadingPatternType4567Brush::Data Class Reference	207
7.122.1 Detailed Description	207
7.123 IDOMShape::Data Class Reference	207
7.123.1 Detailed Description	208
7.124 IDOMSoftMaskBrush::Data Class Reference	208
7.124.1 Detailed Description	208
7.125 IDOMSolidColorBrush::Data Class Reference	208
7.125.1 Detailed Description	209
7.126 IDOMStitchingFunction::Data Class Reference	209
7.126.1 Detailed Description	209
7.127 IDOMTIFFImage::Data Class Reference	210
7.127.1 Detailed Description	210
7.128 IDOMTilingPatternBrush::Data Class Reference	210
7.128.1 Detailed Description	211
7.129 IDOMTransparencyGroup::Data Class Reference	211
7.129.1 Detailed Description	211
7.130 IDOMTrivialImage::Data Class Reference	212
7.130.1 Detailed Description	212
7.131 IDOMType3Font::Data Class Reference	212
7.131.1 Detailed Description	213
7.132 IDOMVisualBrush::Data Class Reference	213
7.132.1 Detailed Description	213
7.133 IEDLNamespace::Data Class Reference	214
7.133.1 Detailed Description	214
7.134 IEDLTempStore::Data Class Reference	214
7.134.1 Detailed Description	215
7.135 IEDLTime::Data Class Reference	215
7.135.1 Detailed Description	215
7.136 IDOMPDFImage::DCTParams Class Reference	215
7.136.1 Detailed Description	216
7.137 EDLIFStream Class Reference	216
7.137.1 Detailed Description	217
7.138 EDLOFStream Class Reference	217
7.138.1 Detailed Description	217
7.139 EDLQName Class Reference	217
7.139.1 Detailed Description	218
7.139.2 Constructor & Destructor Documentation	218
7.139.3 Member Function Documentation	218
7.140 IDOMPDFImage::FlateLZWParams Class Reference	221
7.140.1 Detailed Description	222

7.141 IAbort Class Reference	222
7.141.1 Detailed Description	223
7.142 JawsMako::IAnnotation Class Reference	223
7.142.1 Detailed Description	225
7.142.2 Member Enumeration Documentation	225
7.142.3 Member Function Documentation	226
7.143 JawsMako::IAnnotationAppearance Class Reference	232
7.143.1 Detailed Description	233
7.143.2 Member Function Documentation	233
7.144 JawsMako::IAnnotationReference Class Reference	235
7.144.1 Detailed Description	236
7.144.2 Member Function Documentation	236
7.145 JawsMako::IAnnotationUtils Class Reference	236
7.145.1 Detailed Description	237
7.145.2 Member Function Documentation	237
7.146 JawsMako::IBlendSimplifierTransform Class Reference	238
7.146.1 Detailed Description	239
7.146.2 Member Function Documentation	239
7.147 JawsMako::ICaretAnnotation Class Reference	240
7.147.1 Detailed Description	243
7.147.2 Member Function Documentation	243
7.148 JawsMako::ICFFCIDSplitterTransform Class Reference	243
7.148.1 Detailed Description	245
7.148.2 Member Function Documentation	245
7.149 JawsMako::ICFFSanitizerTransform Class Reference	245
7.149.1 Detailed Description	247
7.149.2 Member Function Documentation	247
7.150 JawsMako::IColorConverterTransform Class Reference	248
7.150.1 Detailed Description	251
7.150.2 Member Function Documentation	251
7.151 IColorManager Class Reference	254
7.151.1 Detailed Description	258
7.151.2 Member Function Documentation	259
7.152 JawsMako::IComplexColorSimplifierTransform Class Reference	274
7.152.1 Detailed Description	276
7.152.2 Member Function Documentation	276
7.153 JawsMako::ICustomTransform Class Reference	277
7.153.1 Detailed Description	279
7.154 IDOMPDFImage::IDecodeParams Class Reference	279
7.154.1 Detailed Description	280
7.155 JawsMako::IDistiller Class Reference	280
7.155.1 Detailed Description	283

7.155.2 Member Enumeration Documentation	284
7.155.3 Member Function Documentation	286
7.156 JawsMako::IDocument Class Reference	305
7.156.1 Detailed Description	307
7.156.2 Member Function Documentation	307
7.157 JawsMako::IDocumentAssembly Class Reference	313
7.157.1 Detailed Description	315
7.157.2 Member Function Documentation	315
7.158 IDOMActionArray Class Reference	318
7.158.1 Detailed Description	320
7.158.2 Member Function Documentation	320
7.159 IDOMActionLaunch Class Reference	322
7.159.1 Detailed Description	323
7.159.2 Member Function Documentation	323
7.160 IDOMArcSegment Class Reference	326
7.160.1 Detailed Description	329
7.160.2 Member Function Documentation	329
7.161 IDOMAudioFile Class Reference	333
7.161.1 Detailed Description	335
7.161.2 Member Function Documentation	335
7.162 IDOMBrush Class Reference	336
7.162.1 Detailed Description	337
7.162.2 Member Function Documentation	337
7.163 IDOMCachedImage Class Reference	339
7.163.1 Detailed Description	341
7.163.2 Member Function Documentation	341
7.164 IDOMCanvas Class Reference	343
7.164.1 Detailed Description	348
7.164.2 Member Function Documentation	349
7.165 IDOMCatalog Class Reference	352
7.165.1 Detailed Description	354
7.165.2 Member Function Documentation	354
7.166 IDOMCharPathGroup Class Reference	358
7.166.1 Detailed Description	363
7.166.2 Member Function Documentation	363
7.167 IDOMColor Class Reference	365
7.167.1 Detailed Description	368
7.167.2 Member Function Documentation	368
7.168 IDOMColorSpace Class Reference	381
7.168.1 Detailed Description	383
7.168.2 Member Function Documentation	383
7.169 IDOMColorSpaceDeviceCMY Class Reference	386

7.169.1 Detailed Description	388
7.169.2 Member Function Documentation	388
7.170 IDOMColorSpaceDeviceCMYK Class Reference	388
7.170.1 Detailed Description	391
7.170.2 Member Function Documentation	391
7.171 IDOMColorSpaceDeviceGray Class Reference	391
7.171.1 Detailed Description	394
7.171.2 Member Function Documentation	394
7.172 IDOMColorSpaceDeviceN Class Reference	394
7.172.1 Detailed Description	397
7.172.2 Member Function Documentation	398
7.173 IDOMColorSpaceDeviceRGB Class Reference	402
7.173.1 Detailed Description	404
7.173.2 Member Function Documentation	404
7.174 IDOMColorSpaceICCBased Class Reference	404
7.174.1 Detailed Description	407
7.174.2 Member Function Documentation	407
7.175 IDOMColorSpaceIndexed Class Reference	409
7.175.1 Detailed Description	411
7.175.2 Member Function Documentation	411
7.176 IDOMColorSpaceLAB Class Reference	413
7.176.1 Detailed Description	415
7.176.2 Member Function Documentation	415
7.177 IDOMColorSpacescRGB Class Reference	417
7.177.1 Detailed Description	419
7.177.2 Member Function Documentation	419
7.178 IDOMColorSpacesGray Class Reference	420
7.178.1 Detailed Description	422
7.178.2 Member Function Documentation	422
7.179 IDOMColorSpacesRGB Class Reference	423
7.179.1 Detailed Description	425
7.179.2 Member Function Documentation	425
7.180 IDOMCompositelImage Class Reference	426
7.180.1 Detailed Description	428
7.180.2 Member Function Documentation	428
7.181 IDOMDePremultiplyFilter Class Reference	430
7.181.1 Detailed Description	431
7.182 IDOMDeviceNColorant Class Reference	431
7.182.1 Detailed Description	432
7.182.2 Member Function Documentation	433
7.183 IDOMExponentialFunction Class Reference	434
7.183.1 Detailed Description	436

7.183.2 Member Function Documentation	436
7.184 IDOMExternalTarget Class Reference	437
7.184.1 Detailed Description	439
7.184.2 Member Function Documentation	439
7.185 IDOMFilteredImage Class Reference	441
7.185.1 Detailed Description	443
7.185.2 Member Function Documentation	443
7.186 IDOMFixedPage Class Reference	446
7.186.1 Detailed Description	450
7.186.2 Member Function Documentation	451
7.187 IDOMFont Class Reference	456
7.187.1 Detailed Description	459
7.187.2 Member Function Documentation	459
7.188 IDOMFontOpenType Class Reference	461
7.188.1 Detailed Description	464
7.188.2 Member Function Documentation	465
7.189 IDOMFontOTFTrueType Class Reference	472
7.189.1 Detailed Description	473
7.190 IDOMFontPCL5 Class Reference	473
7.190.1 Detailed Description	474
7.190.2 Member Function Documentation	474
7.191 IDOMFontPCLXL Class Reference	476
7.191.1 Detailed Description	477
7.191.2 Member Function Documentation	477
7.192 IDOMFontSource Class Reference	479
7.192.1 Detailed Description	480
7.192.2 Member Function Documentation	480
7.193 IDOMFontSourceFromStream Class Reference	481
7.193.1 Detailed Description	483
7.193.2 Member Function Documentation	483
7.194 IDOMFontSourceObfuscationConverter Class Reference	485
7.194.1 Detailed Description	487
7.194.2 Member Function Documentation	487
7.195 IDOMFontSourceStreamFilter Class Reference	489
7.195.1 Detailed Description	491
7.195.2 Member Function Documentation	491
7.196 IDOMForm Class Reference	492
7.196.1 Detailed Description	496
7.196.2 Member Function Documentation	496
7.197 IDOMFormInstance Class Reference	499
7.197.1 Detailed Description	502
7.197.2 Member Function Documentation	503

7.198 IDOMFunction Class Reference	505
7.198.1 Detailed Description	507
7.198.2 Member Function Documentation	507
7.199 IDOMGlyph Class Reference	509
7.199.1 Detailed Description	514
7.199.2 Member Function Documentation	514
7.200 IDOMGlyphIDEnumerator Class Reference	521
7.200.1 Detailed Description	522
7.200.2 Member Function Documentation	522
7.201 IDOMGlyphName Class Reference	524
7.201.1 Detailed Description	525
7.202 IDOMGlyphs Class Reference	525
7.202.1 Detailed Description	532
7.202.2 Member Function Documentation	532
7.203 IDOMGradientBrush Class Reference	550
7.203.1 Detailed Description	552
7.203.2 Member Function Documentation	552
7.204 IDOMGradientStop Class Reference	555
7.204.1 Detailed Description	556
7.204.2 Member Function Documentation	557
7.205 IDOMGroup Class Reference	558
7.205.1 Detailed Description	562
7.205.2 Member Function Documentation	563
7.206 IDOMGroupingFunction Class Reference	565
7.206.1 Detailed Description	568
7.206.2 Member Function Documentation	568
7.207 IDOMHashable Class Reference	569
7.207.1 Detailed Description	570
7.207.2 Member Function Documentation	570
7.208 IDOMICCProfile Class Reference	571
7.208.1 Detailed Description	573
7.208.2 Member Function Documentation	573
7.209 IDOMImage Class Reference	575
7.209.1 Detailed Description	578
7.209.2 Member Function Documentation	578
7.210 IDOMImageBitScalerFilter Class Reference	580
7.210.1 Detailed Description	580
7.210.2 Member Function Documentation	581
7.211 IDOMImageBleederFilter Class Reference	581
7.211.1 Detailed Description	582
7.211.2 Member Function Documentation	582
7.212 IDOMImageBrush Class Reference	583

7.212.1 Detailed Description	586
7.212.2 Member Function Documentation	586
7.213 IDOMImageChannelSelectorFilter Class Reference	594
7.213.1 Detailed Description	594
7.213.2 Member Function Documentation	594
7.214 IDOMImageColorConverterFilter Class Reference	595
7.214.1 Detailed Description	595
7.214.2 Member Function Documentation	596
7.215 IDOMImageColorKeyFilter Class Reference	596
7.215.1 Detailed Description	597
7.215.2 Member Function Documentation	597
7.216 IDOMImageColorSpaceSubstitutionFilter Class Reference	598
7.216.1 Detailed Description	598
7.216.2 Member Function Documentation	598
7.217 IDOMImageDecodeFilter Class Reference	599
7.217.1 Detailed Description	599
7.217.2 Member Function Documentation	600
7.218 IDOMImageDeindexerFilter Class Reference	600
7.218.1 Detailed Description	601
7.218.2 Member Function Documentation	601
7.219 IDOMImageDeviceNToBaseFilter Class Reference	601
7.219.1 Detailed Description	602
7.219.2 Member Function Documentation	602
7.220 IDOMImageDownsamplerFilter Class Reference	603
7.220.1 Detailed Description	603
7.220.2 Member Function Documentation	603
7.221 IDOMImageInverterFilter Class Reference	604
7.221.1 Detailed Description	604
7.221.2 Member Function Documentation	604
7.222 IDOMImageMaskExpanderFilter Class Reference	605
7.222.1 Detailed Description	605
7.222.2 Member Function Documentation	605
7.223 IDOMImageProperties Class Reference	606
7.223.1 Detailed Description	608
7.223.2 Member Function Documentation	608
7.224 IDOMImageSpotMergerFilter Class Reference	611
7.224.1 Detailed Description	612
7.224.2 Member Function Documentation	612
7.225 IDOMInternalTarget Class Reference	614
7.225.1 Detailed Description	616
7.225.2 Member Function Documentation	616
7.226 IDOMJobTk Class Reference	617

7.226.1 Detailed Description	618
7.226.2 Member Function Documentation	618
7.227 IDOMJobTkContent Class Reference	622
7.227.1 Detailed Description	626
7.227.2 Member Function Documentation	626
7.228 IDOMJobTkGenericCharacterData Class Reference	633
7.228.1 Detailed Description	637
7.228.2 Member Function Documentation	637
7.229 IDOMJobTkGenericNode Class Reference	638
7.229.1 Detailed Description	642
7.229.2 Member Function Documentation	642
7.230 IDOMJobTkNode Class Reference	644
7.230.1 Detailed Description	647
7.230.2 Member Function Documentation	648
7.231 IDOMJobTkOwner Class Reference	650
7.231.1 Detailed Description	654
7.231.2 Member Function Documentation	654
7.232 IDOMJobTkValue Class Reference	655
7.232.1 Detailed Description	658
7.232.2 Member Function Documentation	658
7.233 IDOMJPEGImage Class Reference	659
7.233.1 Detailed Description	661
7.233.2 Member Function Documentation	661
7.234 IDOMLinearGradientBrush Class Reference	664
7.234.1 Detailed Description	667
7.234.2 Member Function Documentation	667
7.235 IDOMMaskedBrush Class Reference	670
7.235.1 Detailed Description	674
7.235.2 Member Function Documentation	674
7.236 IDOMMatrix Class Reference	676
7.236.1 Detailed Description	678
7.236.2 Member Function Documentation	678
7.237 IDOMMatteRemoverFilter Class Reference	679
7.237.1 Detailed Description	679
7.238 IDOMMetadata Class Reference	679
7.238.1 Detailed Description	681
7.238.2 Member Function Documentation	682
7.239 IDOMNode Class Reference	684
7.239.1 Detailed Description	687
7.239.2 Member Function Documentation	687
7.240 IDOMNodeFlags Class Reference	697
7.240.1 Detailed Description	698

7.240.2 Member Enumeration Documentation	698
7.240.3 Member Function Documentation	698
7.241 IDOMNullBrush Class Reference	699
7.241.1 Detailed Description	701
7.241.2 Member Function Documentation	701
7.242 IDOMOutline Class Reference	702
7.242.1 Detailed Description	703
7.242.2 Member Function Documentation	703
7.243 IDOMOutlineEntry Class Reference	705
7.243.1 Detailed Description	707
7.243.2 Member Function Documentation	707
7.244 IDOMPageRectTarget Class Reference	712
7.244.1 Detailed Description	714
7.244.2 Member Function Documentation	714
7.245 IDOMPageTarget Class Reference	718
7.245.1 Detailed Description	720
7.245.2 Member Function Documentation	720
7.246 IDOMPathFigure Class Reference	721
7.246.1 Detailed Description	724
7.246.2 Member Function Documentation	724
7.247 IDOMPathGeometry Class Reference	727
7.247.1 Detailed Description	730
7.247.2 Member Function Documentation	730
7.248 IDOMPathGeometryBuilder Class Reference	737
7.248.1 Detailed Description	739
7.248.2 Member Function Documentation	739
7.249 IDOMPathNode Class Reference	742
7.249.1 Detailed Description	750
7.249.2 Member Function Documentation	750
7.250 IDOMPathSegment Class Reference	770
7.250.1 Detailed Description	772
7.250.2 Member Function Documentation	772
7.251 IDOMPCLImage Class Reference	773
7.251.1 Detailed Description	776
7.251.2 Member Function Documentation	776
7.252 IDOMPDFAlternatImage Class Reference	776
7.252.1 Detailed Description	778
7.252.2 Member Function Documentation	778
7.253 IDOMPDFImage Class Reference	780
7.253.1 Detailed Description	782
7.253.2 Member Function Documentation	783
7.254 IDOMPNGImage Class Reference	785

7.254.1 Detailed Description	788
7.254.2 Member Function Documentation	788
7.255 IDOMPolyBezierSegment Class Reference	790
7.255.1 Detailed Description	793
7.255.2 Member Function Documentation	793
7.256 IDOMPolyLineSegment Class Reference	794
7.256.1 Detailed Description	796
7.256.2 Member Function Documentation	796
7.257 IDOMPolyQuadraticBezierSegment Class Reference	798
7.257.1 Detailed Description	800
7.257.2 Member Function Documentation	800
7.258 IDOMPostScriptCalculatorFunction Class Reference	802
7.258.1 Detailed Description	804
7.258.2 Member Function Documentation	804
7.259 IDOMPrintTicket Class Reference	805
7.259.1 Detailed Description	806
7.259.2 Member Function Documentation	807
7.260 IDOMPSDImage Class Reference	807
7.260.1 Detailed Description	809
7.260.2 Member Function Documentation	810
7.261 IDOMPublicKeyPDFSecurityInfo Class Reference	811
7.261.1 Detailed Description	812
7.261.2 Member Enumeration Documentation	812
7.261.3 Member Function Documentation	812
7.262 IDOMRadialGradientBrush Class Reference	813
7.262.1 Detailed Description	817
7.262.2 Member Function Documentation	817
7.263 IDOMRawDataFile Class Reference	820
7.263.1 Detailed Description	822
7.263.2 Member Function Documentation	822
7.264 IDOMRawImage Class Reference	823
7.264.1 Detailed Description	825
7.264.2 Member Function Documentation	825
7.265 IDOMRecombineAlphaImage Class Reference	827
7.265.1 Detailed Description	829
7.265.2 Member Function Documentation	830
7.266 IDOMRecombineImage Class Reference	831
7.266.1 Detailed Description	833
7.266.2 Member Function Documentation	834
7.267 IDOMResource Class Reference	835
7.267.1 Detailed Description	836
7.267.2 Member Function Documentation	837

7.268 IDOMResourceDictionary Class Reference	838
7.268.1 Detailed Description	840
7.268.2 Member Function Documentation	840
7.269 IDOMSampledFunction Class Reference	841
7.269.1 Detailed Description	843
7.269.2 Member Function Documentation	843
7.270 IDOMSecurityInfo Class Reference	845
7.270.1 Detailed Description	846
7.271 IDOMShadingPatternBrush Class Reference	846
7.271.1 Detailed Description	848
7.271.2 Member Function Documentation	848
7.272 IDOMShadingPatternType1Brush Class Reference	852
7.272.1 Detailed Description	855
7.272.2 Member Function Documentation	855
7.273 IDOMShadingPatternType2Brush Class Reference	857
7.273.1 Detailed Description	861
7.273.2 Member Function Documentation	861
7.274 IDOMShadingPatternType3Brush Class Reference	864
7.274.1 Detailed Description	868
7.274.2 Member Function Documentation	868
7.275 IDOMShadingPatternType4567Brush Class Reference	873
7.275.1 Detailed Description	876
7.275.2 Member Function Documentation	877
7.276 IDOMShape Class Reference	883
7.276.1 Detailed Description	886
7.276.2 Member Function Documentation	886
7.277 IDOMSoftMaskBrush Class Reference	890
7.277.1 Detailed Description	892
7.277.2 Member Function Documentation	892
7.278 IDOMSolidColorBrush Class Reference	894
7.278.1 Detailed Description	897
7.278.2 Member Function Documentation	897
7.279 IDOMStandardPDFSecurityInfo Class Reference	902
7.279.1 Detailed Description	903
7.279.2 Member Function Documentation	903
7.280 IDOMStitchingFunction Class Reference	906
7.280.1 Detailed Description	908
7.280.2 Member Function Documentation	908
7.281 IDOMTarget Class Reference	910
7.281.1 Detailed Description	911
7.281.2 Member Function Documentation	911
7.282 IDOMTIFFImage Class Reference	912

7.282.1 Detailed Description	914
7.282.2 Member Enumeration Documentation	914
7.282.3 Member Function Documentation	915
7.283 IDOMTilingPatternBrush Class Reference	918
7.283.1 Detailed Description	920
7.283.2 Member Function Documentation	920
7.284 IDOMTransformableBrush Class Reference	925
7.284.1 Detailed Description	927
7.284.2 Member Function Documentation	927
7.285 IDOMTransparencyGroup Class Reference	927
7.285.1 Detailed Description	932
7.285.2 Member Function Documentation	932
7.286 IDOMTrivialImage Class Reference	938
7.286.1 Detailed Description	941
7.286.2 Member Function Documentation	941
7.287 IDOMType3Font Class Reference	943
7.287.1 Detailed Description	946
7.287.2 Member Function Documentation	946
7.288 IDOMVisualBrush Class Reference	949
7.288.1 Detailed Description	952
7.288.2 Member Function Documentation	952
7.289 IDOMVisualRoot Class Reference	957
7.289.1 Detailed Description	960
7.289.2 Member Function Documentation	961
7.290 IDOMWMPImage Class Reference	961
7.290.1 Detailed Description	963
7.290.2 Member Function Documentation	963
7.291 IEDLClassFactory Class Reference	964
7.291.1 Detailed Description	964
7.291.2 Member Function Documentation	964
7.292 IEDLError Class Reference	967
7.292.1 Detailed Description	967
7.293 IEDLNamespace Class Reference	967
7.293.1 Detailed Description	969
7.293.2 Member Function Documentation	969
7.294 IEDLObject Class Reference	970
7.294.1 Detailed Description	972
7.294.2 Member Function Documentation	972
7.295 IEDLStream Class Reference	973
7.295.1 Detailed Description	975
7.295.2 Member Function Documentation	975
7.296 IEDLTempStore Class Reference	976

7.296.1 Detailed Description	978
7.296.2 Member Function Documentation	978
7.297 IEDLTempStoreObject Class Reference	980
7.297.1 Detailed Description	981
7.297.2 Member Function Documentation	981
7.298 IEDLTime Class Reference	982
7.298.1 Detailed Description	984
7.298.2 Member Function Documentation	984
7.299 IFontHeaderWriteSegmentBlockEnumerator Class Reference	988
7.299.1 Detailed Description	989
7.299.2 Member Function Documentation	989
7.300 IFontPCL5WriteSegmentBlockEnumerator Class Reference	991
7.300.1 Detailed Description	992
7.300.2 Member Function Documentation	992
7.301 JawsMako::IFontProcessorDeferredTransform Class Reference	994
7.301.1 Detailed Description	996
7.301.2 Member Function Documentation	996
7.302 JawsMako::IFontProcessorTransform Class Reference	997
7.302.1 Detailed Description	999
7.302.2 Member Function Documentation	999
7.303 JawsMako::IForm Class Reference	1000
7.303.1 Detailed Description	1002
7.303.2 Member Function Documentation	1003
7.304 JawsMako::IFormDeduplicatorTransform Class Reference	1011
7.304.1 Detailed Description	1013
7.305 JawsMako::IFormField Class Reference	1013
7.305.1 Detailed Description	1015
7.305.2 Member Function Documentation	1016
7.306 JawsMako::IFormUnpackerTransform Class Reference	1024
7.306.1 Detailed Description	1026
7.306.2 Member Function Documentation	1026
7.307 IFrame Class Reference	1026
7.307.1 Detailed Description	1027
7.308 JawsMako::IFreeTextAnnotation Class Reference	1027
7.308.1 Detailed Description	1030
7.308.2 Member Function Documentation	1030
7.309 JawsMako::IHashable Class Reference	1031
7.309.1 Detailed Description	1031
7.309.2 Member Function Documentation	1031
7.310 JawsMako::IJPDSInput Class Reference	1032
7.310.1 Detailed Description	1033
7.310.2 Member Function Documentation	1033

7.311 JawsMako::IJPDSPageRaster Class Reference	1034
7.311.1 Detailed Description	1036
7.312 QImage Decoder Class Reference	1036
7.312.1 Detailed Description	1037
7.312.2 Member Function Documentation	1037
7.313 JawsMako::QImageDownsamplerTransform Class Reference	1038
7.313.1 Detailed Description	1040
7.313.2 Member Function Documentation	1040
7.314 QImage Encoder Class Reference	1042
7.314.1 Detailed Description	1043
7.314.2 Member Function Documentation	1043
7.315 JawsMako::QImageEncoderTransform Class Reference	1043
7.315.1 Detailed Description	1046
7.315.2 Member Enumeration Documentation	1046
7.315.3 Member Function Documentation	1046
7.316 QImageFrame Class Reference	1049
7.316.1 Detailed Description	1051
7.316.2 Member Function Documentation	1051
7.317 QImageFrameWriter Class Reference	1054
7.317.1 Detailed Description	1056
7.317.2 Member Function Documentation	1056
7.318 JawsMako::QImageMergerTransform Class Reference	1058
7.318.1 Detailed Description	1060
7.318.2 Member Function Documentation	1060
7.319 JawsMako::ICustomTransform::Implementation Class Reference	1061
7.319.1 Detailed Description	1063
7.319.2 Member Function Documentation	1063
7.320 JawsMako::IInkAnnotation Class Reference	1077
7.320.1 Detailed Description	1080
7.320.2 Member Function Documentation	1080
7.321 JawsMako::IInput Class Reference	1081
7.321.1 Detailed Description	1082
7.321.2 Member Function Documentation	1082
7.322 IInputEnum< typename T > Class Reference	1085
7.322.1 Detailed Description	1085
7.323 IInputEnumRC< typename T > Class Reference	1085
7.323.1 Detailed Description	1085
7.324 IInputPushbackStream Class Reference	1086
7.324.1 Detailed Description	1089
7.325 IInputStream Class Reference	1090
7.325.1 Detailed Description	1093
7.325.2 Member Function Documentation	1093

7.326 JawsMako::IJawsMako Class Reference	1104
7.326.1 Detailed Description	1107
7.326.2 Member Function Documentation	1107
7.327 JawsMako::IJawsRenderer Class Reference	1108
7.327.1 Detailed Description	1111
7.327.2 Member Function Documentation	1111
7.328 JawsMako::ILayout Class Reference	1123
7.328.1 Detailed Description	1124
7.328.2 Member Function Documentation	1124
7.329 JawsMako::ILayoutFont Class Reference	1126
7.329.1 Detailed Description	1128
7.329.2 Member Enumeration Documentation	1128
7.329.3 Member Function Documentation	1129
7.330 JawsMako::ILayoutFontWeight Class Reference	1130
7.330.1 Detailed Description	1131
7.330.2 Member Enumeration Documentation	1131
7.330.3 Member Function Documentation	1132
7.331 JawsMako::ILayoutImageRun Class Reference	1134
7.331.1 Detailed Description	1135
7.331.2 Member Function Documentation	1135
7.332 JawsMako::ILayoutParagraph Class Reference	1136
7.332.1 Detailed Description	1138
7.332.2 Member Enumeration Documentation	1138
7.332.3 Member Function Documentation	1138
7.333 JawsMako::ILayoutRun Class Reference	1143
7.333.1 Detailed Description	1144
7.334 JawsMako::ILayoutTextRun Class Reference	1145
7.334.1 Detailed Description	1146
7.334.2 Member Function Documentation	1146
7.335 JawsMako::ILineAnnotation Class Reference	1150
7.335.1 Detailed Description	1153
7.335.2 Member Function Documentation	1153
7.336 JawsMako::ILinkAnnotation Class Reference	1155
7.336.1 Detailed Description	1158
7.336.2 Member Function Documentation	1158
7.337 JawsMako::IMarkedContentArtifactDetails Class Reference	1160
7.337.1 Detailed Description	1162
7.338 JawsMako::IMarkedContentDetails Class Reference	1162
7.338.1 Detailed Description	1163
7.339 JawsMako::IMarkedContentStructureDetails Class Reference	1164
7.339.1 Detailed Description	1165
7.340 JawsMako::IMarkupAnnotation Class Reference	1165

7.340.1 Detailed Description	1168
7.340.2 Member Function Documentation	1168
7.341 JawsMako::IMediaHandler Class Reference	1171
7.341.1 Detailed Description	1171
7.341.2 Member Function Documentation	1171
7.342 JawsMako::INamedDestination Class Reference	1171
7.342.1 Detailed Description	1173
7.342.2 Member Function Documentation	1173
7.343 JawsMako::IOptionalContent Class Reference	1174
7.343.1 Detailed Description	1176
7.343.2 Member Function Documentation	1176
7.344 JawsMako::IOptionalContentConfiguration Class Reference	1178
7.344.1 Detailed Description	1180
7.345 JawsMako::IOptionalContentDetails Class Reference	1181
7.345.1 Detailed Description	1182
7.345.2 Member Enumeration Documentation	1182
7.345.3 Member Function Documentation	1183
7.346 JawsMako::IOptionalContentFixerTransform Class Reference	1183
7.346.1 Detailed Description	1185
7.346.2 Member Function Documentation	1185
7.347 JawsMako::IOptionalContentGroup Class Reference	1185
7.347.1 Detailed Description	1187
7.348 JawsMako::IOptionalContentGroupReference Class Reference	1187
7.348.1 Detailed Description	1188
7.349 JawsMako::IOptionalContentGroupUsage Class Reference	1188
7.349.1 Detailed Description	1190
7.349.2 Member Function Documentation	1190
7.350 JawsMako::IOptionalContentGroupUsageApplication Class Reference	1192
7.350.1 Detailed Description	1193
7.351 JawsMako::IOptionalContentVisibilityExpression Class Reference	1193
7.351.1 Detailed Description	1195
7.351.2 Member Enumeration Documentation	1195
7.351.3 Member Function Documentation	1195
7.352 JawsMako::IOutput Class Reference	1196
7.352.1 Detailed Description	1197
7.352.2 Member Function Documentation	1197
7.353 JawsMako::IOutputIntent Class Reference	1200
7.353.1 Detailed Description	1202
7.353.2 Member Function Documentation	1202
7.354 IOutputStream Class Reference	1203
7.354.1 Detailed Description	1205
7.354.2 Member Function Documentation	1205

7.355 JawsMako::IOutputWriter Class Reference	1210
7.355.1 Detailed Description	1211
7.355.2 Member Function Documentation	1211
7.356 JawsMako::IOverprintSimulationTransform Class Reference	1212
7.356.1 Detailed Description	1214
7.356.2 Member Function Documentation	1214
7.357 JawsMako::IOXPSInput Class Reference	1215
7.357.1 Detailed Description	1217
7.357.2 Member Function Documentation	1217
7.358 JawsMako::IOXPSOutput Class Reference	1217
7.358.1 Detailed Description	1220
7.358.2 Member Function Documentation	1220
7.359 JawsMako::IPage Class Reference	1221
7.359.1 Detailed Description	1223
7.359.2 Member Function Documentation	1224
7.360 JawsMako::IPageCropperTransform Class Reference	1228
7.360.1 Detailed Description	1230
7.360.2 Member Function Documentation	1230
7.361 JawsMako::IPageLabel Class Reference	1230
7.361.1 Detailed Description	1231
7.361.2 Member Enumeration Documentation	1232
7.361.3 Member Function Documentation	1233
7.362 JawsMako::IPageLayout Class Reference	1235
7.362.1 Detailed Description	1236
7.362.2 Member Function Documentation	1236
7.363 JawsMako::IPageLayoutData Class Reference	1238
7.363.1 Detailed Description	1239
7.363.2 Member Function Documentation	1239
7.364 JawsMako::IPageLayoutNode Class Reference	1240
7.364.1 Detailed Description	1241
7.364.2 Member Function Documentation	1241
7.365 JawsMako::IPageRaster Class Reference	1241
7.365.1 Detailed Description	1243
7.365.2 Member Function Documentation	1243
7.366 JawsMako::IPatternConverterTransform Class Reference	1244
7.366.1 Detailed Description	1245
7.366.2 Member Function Documentation	1245
7.367 JawsMako::IPCL5Input Class Reference	1246
7.367.1 Detailed Description	1248
7.367.2 Member Function Documentation	1248
7.368 JawsMako::IPCL5Output Class Reference	1253
7.368.1 Detailed Description	1255

7.368.2 Member Function Documentation	1255
7.369 JawsMako::IPCLXAttributeHandler Class Reference	1257
7.369.1 Detailed Description	1258
7.369.2 Member Function Documentation	1258
7.370 JawsMako::IPCLXInput Class Reference	1258
7.370.1 Detailed Description	1260
7.370.2 Member Function Documentation	1260
7.371 JawsMako::IPCLXOutput Class Reference	1265
7.371.1 Detailed Description	1266
7.371.2 Member Function Documentation	1267
7.372 JawsMako::IPDFArray Class Reference	1268
7.372.1 Detailed Description	1270
7.372.2 Member Function Documentation	1270
7.373 JawsMako::IPDFBoolean Class Reference	1276
7.373.1 Detailed Description	1277
7.373.2 Member Function Documentation	1277
7.374 JawsMako::IPDFDictionary Class Reference	1278
7.374.1 Detailed Description	1281
7.374.2 Member Function Documentation	1281
7.375 JawsMako::IPDFFarReference Class Reference	1286
7.375.1 Detailed Description	1288
7.375.2 Member Function Documentation	1288
7.376 JawsMako::IPDFInput Class Reference	1289
7.376.1 Detailed Description	1292
7.376.2 Member Function Documentation	1292
7.377 JawsMako::IPDFInteger Class Reference	1300
7.377.1 Detailed Description	1302
7.377.2 Member Function Documentation	1302
7.378 JawsMako::IPDFNull Class Reference	1302
7.378.1 Detailed Description	1304
7.378.2 Member Function Documentation	1304
7.379 JawsMako::IPDFObject Class Reference	1305
7.379.1 Detailed Description	1306
7.379.2 Member Function Documentation	1307
7.380 JawsMako::IPDFObjectStore Class Reference	1309
7.380.1 Detailed Description	1311
7.380.2 Member Function Documentation	1311
7.381 JawsMako::IPDFOutput Class Reference	1313
7.381.1 Detailed Description	1319
7.381.2 Member Enumeration Documentation	1319
7.381.3 Member Function Documentation	1322
7.382 JawsMako::IPDFPageExtractor Class Reference	1342

7.382.1 Detailed Description	1343
7.382.2 Member Function Documentation	1343
7.383 JawsMako::IPDFPageInserter Class Reference	1346
7.383.1 Detailed Description	1347
7.383.2 Member Function Documentation	1347
7.384 JawsMako::IPDFReal Class Reference	1350
7.384.1 Detailed Description	1351
7.384.2 Member Function Documentation	1351
7.385 JawsMako::IPDFReference Class Reference	1352
7.385.1 Detailed Description	1354
7.385.2 Member Function Documentation	1354
7.386 JawsMako::IPDFString Class Reference	1356
7.386.1 Detailed Description	1358
7.386.2 Member Function Documentation	1358
7.387 JawsMako::IPDFValidator Class Reference	1360
7.387.1 Detailed Description	1361
7.387.2 Member Function Documentation	1361
7.388 JawsMako::IPJLParser Class Reference	1362
7.388.1 Detailed Description	1363
7.388.2 Member Enumeration Documentation	1364
7.388.3 Member Function Documentation	1364
7.389 JawsMako::IPolyAnnotation Class Reference	1365
7.389.1 Detailed Description	1368
7.389.2 Member Function Documentation	1368
7.390 JawsMako::IPopupAnnotation Class Reference	1369
7.390.1 Detailed Description	1371
7.390.2 Member Function Documentation	1371
7.391 JawsMako::IPPMLInput Class Reference	1372
7.391.1 Detailed Description	1374
7.391.2 Member Function Documentation	1374
7.392 JawsMako::IPRNInput Class Reference	1374
7.392.1 Detailed Description	1376
7.392.2 Member Function Documentation	1376
7.393 JawsMako::IProgressMonitor Class Reference	1377
7.393.1 Detailed Description	1378
7.393.2 Member Function Documentation	1378
7.394 JawsMako::IProgressTick Class Reference	1380
7.394.1 Detailed Description	1381
7.394.2 Member Function Documentation	1381
7.395 JawsMako::IPSCCommentMonitor Class Reference	1383
7.395.1 Detailed Description	1383
7.395.2 Member Function Documentation	1383

7.396 JawsMako::IPSInjector Class Reference	1384
7.396.1 Detailed Description	1385
7.396.2 Member Function Documentation	1385
7.397 JawsMako::IPSInput Class Reference	1387
7.397.1 Detailed Description	1389
7.397.2 Member Function Documentation	1389
7.398 JawsMako::IPSOOutput Class Reference	1390
7.398.1 Detailed Description	1393
7.398.2 Member Function Documentation	1393
7.399 IPushbackStream Class Reference	1397
7.399.1 Detailed Description	1398
7.399.2 Member Function Documentation	1398
7.400 IRAInputPushbackStream Class Reference	1399
7.400.1 Detailed Description	1403
7.401 IRAInputStream Class Reference	1403
7.401.1 Detailed Description	1407
7.402 IRAOutputStream Class Reference	1408
7.402.1 Detailed Description	1410
7.403 IRAStream Class Reference	1410
7.403.1 Detailed Description	1411
7.403.2 Member Function Documentation	1411
7.404 IRCObject Class Reference	1412
7.404.1 Detailed Description	1414
7.404.2 Member Function Documentation	1414
7.405 JawsMako::IRedactionAnnotation Class Reference	1415
7.405.1 Detailed Description	1418
7.405.2 Member Function Documentation	1418
7.406 JawsMako::IRedactorTransform Class Reference	1419
7.406.1 Detailed Description	1421
7.406.2 Member Function Documentation	1421
7.407 JawsMako::IRendererTransform Class Reference	1422
7.407.1 Detailed Description	1427
7.407.2 Member Function Documentation	1427
7.408 IRunnable Class Reference	1437
7.408.1 Detailed Description	1438
7.408.2 Member Function Documentation	1438
7.409 JawsMako::ISeparator Class Reference	1438
7.409.1 Detailed Description	1439
7.409.2 Member Function Documentation	1439
7.410 ISession Class Reference	1442
7.410.1 Detailed Description	1443
7.410.2 Member Function Documentation	1443

7.411 JawsMako::IShapeAnnotation Class Reference	1446
7.411.1 Detailed Description	1449
7.411.2 Member Function Documentation	1449
7.412 JawsMako::ISkiaRenderer Class Reference	1451
7.412.1 Detailed Description	1452
7.412.2 Member Function Documentation	1452
7.413 JawsMako::ISoftMaskConverterTransform Class Reference	1453
7.413.1 Detailed Description	1454
7.414 JawsMako::ISoundAnnotation Class Reference	1454
7.414.1 Detailed Description	1457
7.414.2 Member Function Documentation	1457
7.415 JawsMako::IStampAnnotation Class Reference	1458
7.415.1 Detailed Description	1461
7.415.2 Member Function Documentation	1461
7.416 JawsMako::IStrokerTransform Class Reference	1462
7.416.1 Detailed Description	1465
7.416.2 Member Function Documentation	1465
7.417 JawsMako::IStructure Class Reference	1466
7.417.1 Detailed Description	1467
7.417.2 Member Function Documentation	1468
7.418 JawsMako::IStructureElement Class Reference	1468
7.418.1 Detailed Description	1470
7.418.2 Member Function Documentation	1471
7.419 JawsMako::IStructureElementChild Class Reference	1472
7.419.1 Detailed Description	1473
7.420 JawsMako::IStructureElementReference Class Reference	1473
7.420.1 Detailed Description	1474
7.421 JawsMako::IStructureElementReferenceChild Class Reference	1474
7.421.1 Detailed Description	1475
7.422 JawsMako::IStructureMarkedContentReferenceChild Class Reference	1475
7.422.1 Detailed Description	1477
7.423 JawsMako::IStructureObjectReferenceChild Class Reference	1477
7.423.1 Detailed Description	1478
7.424 JawsMako::ISVGGenerator Class Reference	1478
7.424.1 Detailed Description	1480
7.424.2 Member Function Documentation	1480
7.425 Iterator Class Reference	1483
7.425.1 Detailed Description	1483
7.426 JawsMako::ITextAnnotation Class Reference	1484
7.426.1 Detailed Description	1487
7.426.2 Member Function Documentation	1487
7.427 JawsMako::ITextMarkupAnnotation Class Reference	1488

7.427.1 Detailed Description	1491
7.427.2 Member Function Documentation	1491
7.428 JawsMako::ITextRun Class Reference	1492
7.428.1 Detailed Description	1493
7.428.2 Member Function Documentation	1493
7.429 JawsMako::ITextSearch Class Reference	1494
7.429.1 Detailed Description	1495
7.429.2 Member Function Documentation	1495
7.430 JawsMako::ITextSelect Class Reference	1496
7.430.1 Detailed Description	1497
7.430.2 Member Function Documentation	1497
7.431 JawsMako::IThreads Class Reference	1497
7.431.1 Detailed Description	1498
7.432 JawsMako::ITransform Class Reference	1499
7.432.1 Detailed Description	1500
7.432.2 Member Function Documentation	1501
7.433 JawsMako::ITransformChain Class Reference	1504
7.433.1 Detailed Description	1506
7.433.2 Member Function Documentation	1506
7.434 JawsMako::IType3UnpackerTransform Class Reference	1508
7.434.1 Detailed Description	1510
7.434.2 Member Function Documentation	1510
7.435 JawsMako::IUnicodeHelper Class Reference	1511
7.435.1 Detailed Description	1512
7.436 JawsMako::IWidgetAnnotation Class Reference	1512
7.436.1 Detailed Description	1516
7.436.2 Member Function Documentation	1517
7.437 JawsMako::IWidgetAppearanceCharacteristics Class Reference	1528
7.437.1 Detailed Description	1530
7.437.2 Member Function Documentation	1530
7.438 JawsMako::IXAMLGenerator Class Reference	1534
7.438.1 Detailed Description	1536
7.438.2 Member Function Documentation	1536
7.439 JawsMako::IXPSInput Class Reference	1543
7.439.1 Detailed Description	1545
7.439.2 Member Function Documentation	1545
7.440 JawsMako::IXPSOutput Class Reference	1546
7.440.1 Detailed Description	1549
7.440.2 Member Function Documentation	1549
7.441 IDOMPDFImage::JBIG2Params Class Reference	1554
7.441.1 Detailed Description	1555
7.442 PointImpl< PointType > Class Template Reference	1555

7.442.1 Detailed Description	1555
7.443 PValue Class Reference	1555
7.443.1 Detailed Description	1556
7.444 SignatureID Class Reference	1556
7.444.1 Detailed Description	1556
8 File Documentation	1558
8.1 edblend.h File Reference	1558
8.2 edlmath.h File Reference	1558
8.3 edlnamespaces.h File Reference	1558
8.4 edlproperty.h File Reference	1558
8.5 edlqname.h File Reference	1558
8.6 edlquartz.h File Reference	1558
8.7 edlstring.h File Reference	1558
8.8 edltime.h File Reference	1558
8.9 edltypes.h File Reference	1558
8.10 edlvector.h File Reference	1558
8.11 edlversion.h File Reference	1558
8.12 idombrush.h File Reference	1558
8.13 idomfont.h File Reference	1558
8.14 idomfontpcl.h File Reference	1558
8.15 idomform.h File Reference	1558
8.16 idomnode.h File Reference	1558
8.17 idomresources.h File Reference	1558
8.18 iedcollection.h File Reference	1558
8.19 iedlenum.h File Reference	1558
8.20 iedltree.h File Reference	1558
8.21 ifilespec.h File Reference	1558
8.22 iimagecodec.h File Reference	1558
8.23 ircobject.h File Reference	1558
8.24 memutils.h File Reference	1558
8.25 objclassid.h File Reference	1558
8.26 platform.h File Reference	1558
8.27 smartptr.h File Reference	1558
8.28 customtransform.h File Reference	1558
8.29 distiller.h File Reference	1558
8.30 ijpdinput.h File Reference	1558
8.31 interactive.h File Reference	1558
8.32 jawsmako.h File Reference	1558
8.33 optionalcontent.h File Reference	1558
8.34 pcl5input.h File Reference	1558
8.35 pcl5output.h File Reference	1558

- 8.36 pclxlinput.h File Reference 1558
- 8.37 pclxloutput.h File Reference 1558
- 8.38 pdfinput.h File Reference 1558
- 8.39 pdfobjects.h File Reference 1558
- 8.40 pdfoutput.h File Reference 1558
- 8.41 pdfpage.h File Reference 1558
- 8.42 ppminput.h File Reference 1558
- 8.43 prninput.h File Reference 1558
- 8.44 psinput.h File Reference 1558
- 8.45 separator.h File Reference 1558
- 8.46 structure.h File Reference 1558
- 8.47 text.h File Reference 1558
- 8.48 transforms.h File Reference 1558
- 8.49 types.h File Reference 1558
- 8.50 xpsoutput.h File Reference 1558

Index 1559

1 Version 7.4.0 API Documentation

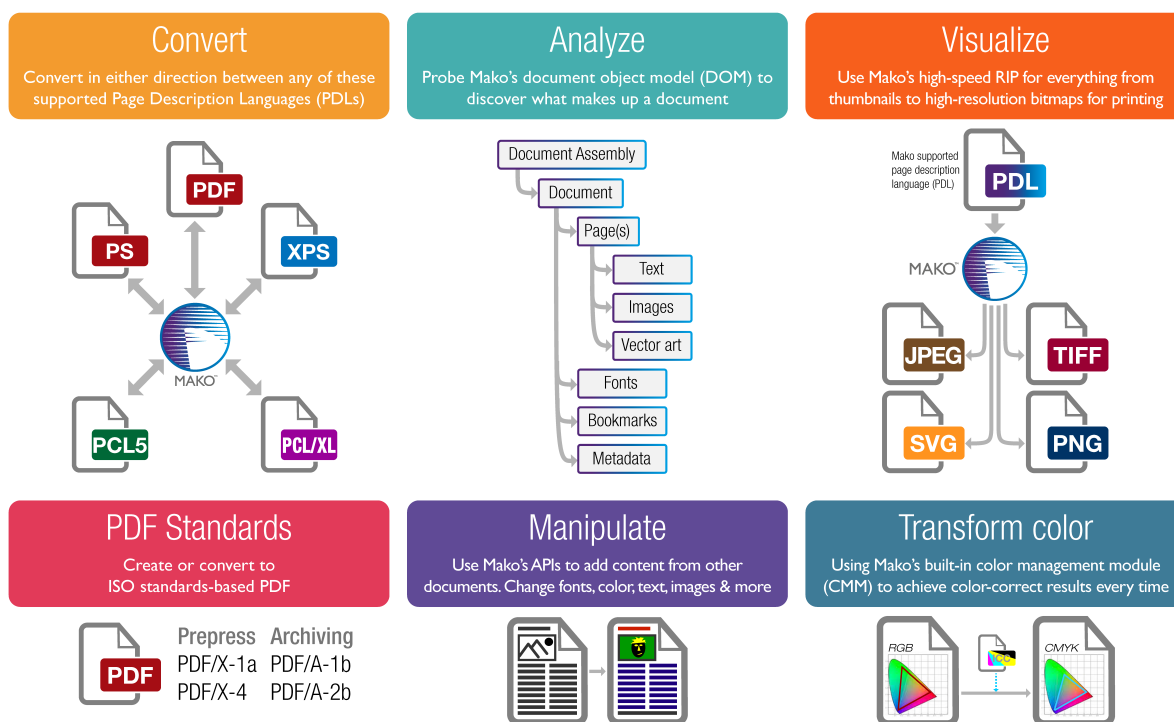


Figure 1 Mako functions

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
- 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, macOS & Linux).
 - Simple Examples (`./makoapps/simpleexamples`). A sample application demonstrating numerous uses of the Mako APIs (Windows, macOS & Linux).
 - iOS XPS Viewer (`./makoapps/iOSXPSViewer`) for iOS that provides a simple XPS viewer using the Mako APIs.

- MakoDistiller (`./makoapps/makodistiller`). A sample application for converting PostScript to PDF that offers numerous options to control how the PDF is generated. The command-line options are documented in detail on [this documentation page](#). (Windows, macOS & Linux).
 - SVG Generator (`./makoapps/svggenerator`). A sample application to generate an SVG from Mako, showing exactly how it's done. (Windows, macOS & Linux)
 - 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. (Windows, Visual Studio 2017 only).
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:

BoxTpl< PointType >	106
BoxTpl< double >	106
JawsMako::CAnnotationBorder	106
JawsMako::IXAMLGenerator::CAnnotationXAML	108
CClassID	111
CClassParams	113
IDOMArcSegment::Data	151
IDOMAudioFile::Data	151
IDOMCanvas::Data	153
IDOMCharPathGroup::Data	153
IDOMColorSpaceDeviceN::Data	154
IDOMColorSpaceICCBased::Data	155
IDOMColorSpaceIndexed::Data	155
IDOMColorSpaceLAB::Data	156
IDOMDePremultiplierFilter::Data	157
IDOMDeviceNColorant::Data	158
IDOMExponentialFunction::Data	159
IDOMFixedPage::Data	160

IDOMFont::Data	160
IDOMFontOpenType::Data	161
IDOMFontOpenTypeTT::Data	162
IDOMFontPCL5::Data	163
IDOMFontPCLXL::Data	164
IDOMFontSource::Data	165
IDOMFontSourceObfuscationConverter::Data	167
IDOMFontSourceStreamFilter::Data	168
IDOMFontSourceFromStream::Data	166
IDOMForm::Data	169
IDOMFormInstance::Data	170
IDOMGlyph::Data	171
IDOMGlyphs::Data	171
IDOMGradientStop::Data	172
IDOMGroup::Data	172
IDOMGroupingFunction::Data	173
IDOMICCProfile::Data	174
IDOMImage::Data	174
IDOMCachedImage::Data	152
IDOMCompositelImage::Data	156
IDOMFilteredImage::Data	159
IDOMPCLImage::Data	193
IDOMPDFImage::Data	195
IDOMPSDImage::Data	199
IDOMRawImage::Data	201
IDOMRecombineAlphalImage::Data	202
IDOMRecombineImage::Data	202
IDOMTIFFImage::Data	210
IDOMImageBitScalerFilter::Data	175
IDOMImageBleederFilter::Data	176
IDOMImageBrush::Data	177
IDOMImageChannelSelectorFilter::Data	177

IDOMImageColorConverterFilter::Data	178
IDOMImageColorKeyFilter::Data	179
IDOMImageColorSpaceSubstitutionFilter::Data	179
IDOMImageDecodeFilter::Data	180
IDOMImageDeindexerFilter::Data	181
IDOMImageDeviceNToBaseFilter::Data	181
IDOMImageDownsamplerFilter::Data	182
IDOMImageInverterFilter::Data	183
IDOMImageMaskExpanderFilter::Data	183
IDOMImageMatteRemoverFilter::Data	184
IDOMImageSpotMergerFilter::Data	185
IDOMJobTkContent::Data	185
IDOMJobTkGenericCharacterData::Data	186
IDOMJobTkGenericNode::Data	187
IDOMJobTkNode::Data	187
IDOMJobTkValue::Data	188
IDOMLinearGradientBrush::Data	189
IDOMMaskedBrush::Data	189
IDOMMatrix::Data	190
IDOMNullBrush::Data	190
IDOMOutline::Data	191
IDOMOutlineEntry::Data	191
IDOMPDFAlternateImage::Data	194
IDOMPathFigure::Data	192
IDOMPathGeometry::Data	192
IDOMPathNode::Data	193
IDOMPolyBezierSegment::Data	196
IDOMPolyLineSegment::Data	196
IDOMPolyQuadraticBezierSegment::Data	197
IDOMPostScriptCalculatorFunction::Data	198
IDOMPrintTicket::Data	198
IDOMRadialGradientBrush::Data	200

IDOMRawDataFile::Data	200
IDOMResourceDictionary::Data	203
IDOMSampledFunction::Data	204
IDOMShadingPatternType1Brush::Data	205
IDOMShadingPatternType2Brush::Data	205
IDOMShadingPatternType3Brush::Data	206
IDOMShadingPatternType4567Brush::Data	207
IDOMShape::Data	207
IDOMSoftMaskBrush::Data	208
IDOMSolidColorBrush::Data	208
IDOMStitchingFunction::Data	209
IDOMTilingPatternBrush::Data	210
IDOMTransparencyGroup::Data	211
IDOMTrivialImage::Data	212
IDOMType3Font::Data	212
IDOMVisualBrush::Data	213
IEDLNamespace::Data	214
IEDLTempStore::Data	214
IEDLTime::Data	215
JawsMako::IJawsRenderer::CColorSpotHalftone	115
JawsMako::IJawsRenderer::CColorThresholdArrayHalftone	115
JawsMako::IPDFValidator::CContentError	116
JawsMako::IJawsRenderer::CEDSHalftone	116
JawsMako::CFieldOption	117
JawsMako::IJawsRenderer::CFrameBufferInfo	118
JawsMako::IPDFValidator::CGeneralError	118
CGlyphsCluster	119
CGlyphsClusters	120
CIndicesGlyph	120
JawsMako::ILayoutFont::CLayoutFontItem	122
IDOMShadingPatternType4567Brush::CMeshEntry	122
JawsMako::IPDFValidator::CPageErrors	124

JawsMako::CPDFFarReference	124
JawsMako::IPDFInput::CPdfFontInfo	126
JawsMako::CPDFReference	127
JawsMako::IPDFInput::CPdfScannedInk	129
JawsMako::IPJLParser::CPjIAttributeValue	129
JawsMako::CQuadPoint	130
JawsMako::CRectInset	130
JawsMako::ISVGGenerator::CResourceEntry	131
JawsMako::IXAMLGenerator::CResourceEntry	131
IDOMShape::CShapeDetails	131
IDOMShape::CShapeDetails::Cspan	132
JawsMako::IJawsRenderer::CSpotHalftone	133
JawsMako::CTemporaryStoreParameters	133
JawsMako::IJawsRenderer::CThresholdArrayHalftone	134
JawsMako::IJawsRenderer::CThresholdHalftone	135
CTransformMatrix< TItem >	135
CTransformMatrix< double >	135
JawsMako::CTransformState	148
JawsMako::CXFAPacket	150
JawsMako::IAnnotationUtils::CXMLResource	150
EDLIFStream	216
EDLOFStream	217
EDLQName	217
JawsMako::IAnnotationUtils	236
IDOMDePremultiplyFilter	430
IDOMFontOTFTrueType	472
IDOMHashable	569
IDOMColor	365
IDOMColorSpace	381
IDOMColorSpaceDeviceCMY	386
IDOMColorSpaceDeviceCMYK	388
IDOMColorSpaceDeviceGray	391

IDOMColorSpaceDeviceN	394
IDOMColorSpaceDeviceRGB	402
IDOMColorSpaceICCBased	404
IDOMColorSpaceIndexed	409
IDOMColorSpaceLAB	413
IDOMColorSpacesGray	420
IDOMColorSpacesRGB	423
IDOMColorSpacescRGB	417
IDOMDeviceNColorant	431
IDOMFontSource	479
IDOMFont	456
IDOMFontOpenType	461
IDOMType3Font	943
IDOMFontSourceFromStream	481
IDOMFontSourceStreamFilter	489
IDOMFontSourceObfuscationConverter	485
IDOMFunction	505
IDOMExponentialFunction	434
IDOMGroupingFunction	565
IDOMPostScriptCalculatorFunction	802
IDOMSampledFunction	841
IDOMStitchingFunction	906
IDOMICCProfile	571
IDOMImage	575
IDOMCachedImage	339
IDOMCompositelImage	426
IDOMFilteredImage	441
IDOMJPEGImage	659
IDOMPCLImage	773
IDOMPDFImage	780
IDOMPNGImage	785
IDOMPSDImage	807

IDOMRawImage	823
IDOMRecombineAlphaImage	827
IDOMRecombineImage	831
IDOMTIFFImage	912
IDOMTrivialImage	938
IDOMWMPImage	961
IDOMImageBrush	583
IDOMMaskedBrush	670
IDOMPDFAlternateImage	776
IDOMPathFigure	721
IDOMPathGeometry	727
IDOMPathSegment	770
IDOMArcSegment	326
IDOMPolyBezierSegment	790
IDOMPolyLineSegment	794
IDOMPolyQuadraticBezierSegment	798
IDOMShadingPatternType1Brush	852
IDOMShadingPatternType2Brush	857
IDOMShadingPatternType3Brush	864
IDOMShadingPatternType4567Brush	873
IDOMSolidColorBrush	894
InputStream	1090
InputPushbackStream	1086
IRAInputStream	1403
IRAInputPushbackStream	1399
IDOMMatteRemoverFilter	679
IDOMNodeFlags	697
IEDLClassFactory	964
JawsMako::IJawsMako	1104
IEDLError	967
IFrame	1026
JawsMako::IHashable	1031

JawsMako::IOptionalContentDetails	1181
JawsMako::IOptionalContentGroupReference	1187
JawsMako::IOptionalContentVisibilityExpression	1193
JawsMako::IPDFObject	1305
JawsMako::IPDFArray	1268
JawsMako::IPDFBoolean	1276
JawsMako::IPDFDictionary	1278
JawsMako::IPDFFarReference	1286
JawsMako::IPDFInteger	1300
JawsMako::IPDFNull	1302
JawsMako::IPDFReal	1350
JawsMako::IPDFReference	1352
JawsMako::IPDFString	1356
JawsMako::IPageRaster	1241
JawsMako::IJPDSPageRaster	1034
JawsMako::ICustomTransform::Implementation	1061
IInputEnum< typename T >	1085
IInputEnumRC< typename T >	1085
JawsMako::IMediaHandler	1171
JawsMako::IPCLXLAttributeHandler	1257
JawsMako::IPSCCommentMonitor	1383
JawsMako::IPSInjector	1384
IPushbackStream	1397
IInputPushbackStream	1086
IRAInputPushbackStream	1399
IRASStream	1410
IRAInputStream	1403
IRAOutputStream	1408
IRCOBJECT	1412
IAbort	222
IDOMFontOpenType::CCIDMap	108
IDOMGlyphName	524

IDOMPDFImage::IDecodeParams	279
IDOMPDFImage::CCITTFaxParams	110
IDOMPDFImage::DCTParams	215
IDOMPDFImage::FlateLZWParams	221
IDOMPDFImage::JBIG2Params	1554
IEDLObject	970
IColorManager	254
IDOMBrush	336
IDOMNullBrush	699
IDOMSolidColorBrush	894
IDOMTransformableBrush	925
IDOMGradientBrush	550
IDOMLinearGradientBrush	664
IDOMRadialGradientBrush	813
IDOMImageBrush	583
IDOMShadingPatternBrush	846
IDOMShadingPatternType1Brush	852
IDOMShadingPatternType2Brush	857
IDOMShadingPatternType3Brush	864
IDOMShadingPatternType4567Brush	873
IDOMSoftMaskBrush	890
IDOMTilingPatternBrush	918
IDOMVisualBrush	949
IDOMCatalog	352
IDOMColor	365
IDOMColorSpace	381
IDOMDeviceNColorant	431
IDOMFontSource	479
IDOMFunction	505
IDOMGlyphIDEnumerator	521
IDOMGradientStop	555
IDOMImageProperties	606

IDOMJobTk	617
IDOMMatrix	676
IDOMMetadata	679
IDOMNode	684
IDOMForm	492
IDOMFormInstance	499
IDOMGlyph	509
IDOMGlyphs	525
IDOMGroup	558
IDOMCharPathGroup	358
IDOMTransparencyGroup	927
IDOMCanvas	343
IDOMJobTkContent	622
IDOMJobTkGenericCharacterData	633
IDOMJobTkGenericNode	638
IDOMJobTkNode	644
IDOMJobTkOwner	650
IDOMFixedPage	446
IDOMJobTkValue	655
IDOMPathNode	742
IDOMVisualRoot	957
IDOMOutline	702
IDOMOutlineEntry	705
IDOMPDFAlternateImage	776
IDOMPathFigure	721
IDOMPathGeometry	727
IDOMPathGeometryBuilder	737
IDOMPathSegment	770
IDOMResource	835
IDOMAudioFile	333
IDOMICCProfile	571
IDOMImage	575

IDOMPrintTicket	805
IDOMRawDataFile	820
IDOMResourceDictionary	838
IDOMSecurityInfo	845
IDOMShape	883
IDOMTarget	910
IDOMActionArray	318
IDOMActionLaunch	322
IDOMExternalTarget	437
IDOMInternalTarget	614
IDOMPageRectTarget	712
IDOMPageTarget	718
IEDLNamespace	967
IEDLStream	973
IInputStream	1090
IOutputStream	1203
IRAOutputStream	1408
IEDLTempStore	976
IEDLTime	982
IFontHeaderWriteSegmentBlockEnumerator	988
IFontPCL5WriteSegmentBlockEnumerator	991
IImageDecoder	1036
IImageEncoder	1042
IImageFrame	1049
IImageFrameWriter	1054
IRunnable	1437
ISession	1442
JawsMako::IJawsMako	1104
IEDLTempStoreObject	980
JawsMako::IAnnotation	223
JawsMako::ILinkAnnotation	1155
JawsMako::IMarkupAnnotation	1165

JawsMako::ICaretAnnotation	240
JawsMako::IFreeTextAnnotation	1027
JawsMako::IInkAnnotation	1077
JawsMako::ILineAnnotation	1150
JawsMako::IPolyAnnotation	1365
JawsMako::IRedactionAnnotation	1415
JawsMako::IShapeAnnotation	1446
JawsMako::ISoundAnnotation	1454
JawsMako::IStampAnnotation	1458
JawsMako::ITextAnnotation	1484
JawsMako::ITextMarkupAnnotation	1488
JawsMako::IPopupAnnotation	1369
JawsMako::IWidgetAnnotation	1512
JawsMako::IAnnotationAppearance	232
JawsMako::IAnnotationReference	235
JawsMako::IDistiller	280
JawsMako::IDocument	305
JawsMako::IDocumentAssembly	313
JawsMako::IForm	1000
JawsMako::IFormField	1013
JawsMako::IInput	1081
JawsMako::IJPDSInput	1032
JawsMako::IPCL5Input	1246
JawsMako::IPCLXLInput	1258
JawsMako::IPDFInput	1289
JawsMako::IPPMLInput	1372
JawsMako::IPRNInput	1374
JawsMako::IPSInput	1387
JawsMako::IXPSInput	1543
JawsMako::IOXPSInput	1215
JawsMako::IJawsRenderer	1108
JawsMako::ILayout	1123

JawsMako::ILayoutFont	1126
JawsMako::ILayoutFontWeight	1130
JawsMako::ILayoutParagraph	1136
JawsMako::ILayoutRun	1143
JawsMako::ILayoutImageRun	1134
JawsMako::ILayoutTextRun	1145
JawsMako::IMarkedContentDetails	1162
JawsMako::IMarkedContentArtifactDetails	1160
JawsMako::IMarkedContentStructureDetails	1164
JawsMako::INamedDestination	1171
JawsMako::IOptionalContent	1174
JawsMako::IOptionalContentConfiguration	1178
JawsMako::IOptionalContentConfiguration::COrderEntry	123
JawsMako::IOptionalContentDetails	1181
JawsMako::IOptionalContentGroup	1185
JawsMako::IOptionalContentGroupReference	1187
JawsMako::IOptionalContentGroupUsage	1188
JawsMako::IOptionalContentGroupUsageApplication	1192
JawsMako::IOptionalContentVisibilityExpression	1193
JawsMako::IOutput	1196
JawsMako::IPCL5Output	1253
JawsMako::IPCLXLOutput	1265
JawsMako::IPDFOutput	1313
JawsMako::IPSOOutput	1390
JawsMako::IXPSOutput	1546
JawsMako::IOXPSOutput	1217
JawsMako::IOutputIntent	1200
JawsMako::IOutputWriter	1210
JawsMako::IPDFObject	1305
JawsMako::IPDFObjectStore	1309
JawsMako::IPDFPageExtractor	1342
JawsMako::IPDFPageInserter	1346

JawsMako::IPDFValidator	1360
JawsMako::IPJLParser	1362
JawsMako::IPage	1221
JawsMako::IPageLabel	1230
JawsMako::IPageLayout	1235
JawsMako::IPageLayoutData	1238
JawsMako::IPageLayoutNode	1240
JawsMako::IPageRaster	1241
JawsMako::IProgressMonitor	1377
JawsMako::IProgressTick	1380
JawsMako::ISVGGenerator	1478
JawsMako::ISeparator	1438
JawsMako::ISkiaRenderer	1451
JawsMako::IStructure	1466
JawsMako::IStructureElement	1468
JawsMako::IStructureElementChild	1472
JawsMako::IStructureElementReferenceChild	1474
JawsMako::IStructureMarkedContentReferenceChild	1475
JawsMako::IStructureObjectReferenceChild	1477
JawsMako::IStructureElementReference	1473
JawsMako::ITextRun	1492
JawsMako::ITextSearch	1494
JawsMako::ITextSelect	1496
JawsMako::IThreads	1497
JawsMako::ITransform	1499
JawsMako::IBlendSimplifierTransform	238
JawsMako::ICFFCIDSplitterTransform	243
JawsMako::ICFFSanitizerTransform	245
JawsMako::IColorConverterTransform	248
JawsMako::IComplexColorSimplifierTransform	274
JawsMako::ICustomTransform	277
JawsMako::IFontProcessorDeferredTransform	994

JawsMako::IFontProcessorTransform	997
JawsMako::IFormDeduplicatorTransform	1011
JawsMako::IFormUnpackerTransform	1024
JawsMako::IImageDownsamplerTransform	1038
JawsMako::IImageEncoderTransform	1043
JawsMako::IImageMergerTransform	1058
JawsMako::IOptionalContentFixerTransform	1183
JawsMako::IOverprintSimulationTransform	1212
JawsMako::IPageCropperTransform	1228
JawsMako::IPatternConverterTransform	1244
JawsMako::IRedactorTransform	1419
JawsMako::IRendererTransform	1422
JawsMako::ISoftMaskConverterTransform	1453
JawsMako::IStrokerTransform	1462
JawsMako::IType3UnpackerTransform	1508
JawsMako::ITransformChain	1504
JawsMako::IUnicodeHelper	1511
JawsMako::IWidgetAppearanceCharacteristics	1528
JawsMako::IXAMLGenerator	1534
Iterator	1483
PointTmpl< PointType >	1555
PointTmpl< double >	1555
PointTmpl< TItem >	1555
PValue	1555
SignatureID	1556

4 Class Index

4.1 Class List

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

BoxTmpl< PointType >	
Template for a PDF-style box. Similar to a rectangle but specified using a left, bottom, right and top coordinate	106

JawsMako::CAnnotationBorder	
A class representing an annotation's border (described in the PDF Specification as <code>BorderStyle</code>). 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	106
JawsMako::IXAMLGenerator::CAnnotationXAML	
Class for receiving XAML generated for annotation appearances in a bulk fashion	108
IDOMFontOpenType::CCIDMap	
For TrueType-based CIDFonts, the mapping from CIDs to GlyphIds for non-identity orderings	108
IDOMPDFImage::CCITTFaxParams	
Class to hold filter parameters for CCITTFax-compressed image data. Please see the PDF specification for the meaning of these parameters	110
CClassID	
An object to represent a 128-bit globally unique ID	111
CClassParams	
EDL Object Interface	113
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)	115
JawsMako::IJawsRenderer::CColorThresholdArrayHalftone	
As per <code>CThresholdArrayHalftone</code> , 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)	115
JawsMako::IPDFValidator::CContentError	
A class describing validation errors present in a particular location on a page	116
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	116
JawsMako::CFieldOption	
Class to store form field option array entries	117
JawsMako::IJawsRenderer::CFrameBufferInfo	
Description of a frame buffer for use with <code>renderSeparationsToFrameBuffers</code>	118
JawsMako::IPDFValidator::CGeneralError	
A class describing validation errors found in a PDF that are not tied to a location on a page	118
CGlyphsCluster	
A single cluster generated from the parallel Indices and Unicode strings present in an <code>IDOMGlyphs</code> node. This represents the smallest logical unit of text. Here, a single cluster may represent:	119

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: 120

CIndicesGlyph

Utility code allowing the simpler manipulation of glyph clusters represented by the Unicode↔String and Indices entries of an **IDOMGlyphs** object 120

JawsMako::ILayoutFont::CLayoutFontItem

An association of a font and it's font index 122

IDOMShadingPatternType4567Brush::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 122

JawsMako::IOptionalContentConfiguration::COrderEntry

Class for presenting the order that groups should be displayed in a user interface. May be arranged in a tree 123

JawsMako::IPDFValidator::CPageErrors

A collection of all the errors associated with a page 124

JawsMako::CPDFFarReference

A simple concrete class representing indirect reference data stored in a remote context, such as (for example) from a different PDF document 124

JawsMako::IPDFInput::CPdfFontInfo

Information about a font in a PDF file, obtained by scanning the PDF font structures 126

JawsMako::CPDFReference

A simple concrete class representing indirect reference data 127

JawsMako::IPDFInput::CPdfScannedInk

Basic information about an ink used in a PDF file, obtained by scanning the PDF page tree 129

JawsMako::IPJLParser::CPjIAttributeValue

A captured PjL attribute 129

JawsMako::CQuadPoint

A representation of a PDF Quadpoint, in DOM coordinates 130

JawsMako::CRectInset

A class which specifies an inset from a rectangle 130

JawsMako::ISVGGenerator::CResourceEntry

Resource entry 131

JawsMako::IXAMLGenerator::CResourceEntry

Resource entry 131

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 131

IDOMShape::CShapeDetails::C Span

Representation of a series of rectangles sharing the same y span 132

JawsMako::JawsRenderer::CSpotHalftone

Description of a simple spot halftone, at 45 degrees, using Jaws's default spot function. Used for monochrome rendering 133

JawsMako::CTemporaryStoreParameters	Allows the temporary storage parameters to be optionally overridden	133
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	134
JawsMako::IJawsRenderer::CThresholdHalftone	A halftone representing a simple threshold. Used for monochrome rendering	135
CTransformMatrix< TItem >	Matrix class - special 3x2 matrix	135
JawsMako::CTransformState	Class for tracking the graphics state leading to the point where a transform is applied	148
JawsMako::CXFAPacket	A class encapsulating an entry in an XFA array	150
JawsMako::IAnnotationUtils::CXMLResource	Simple class for tracking streams associated with XML generated by <code>generateXMLForDocument()</code>	150
IDOMArcSegment::Data	Initialization data	151
IDOMAudioFile::Data	Initialization data	151
IDOMCachedImage::Data	Initialization data	152
IDOMCanvas::Data	Initialization data	153
IDOMCharPathGroup::Data	Initialization data	153
IDOMColorSpaceDeviceN::Data	Initialization data	154
IDOMColorSpaceICCBased::Data	Initialization data	155
IDOMColorSpaceIndexed::Data	Initialization data	155
IDOMColorSpaceLAB::Data	Initialization data	156
IDOMCompositeImage::Data	Initialization data	156
IDOMDePremultiplierFilter::Data	Initialization data	157
IDOMDeviceNColorant::Data	Initialization data	158
IDOMExponentialFunction::Data	Initialization data	159

IDOMFilteredImage::Data	
Initialization data	159
IDOMFixedPage::Data	
Initialization data	160
IDOMFont::Data	
Initialization data	160
IDOMFontOpenType::Data	
Initialization data	161
IDOMFontOpenTypeTT::Data	
Initialization data	162
IDOMFontPCL5::Data	
Initialization data	163
IDOMFontPCLXL::Data	
Initialization data	164
IDOMFontSource::Data	165
IDOMFontSourceFromStream::Data	
Initialization data	166
IDOMFontSourceObfuscationConverter::Data	
Initialization data	167
IDOMFontSourceStreamFilter::Data	
Initialization data	168
IDOMForm::Data	
Initialization data	169
IDOMFormInstance::Data	
Initialization data	170
IDOMGlyph::Data	
Initialization data	171
IDOMGlyphs::Data	
Initialization data	171
IDOMGradientStop::Data	
Initialization data	172
IDOMGroup::Data	
Initialization data	172
IDOMGroupingFunction::Data	
Initialization data	173
IDOMICCProfile::Data	
Initialization data	174
IDOMImage::Data	
Initialization data	174
IDOMImageBitScalerFilter::Data	
Initialization data	175

IDOMImageBleederFilter::Data	
Initialization data	176
IDOMImageBrush::Data	
Initialization data	177
IDOMImageChannelSelectorFilter::Data	
Initialization data	177
IDOMImageColorConverterFilter::Data	
Initialization data	178
IDOMImageColorKeyFilter::Data	
Initialization data	179
IDOMImageColorSpaceSubstitutionFilter::Data	
Initialization data	179
IDOMImageDecodeFilter::Data	
Initialization data	180
IDOMImageDeindexerFilter::Data	
Initialization data	181
IDOMImageDeviceNToBaseFilter::Data	
Initialization data	181
IDOMImageDownsamplerFilter::Data	
Initialization data	182
IDOMImageInverterFilter::Data	
Initialization data	183
IDOMImageMaskExpanderFilter::Data	
Initialization data	183
IDOMImageMatteRemoverFilter::Data	
Initialization data	184
IDOMImageSpotMergerFilter::Data	
Initialization data	185
IDOMJobTkContent::Data	
Initialization data	185
IDOMJobTkGenericCharacterData::Data	
Initialization data	186
IDOMJobTkGenericNode::Data	
Initialization data	187
IDOMJobTkNode::Data	
Initialization data	187
IDOMJobTkValue::Data	
Initialization data	188
IDOMLinearGradientBrush::Data	
Initialization data	189
IDOMMaskedBrush::Data	
Initialization data	189

IDOMMatrix::Data	
Initialization data	190
IDOMNullBrush::Data	
Initialization data	190
IDOMOutline::Data	
Initialization data	191
IDOMOutlineEntry::Data	
Initialization data	191
IDOMPathFigure::Data	
Initialization data	192
IDOMPathGeometry::Data	
Initialization data	192
IDOMPathNode::Data	
Initialization data	193
IDOMPCLImage::Data	
Initialization data	193
IDOMPDFAlternateImage::Data	
Initialization data	194
IDOMPDFImage::Data	
Initialization data	195
IDOMPolyBezierSegment::Data	
Initialization data	196
IDOMPolyLineSegment::Data	
Initialization data	196
IDOMPolyQuadraticBezierSegment::Data	
Initialization data	197
IDOMPostScriptCalculatorFunction::Data	
Initialization data	198
IDOMPrintTicket::Data	
Initialization data	198
IDOMPSDImage::Data	
Initialization data	199
IDOMRadialGradientBrush::Data	
Initialization data	200
IDOMRawDataFile::Data	
Initialization data	200
IDOMRawImage::Data	
Initialization data	201
IDOMRecombineAlphaImage::Data	
Initialization data	202
IDOMRecombineImage::Data	
Initialization data	202

IDOMResourceDictionary::Data	
Initialization data	203
IDOMSampledFunction::Data	
Initialization data	204
IDOMShadingPatternType1Brush::Data	
Initialization data	205
IDOMShadingPatternType2Brush::Data	
Initialization data	205
IDOMShadingPatternType3Brush::Data	
Initialization data	206
IDOMShadingPatternType4567Brush::Data	
Initialization data	207
IDOMShape::Data	
Initialization data	207
IDOMSoftMaskBrush::Data	
Initialization data	208
IDOMSolidColorBrush::Data	
Initialization data	208
IDOMStitchingFunction::Data	
Initialization data	209
IDOMTIFFImage::Data	
Initialization data	210
IDOMTilingPatternBrush::Data	
Initialization data	210
IDOMTransparencyGroup::Data	
Initialization data	211
IDOMTrivialImage::Data	
Initialization data	212
IDOMType3Font::Data	
Initialization data	212
IDOMVisualBrush::Data	
Initialization data	213
IEDLNamespace::Data	
Initialization data	214
IEDLTempStore::Data	
Initialization data	214
IEDLTime::Data	
Initialization data	215
IDOMPDFImage::DCTParams	
Class to hold filter parameters for DCT-compressed image data. Please see the PDF specification for the meaning of these parameters	215

EDLIFStream	216
An ifstream that can deal with UTF8 file names on all platforms	
EDLOFStream	217
An ofstream that can deal with UTF8 file names on all platforms	
EDLQName	217
Implementation of qualified name class	
IDOMPDFImage::FlateLZWParams	221
Class to hold filter parameters for Flate or LZW-compressed image data. Please see the PDF specification for the meaning of these parameters	
IAbort	222
A simple class to signal an abort in processing	
JawsMako::IAnnotation	223
An interface class for an annotation	
JawsMako::IAnnotationAppearance	232
An interface class for an annotation appearance, describing the graphical content of an annotation in a given usage and state. Annotation appearances are immutable	
JawsMako::IAnnotationReference	235
A generic reference to an annotation. The target annotation might not be loaded. Chiefly used to refer to annotations from a Form	
JawsMako::IAnnotationUtils	236
JawsMako::IBlendSimplifierTransform	238
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	
JawsMako::ICaretAnnotation	240
A generic interface class for a caret annotation It is intended that future releases of JawsMako will extend this interface	
JawsMako::ICFFCIDSplitterTransform	243
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	
JawsMako::ICFFSanitizerTransform	245
A simple transform that scans and sanitises CFF Fonts for wider interoperability	
JawsMako::IColorConverterTransform	248
A transform for color conversion, converting all appropriate DOM contents to a desired target color space	
IColorManager	254
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 <code>get()</code> static function	
JawsMako::IComplexColorSimplifierTransform	274
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	

JawsMako::ICustomTransform		
	A transform that allows the implementation to be provided externally	277
IDOMPdfImage::IDecodeParams		
	Abstract interface for per-image decoding filter parameters	279
JawsMako::IDistiller		
	An instance of the JawsMako Distiller class	280
JawsMako::IDocument		
	A document from an IDocumentAssembly , allowing for high level document and page management, and providing on-demand lazy loading of page markup	305
JawsMako::IDocumentAssembly		
	A self contained collection of IDocuments	313
IDOMActionArray		
	IDOMActionArray interface	318
IDOMActionLaunch		
	IDOMActionLaunch interface	322
IDOMArcSegment		
	Interface to Arc Segment element	326
IDOMAudioFile		
	IDOMAudioFile interface	333
IDOMBrush		
	Interface to the brush element	336
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	339
IDOMCanvas		
	A canvas is a special form of an isolated, non-knockout, normal blended transparency group	343
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	352
IDOMCharPathGroup		
	IDOMCharPathGroup interface Interface to DOM node representing Group Elements that consist of stroked text	358
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	365
IDOMColorSpace		
	IDOMColorSpace interface	381
IDOMColorSpaceDeviceCMY		
	Represents the default CMY color space. NOTE: Currently for internal use only; Do not use this color space in your own applications	386
IDOMColorSpaceDeviceCMYK		
	Represents the default CMYK color space	388

IDOMColorSpaceDeviceGray IDOMColorSpaceDeviceGray interface	391
IDOMColorSpaceDeviceN This color space is analogous to the PostScript/PDF DeviceN/Separation color spaces	394
IDOMColorSpaceDeviceRGB IDOMColorSpaceDeviceRGB interface	402
IDOMColorSpaceICCBased Represents a color space described by an ICC profile	404
IDOMColorSpaceIndexed This color space is analogous to the PostScript/PDF Indexed color space	409
IDOMColorSpaceLAB This color space is as described in section 4.5.4 of the PDF 1.7 Reference Manual	413
IDOMColorSpacescRGB Represents the scRGB color space	417
IDOMColorSpacesGray Represents a gray color space using the sRGB gamma and WhitePoint	420
IDOMColorSpacesRGB Represents the RGB color space	423
IDOMCompositImage 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	426
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	430
IDOMDeviceNColorant This class enables the specification of colorant information for PDF style NChannel variants of DeviceN color spaces	431
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	434
IDOMExternalTarget IDOMExternalTarget interface	437
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	441
IDOMFixedPage Represents <FixedPage> element	446
IDOMFont IDOMFont Base Class	456
IDOMFontOpenType IDOMFontOpenType interface	461

IDOMFontOTFTrueType	
Opentype Font	472
IDOMFontPCL5	
IDOMFontPCL5 (PCL5 Truetype) derived from an OpenType font source	473
IDOMFontPCLXL	
This class models PCL XL TrueType and bitmap fonts derived from an OpenType font source	476
IDOMFontSource	
The font source for the class IDOMFont . This class describes the different ways fonts are constructed	479
IDOMFontSourceFromStream	
The source for IDOMFont when sourced from an existing stream	481
IDOMFontSourceObfuscationConverter	
Interface for a font sourced from a converter that performs obfuscation and deobfuscation	485
IDOMFontSourceStreamFilter	
An abstract interface for fonts sourced from a font stream filter	489
IDOMForm	
Interface to DOM node representing PDF-style Form XObjects	492
IDOMFormInstance	
IDOMFormInstance interface. This describes an instance of an IDOMForm in a DOM tree	499
IDOMFunction	
Base class for PDF/PS Style functions All function instances throw IEDLError exceptions on failure	505
IDOMGlyph	
Abstract class modelling a single character from a font	509
IDOMGlyphIDEnumerator	
DOM GlyphID Enumerator	521
IDOMGlyphName	524
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	525
IDOMGradientBrush	
A common interface for both IDOMLinearGradient and IDOMRadialGradient . Provides straightforward access to common attributes	550
IDOMGradientStop	
IDOMGradientStop defines the ramp of colors to use on a gradient	555
IDOMGroup	
Interface to DOM node representing Group Elements	558
IDOMGroupingFunction	
Interface to encapsulate an array of x-input-1-output functions	565
IDOMHashable	
Abstract interface for objects that can be hashed	569

IDOMICCProfile		
IDOMICCProfile interface		571
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	575
IDOMImageBitScalerFilter	An image filter that presents an image as an image with a different bits per sample	580
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	581
IDOMImageBrush	Provides an interface to a DOM image brush object	583
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	594
IDOMImageColorConverterFilter	An image filter that presents a color converted version of an image	595
IDOMImageColorKeyFilter	An image filter that presents a masked image where colors within a given range are masked out, analogous to a green screen. The source image must not have a mask or alpha channel	596
IDOMImageColorSpaceSubstitutionFilter	An image filter that presents an identical image, just with the colorspace substituted	598
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	599
IDOMImageDeindexerFilter	An image filter that presents an image with an Indexed color space as a simple eight bit image	600
IDOMImageDeviceNToBaseFilter	An image filter that presents an image with a DeviceN color space as a simple image in the alternate space	601
IDOMImageDownsamplerFilter	An image filter that presents a downsampled version of an image	603
IDOMImageInverterFilter	An image filter that presents a bitwise inverted form of the source image	604
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	605
IDOMImageProperties	The IDOMImageProperties interface provides access to an underlying implementation which stores miscellaneous information about the associated image	606

IDOMImageSpotMergerFilter	An image filter that merges selected spot components into the process components of an image. This is conceptually similar to IDOMImageDeviceNToBaseFilter , but:	611
IDOMInternalTarget	The IDOMInternalTarget interface describes the targets of hyperlinks that are in the same document but not on the current page	614
IDOMJobTk	Represents an EDL JobTicket	617
IDOMJobTkContent	Represents the content element of the JobTicket	622
IDOMJobTkGenericCharacterData	Interface to the IDOMJobTkGenericCharacterData node	633
IDOMJobTkGenericNode	Interface to the IDOMJobTkGenericNode node	638
IDOMJobTkNode	Represents a Job Ticket Node	644
IDOMJobTkOwner	Interface to the IDOMJobTkOwner node	650
IDOMJobTkValue	Represents a Job Ticket value element	655
IDOMJPEGImage	Interface to a class representing a JPEG (.jpg or .jpeg) image	659
IDOMLinearGradientBrush	IDOMLinearGradientBrush interface. A linear gradient brush is used to specify a gradient along a vector	664
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	670
IDOMMatrix	Defines the render transform matrix	676
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	679
IDOMMetadata	The IDOMMetadata interface provides access to the metadata attached to the Document↔ Sequence node. The IDOMMetadata interface is designed to be flexible enough to represent different types of metadata	679
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	684

IDOMNodeFlags	A collection of bit flags used to signal various conditions of the node. For example, the <code>e↔</code> <code>NodeRenderFlag</code> flag identifies nodes that require rendering	697
IDOMNullBrush	<code>IDOMNullBrush</code> provides a way of representing the default marking brush in a Type3 postscript glyph definition or a tiling pattern with <code>paintType 2</code> . This is more of a placeholder that gets replaced when the Type3 glyph or <code>paintType 2</code> tiling pattern is actually invoked	699
IDOMOutline	Represents the outline of the document, which is the collection of bookmarks for the document	702
IDOMOutlineEntry	Represents an index to a specific location in the document or a specific location external to the document	705
IDOMPageRectTarget	<code>IDOMPageRectTarget</code> nodes are used to describe hyperlinks on a page rectangle to targets on the same page	712
IDOMPageTarget	<code>IDOMPageTarget</code> nodes are used to describe hyperlinks on a page to targets on the same page	718
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 <code>IEDLError</code> for error handling	721
IDOMPathGeometry	Interface to a path geometry node	727
IDOMPathGeometryBuilder	Interface to a path geometry builder	737
IDOMPathNode	Interface to an EDL path node. A path node specifies a geometry that can be filled or stroked with a brush	742
IDOMPathSegment	Interface to path segment element. The path segment is the smallest unit in a path geometry	770
IDOMPCLImage	Interface to a class representing an image extracted from a PCLXL file	773
IDOMPDFAlternateImage	An encapsulation of a PDF alternate image, which may be referenced from an image or mask brush	776
IDOMPDFImage	Interface to a class representing an image extracted from a PDF file. Intended to be only used with the <code>JawsMako</code> APIs	780
IDOMPNGImage	Interface to a class representing a PNG (.png) image	785
IDOMPolyBezierSegment	Interface to a path segment node describing a set of cubic Bézier curves	790

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	794
IDOMPolyQuadraticBezierSegment	
Interface to a polyquadratic Bézier segment. A polyquadratic 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 (x _{2n-1} , y _{2n-1}) followed by the end point (x _{2n} , y _{2n}) for each quadratic Bézier curve (where n represents the quadratic Bézier curve)	798
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	802
IDOMPrintTicket	
IDOMPrintTicket interface	805
IDOMPSDImage	
IDOMPSDImage interface	807
IDOMPublicKeyPDFSecurityInfo	
Represents security information from PDF public key (certificate based) security handling. Inherits from IDOMStandardPDFSecurityInfo , from which it diverges in a modest fashion	811
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	813
IDOMRawDataFile	
IDOMRawDataFile interface	820
IDOMRawImage	
Interface to a class representing a raw image	823
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	827
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	831
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	835

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	838
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	841
IDOMSecurityInfo	Base DOM security class	845
IDOMShadingPatternBrush	IDOMShadingBrush provides a way of representing a PS style shading pattern	846
IDOMShadingPatternType1Brush	IDOMShadingBrush provides a way of representing a PS style type 1 shading pattern	852
IDOMShadingPatternType2Brush	IDOMShadingBrush provides a way of representing a PS style type 2 shading pattern	857
IDOMShadingPatternType3Brush	IDOMShadingPatternType3Brush provides a way of representing a PS style type 2 shading pattern	864
IDOMShadingPatternType4567Brush	IDOMShadingPatternType4567Brush provides a way of representing a PS style type 4 shading pattern	873
IDOMShape	Interface to an IDOMShape	883
IDOMSoftMaskBrush	IDOMSoftMaskBrush provides a way of representing a PDF style soft mask in its entirety. 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	890
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	894
IDOMStandardPDFSecurityInfo	Represents security information from PDF Standard encryption handler	902
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	906
IDOMTarget	Base class for defining hyperlink targets in a document	910
IDOMTIFFImage	IDOMTIFFImage interface	912
IDOMTilingPatternBrush	IDOMTilingPatternBrush provides a way of representing a PS style tiling pattern	918

IDOMTransformableBrush	Abstract interface for a brush to which a render transform may be applied	925
IDOMTransparencyGroup	IDOMTransparencyGroup interface. Analogous to PDF Transparency groups	927
IDOMTrivialImage	Interface for a "trivial" image	938
IDOMType3Font	Representation of a PostScript/PDF Type 3 Font. At present, the stream cannot be set, only retrieved	943
IDOMVisualBrush	A visual brush is used to fill a region with a vector drawing	949
IDOMVisualRoot	IDOMVisualRoot interface	957
IDOMWMPImage	IDOMWMPImage interface	961
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	964
IEDLError	An abstract class for EDL exceptions	967
IEDLNamespace	Interface to EDL Namespace class	967
IEDLObject	IEDLObject is an abstract base class that is used by all classes that are intended to be created via an EDL class factory	970
IEDLStream	Generic stream. Abstract base class for EDL stream subsystem	973
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	976
IEDLTempStoreObject	A mechanism for storing and accessing temporary data for use with Jaws Mako	980
IEDLTime	Interface to EDL date-time class	982
IFontHeaderWriteSegmentBlockEnumerator	IFontHeaderWriteSegmentBlockEnumerator Enumerates over the PCLXL Font Header block items for the XL ReadFontHeader operator	988
IFontPCL5WriteSegmentBlockEnumerator	IFontPCL5WriteSegmentBlockEnumerator Enumerates over the PCL5 font blocks	991
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	994

JawsMako::IFontProcessorTransform	
A transform for performing font subsetting and merging	997
JawsMako::IForm	
An interface class for an interactive form, which tracks a tree of IFormFields and widgets	1000
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	1011
JawsMako::IFormField	
An interface class for a form field. A form field may have multiple child fields and widget annotations, arranged in a tree	1013
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	1024
IFrame	
A frame, into which text can be flowed by the layout engine	1026
JawsMako::IFreeTextAnnotation	
A generic interface class for a free text annotation It is intended that future releases of JawsMako will extend this interface	1027
JawsMako::IHashable	
Simple interface to provide a consistent hashing method for Mako objects	1031
JawsMako::IJPDSInput	
An instance of the JawsMako IJPDS input class	1032
JawsMako::IJPDSPageRaster	
An instance of the JawsMako IJPDS page raster that contains the source input page number and rip number	1034
IImageDecoder	
IImageDecoder returns IImageFrame objects as requested by the client. This object knows about the imageformat internals and knows how to unpack the image	1036
JawsMako::IImageDownsamplerTransform	
A transform for downsampling images above a given effective resolution to a desired target effective resolution	1038
IImageEncoder	
IImageEncoder accepts IImageFrame objects and streams it out to an image format	1042
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	1043
IImageFrame	
IImageFrame encapsulates an EDL image with its details	1049
IImageFrameWriter	
IImageFrameWriter writes an image from an imageframe	1054

JawsMako::ImageMergerTransform	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	1058
JawsMako::ICustomTransform::Implementation	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	1061
JawsMako::IInkAnnotation	A generic interface class for a ink annotation It is intended that future releases of JawsMako will extend this interface	1077
JawsMako::IInput	Abstract input source that can open files from disk or a stream and create an IDocumentAssembly for the contents	1081
IInputEnum< typename T >	Iterator template class to allow iteration over a collection of instances of type <T>	1085
IInputEnumRC< typename T >	Reference-counted iterator template class to allow iteration over a collection of instances of type <T>	1085
IInputPushbackStream	Input Stream with pushback support	1086
IInputStream	Generic input stream. Abstract base class for all input streams	1090
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	1104
JawsMako::IJawsRenderer	A renderer that uses the Jaws RIP to create images from arbitrary DOM	1108
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	1123
JawsMako::ILayoutFont	ILayoutFont interface	1126
JawsMako::ILayoutFontWeight	Font weights used by ILayoutFont for font selection	1130
JawsMako::ILayoutImageRun	A run of image content to be added to an ILayoutParagraph	1134
JawsMako::ILayoutParagraph	A paragraph, consisting of a number of runs of content	1136
JawsMako::ILayoutRun	A run of content to be added to an ILayoutParagraph	1143
JawsMako::ILayoutTextRun	A run of text content to be added to an ILayoutParagraph	1145

JawsMako::ILineAnnotation	An interface class for a line annotation. It is intended that future releases of JawsMako will extend this interface	1150
JawsMako::ILinkAnnotation	A generic interface class for a link annotation It is intended that future releases of JawsMako will extend this interface	1155
JawsMako::IMarkedContentArtifactDetails	A subclass of IMarkedContentDetails that is created when the content is a logical structure Artifact	1160
JawsMako::IMarkedContentDetails	Details of Marked Content applied to an IDOMGroup	1162
JawsMako::IMarkedContentStructureDetails	A subclass of IMarkedContentDetails that is created when the marked content is associated with the document's structure	1164
JawsMako::IMarkupAnnotation	An interface class for markup annotations. It is intended that future releases of JawsMako will extend this interface	1165
JawsMako::IMediaHandler	Interface allowing users of supported input types to monitor/handle unsatisfied media requests. For input interfaces that support this feature, use <code>setMediaHandler()</code> to install subclasses of this type	1171
JawsMako::INamedDestination	A named destination in a PDF Document	1171
JawsMako::IOptionalContent	Root level optional content information for an entire document	1174
JawsMako::IOptionalContentConfiguration	A configuration for optional content	1178
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	1181
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	1183
JawsMako::IOptionalContentGroup	Interface for an optional content group	1185
JawsMako::IOptionalContentGroupReference	A reference to an optional content group	1187
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	1188
JawsMako::IOptionalContentGroupUsageApplication	Interface for controlling how IOptionalContentGroupUsage is applied, and for what groups	1192

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	1193
JawsMako::IOutput	Abstract output sink that can output DOM to a file or stream in a given output format	1196
JawsMako::IOutputIntent	Interface class representing a PDF output intent	1200
IOutputStream	Generic output stream. Abstract base class for output streams	1203
JawsMako::IOutputWriter	A writer for writing individual pages and documents to an output in piecemeal fashion	1210
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	1212
JawsMako::IOXPSInput	An instance of the JawsMako OpenXPS input class	1215
JawsMako::IOXPSOutput	Interface for the OpenXPS IOutput class	1217
JawsMako::IPage	A page from an IDocument , allowing high level page management, and providing on-demand access to page contents	1221
JawsMako::IPageCropperTransform	Very simple transform for cropping pages to one of the standard boxes	1228
JawsMako::IPageLabel	Interface class representing a PDF page label	1230
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	1235
JawsMako::IPageLayoutData	Provides a representation of the analyzed page layout by organizing and allowing access to collections of IPageLayoutNodes	1238
JawsMako::IPageLayoutNode	Simple data type representing a part of an analyzed page	1240
JawsMako::IPageRaster	A rasterized equivalent of a page from an IDocument	1241
JawsMako::IPatternConverterTransform	Transform to convert PDF/PS tiling/shading patterns to XPS- compatible forms where possible	1244
JawsMako::IPCL5Input	An instance of the JawsMako PCL5 input class	1246
JawsMako::IPCL5Output	Interface for the PCL5 IOutput class	1253
JawsMako::IPCLXLAttributeHandler	Interface allowing users to monitor/handle illegal PCL/XL operator attributes	1257

JawsMako::IPCLXLInput	An instance of the JawsMako PCL/XL input class	1258
JawsMako::IPCLXLOutput	Interface for the PCLXL IOutput class	1265
JawsMako::IPDFArray	A simple class representing a mutable array of other PDF objects	1268
JawsMako::IPDFBoolean	A simple immutable boolean PDF object type	1276
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	1278
JawsMako::IPDFFarReference	A simple class representing an immutable PDF indirect reference from a remote context such as (for example) from a different PDF document	1286
JawsMako::IPDFInput	An instance of the JawsMako PDF input class	1289
JawsMako::IPDFInteger	A simple immutable integer PDF object type	1300
JawsMako::IPDFNull	A simple immutable "null" pdf object	1302
JawsMako::IPDFObject	Abstract interface for a PDF internal object and common interfaces. All non-composite objects are immutable. All PDF objects are hashable	1305
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	1309
JawsMako::IPDFOutput	Interface for the PDF IOutput class	1313
JawsMako::IPDFPageExtractor	An instance of the JawsMako PDFPageExtractor class	1342
JawsMako::IPDFPageInserter	An instance of the JawsMako PDFPageInserter class	1346
JawsMako::IPDFReal	A simple immutable floating-point PDF object type	1350
JawsMako::IPDFReference	A simple class representing an immutable PDF indirect reference	1352
JawsMako::IPDFString	A simple immutable string pdf object containing raw unencoded data or PDF Text information	1356
JawsMako::IPDFValidator	A class for validating PDF documents against published PDF standards such as PDF/X	1360
JawsMako::IPJLParser	A PJL (Printer Job Language) parser for Mako	1362

JawsMako::IPolyAnnotation

An interface class for a polygon or polyline annotation. It is intended that future releases of JawsMako will extend this interface 1365

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 1369

JawsMako::IPPMLInput

An instance of the JawsMako PPML input class 1372

JawsMako::IPRNInput

An instance of the JawsMako PRN input class 1374

JawsMako::IProgressMonitor

An abstract class encapsulating both an **IProgressTick** and an **IAbort** so that the caller can monitor the progress and/or abort the processing 1377

JawsMako::IProgressTick

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()** 1380

JawsMako::IPSCCommentMonitor

Interface allowing users to monitor comments in PostScript input. Use **setCommentMonitor()** to install subclasses of this type 1383

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 1384

JawsMako::IPSInput

An instance of the JawsMako PS input class 1387

JawsMako::IPSOutput

Interface for the PS **IOutput** class 1390

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()** 1397

IRAInputPushbackStream

Random-access Input Stream with pushback support 1399

IRAInputStream

Random Access Input Stream 1403

IRAOutputStream

Random Access Output Stream 1408

IRAStream

Abstract base class for "Random-Access" streams i.e. streams that can be arbitrarily re-positioned 1410

IRCOBJECT

Base class Interface for all Reference Counted objects 1412

JawsMako::IRedactionAnnotation	
A generic interface class for a redaction annotation	1415
JawsMako::IRedactorTransform	
A transform for applying redaction redactions	1419
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	1422
IRunnable	
Interface to filter's runnable classes	1437
JawsMako::ISeparator	
An instance of the JawsMako Separator class	1438
ISession	
EDL session class	1442
JawsMako::IShapeAnnotation	
A generic interface class for circle and square annotations. It is intended that future releases of JawsMako will extend this interface	1446
JawsMako::ISkiaRenderer	
A renderer that can paint XPS compatible DOM into a Skia canvas using the Skia API	1451
JawsMako::ISoftMaskConverterTransform	
A transform that converts PDF style soft masks to opacity masks, suitable for XPS	1453
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	1454
JawsMako::IStampAnnotation	
A generic interface class for a stamp annotation	1458
JawsMako::IStrokerTransform	
A transform for converting some or all stroked paths into plain filled paths	1462
JawsMako::IStructure	
Top level tracking structure describing the logical structure of the document	1466
JawsMako::IStructureElement	
A structure element in the structure tree	1468
JawsMako::IStructureElementChild	
A child of a structure element. Either points to actual marked content, or another structure element	1472
JawsMako::IStructureElementReference	
A token-like class encapsulating a reference to a structure element	1473
JawsMako::IStructureElementReferenceChild	
A child of a structure element that points to another structure element	1474
JawsMako::IStructureMarkedContentReferenceChild	
A child of a structure element that points to a piece of marked content. Note; to create these, please see IStructureElement::createMarkedContentReferencePair()	1475

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()	1477
JawsMako::ISVGGenerator	
An SVG generator for JawsMako, allowing simple generation of SVG fragments for individual DOM nodes or entire pages	1478
Iterator	
A convenience iterator-like class for iterating through the contents of a dictionary	1483
JawsMako::ITextAnnotation	
A generic interface class for a text (sticky note) annotation	1484
JawsMako::ITextMarkupAnnotation	
A generic interface class for a text markup annotation It is intended that future releases of JawsMako will extend this interface	1488
JawsMako::ITextRun	
A run of text, containing unicode information, the position, transformation and bounds of the text	1492
JawsMako::ITextSearch	
Perform text searching using the page information obtained from an IPageLayout	1494
JawsMako::ITextSelect	
Perform text selection using the page information obtained from an IPageLayout	1496
JawsMako::IThreads	
An interface class for document threads, Currently a stub interface	1497
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	1499
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	1504
JawsMako::IType3UnpackerTransform	
A transform for unpacking glyphs using a Type 3 font into regular DOM	1508
JawsMako::IUnicodeHelper	
An interface into language specific unicode helpers	1511
JawsMako::IWidgetAnnotation	
An interface class for a widget annotation It is intended that future releases of JawsMako will extend this interface	1512
JawsMako::IWidgetAppearanceCharacteristics	
Appearance information for a widget annotation	1528
JawsMako::IXAMLGenerator	
A XAML generator for Mako, allowing simple generation of XAML fragments for individual DOM nodes or entire pages	1534
JawsMako::IXPSInput	
An instance of the JawsMako XPS input class	1543

JawsMako::IXPSOutput	Interface for the XPS IOutput class	1546
IDOMPDFImage::JBIG2Params	Class to hold filter parameters for JBIG2-compressed image data. Please see the PDF specification for the meaning of these parameters	1554
PointTpl< PointType >	Geometry primitives including: point, rectangle and matrix types supporting both integer and floating point values within	1555
PValue	Stores a "property" value that is tagged with an enumeration value that indicates the underlying type	1555
SignatureID	Opentype table signatures	1556

5 File Index

5.1 File List

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

edlblend.h		1558
edlmath.h	(very thin) portability layer around operating system provided math functionality but also includes a definition of "pi" and functions to interconvert between (angular) degrees and radians	1558
edlnamespaces.h	EDL C++ namespace(s)	1558
edlproperty.h	EDL uses the concept of a "property" that can store a value that has one of a number of different types (integer, string, pointer, time etc.)	1558
edlqname.h	IEDLNamespace and EDLQName classes	1558
edlquartz.h	A renderer that allows painting of EDL DOM into a Quartz2D Core Graphics context. Currently only XPS-compatible DOM is supported. The renderer is reentrant and can be used on multiple threads, providing that the destination context supports this. The renderer also caches certain objects to improve performance of repeat renders	1558
edlstring.h	EDLString and EDLSysString classes and associated EDL string manipulation functions	1558
edltime.h	IEDLTime class that represents the EDL date-time	1558
edltypes.h	EDL "standard" types including known bit-length signed and unsigned integer type[def]s and definitions of EDL_TRUE and EDL_FALSE	1558
edlvector.h	Simple buffer class which preserves the allocation context where the buffers were created	1558

edlversion.h	Defines Mako version numbers	1558
idombrush.h		1558
idomfont.h		1558
idomfontpcl.h	IDOMFont PCL generation interfaces	1558
idomform.h		1558
idomnode.h		1558
idomresources.h	IDOMResources interface	1558
iedlcollection.h	EDL "collection" interface	1558
iedlenum.h	EDL iterator template classes designed to allow iteration over the contents of a collection. Then a set of typedefs that provide collections of doubles, FPoints, FRects, EDLStrings and EDLSysStrings that can then be iterated over	1558
iedltree.h	EDL "tree" interface	1558
ifilespec.h	Interface to File specification	1558
iimagecodec.h		1558
ircobject.h	Interface for Reference Counted Object	1558
memutils.h	EDL portability wrappers around memset(), memcpy() and memcmp() to allow EDL to use alternate implementations where necessary	1558
objclassid.h	An object class ID is a 128-bit globally unique ID (i.e. a GUID). In EDL it is abstracted within a CClassID class that provides methods for constructing GUIDs from 4 unsigned 32-bit integers or a string or even another CClassID and a method for comparing GUIDs	1558
platform.h	Platform-dependent defines, enumerations, types etc. that are visible through the EDL API	1558
smartptr.h	EDL smart pointers which, in conjunction with the IRCOject class provide reference-counted and automatically garbage-collected IEDLObjects which are typically returned by one of the EDL class factories	1558
customtransform.h	A framework to allow transforms to be written externally of Mako	1558
distiller.h	PostScript to PDF conversion	1558
ijpdsinput.h	IJPDS Input-Specific Features	1558

interactive.h		
JawsMako interactive features		1558
jawsmake.h		1558
optionalcontent.h		1558
pcl5input.h		
PCL5 Input-Specific Features		1558
pcl5output.h		
JawsMako PCL5 Output		1558
pclxlinput.h		
PCL/XL Input-Specific Features		1558
pclxloutput.h		
JawsMako PCLXL Output		1558
pdfinput.h		
PDF Input-Specific Features		1558
pdfobjects.h		1558
pdfoutput.h		
JawsMako PDF Output		1558
pdfpage.h		
PDF Page Management Features		1558
ppmlinput.h		
Ppml Input-Specific Features		1558
prninput.h		
PRN Input-Specific Features		1558
psinput.h		
PostScript Input-Specific Features		1558
separator.h		1558
structure.h		1558
text.h		1558
transforms.h		
Transforms for JawsMako, allowing direct modification of individual brushes, nodes, or entire pages		1558
types.h		1558
xpsoutput.h		
JawsMako XPS Output		1558

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::get↔EDLErrorString()**)*

- [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::Error](#) 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

<i>tempDir</i>	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.
<i>cacheDir</i>	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.
<i>tempStoreParams</i>	The desired temporary store parameters. See CTemporaryStoreParams for details.
<i>initialParams</i>	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_RES_DICT_OBJECT	XPS input: Resource dictionary is missing the key attribute.
EDL_ERR_ADD_OBJECT_RES_DICT	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_IGNOREABLE	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_IGNOREABLE	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 its entirety. 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

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

eColorInterpolationMode

enum `eColorInterpolationMode`

Color interpolation mode type enumeration.

Enumerator

<code>eSRgbLinearInterpolation</code>	Linear interpolation for sRGB color space.
<code>eSCRGBLinearInterpolation</code>	Linear interpolation for scRGB color space.

eSpreadMethodenum `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.

eTilingModeenum `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.

eViewUnitsenum `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::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` { }
- enum `IFontHeaderWriteSegmentBlockEnumerator::eFontHeaderWriteSegmentItem`
The enumeration of the segment types to be enumerated over.
- enum `IDOMFontPCLXL::ePCLXLFontType` { `IDOMFontPCLXL::ePCLXLFontTypeTTF` , `IDOMFontPCLXL::ePCLXLFontTypeBit` }
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 substitute for font undefined glyphs.
----------------	---

6.1.4.8 Geometry

DOM interfaces for path geometry, figures and segments.

Classes

- class `PointTpl< PointType >`
Geometry primitives including: point, rectangle and matrix types supporting both integer and floating point values within.
- class `BoxTpl< 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 or stroked 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 polyquadratic Bézier segment. A polyquadratic 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 >::elsComplex](#) = 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::eClipLongMitters](#) , [IDOMPathNode::eBevelLongMitters](#) }
- Chooses how miters that extend beyond the miter limit should be treated. ClipLongMitters specifies XPS style behaviour, where miters extending beyond the limit are clipped to the limit. BevelLongMitters 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 outside. 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 inside.

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 [IDOMTrivialImage](#)

Interface for a "trivial" 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 an 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 [IDOMCompositImage](#)

Interface to a class representing an 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 colorspace substituted.
- class [IDOMImageColorConverterFilter](#)
An image filter that presents a color 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 color space as a simple eight bit image.
- class [IDOMImageDeviceNToBaseFilter](#)
An image filter that presents an image with a DeviceN color 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 colors 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::eAnnotationEditingAllowed](#)
= 0x0020 ,
[IDOMStandardPDFSecurityInfo::eFormFillingAllowed](#) = 0x0100 , [IDOMStandardPDFSecurityInfo::eContentAccessibilityExtractionAllowed](#)
= 0x0200 , [IDOMStandardPDFSecurityInfo::eDocumentAssemblyAllowed](#) = 0x0400 , [IDOMStandardPDFSecurityInfo::eHighQualityPrintAllowed](#)
= 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 as 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` = `eTextStyleItalic` | `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

<code>eExternal</code>	(PDF URI Action) Resolve a uniform resource identifier.
<code>eInternal</code>	XPS internal target.
<code>ePage</code>	XPS page target.
<code>ePageRect</code>	(PDF Goto Action) Go to a destination in the current document.
<code>eActionGoToR</code>	("Go-to remote") Go to a destination in another document.
<code>eActionGoToE</code>	("Go-to embedded"; PDF 1.6) Go to a destination in an embedded file.
<code>eActionLaunch</code>	Launch an application, usually to open a file.
<code>eActionThread</code>	Begin reading an article thread.
<code>eActionSound</code>	(PDF 1.2) Play a sound.
<code>eActionMovie</code>	(PDF 1.2) Play a movie.
<code>eActionHide</code>	(PDF 1.2) Set an annotation's Hidden flag.
<code>eActionNamed</code>	(PDF 1.2) Execute an action predefined by the viewer application.
<code>eActionSubmitForm</code>	(PDF 1.2) Send data to a uniform resource locator.
<code>eActionResetForm</code>	(PDF 1.2) Set fields to their default values.
<code>eActionImportData</code>	(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::IJPDSInput`
An instance of the JawsMako IJPDS input class.
- class `JawsMako::IJPDSPageRaster`
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::IPSOOutput](#)
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 edlSprintf (char *buffer, size_t n, const char *format,...)`
Implementation of `sprintf` that always uses the C locale. Parameters as per `sprintf()`.
- `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

<i>subdir</i>	Specifies the first part of the path to the temporary directory to be created. The path is treated as a UTF8 string.
<i>pidNo</i>	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

<i>filename</i>	The path to the file.
<i>mode</i>	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

<i>subdir</i>	Specifies the first part of the path to the temporary directory to be created. The path is treated as a UTF8 string.
<i>pidNo</i>	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

<i>subdir</i>	Specifies the first part of the path to the temporary directory to be created. The path is treated as a UTF8 string.
<i>pidNo</i>	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.
<i>fullSubDir</i>	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

<i>dir</i>	Specifies the path to be created. If the parameter is specified as <code>/foo/bar/dir</code> , and the path <code>/foo/bar</code> already exists, the directory <code>dir</code> will be created within <code>bar</code> . If the parameter is specified as <code>/foo/bar</code> and <code>/foo</code> already exists, then both <code>bar</code> and <code>bar/dir</code> 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

<i>dir</i>	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

<i>buffer</i>	C string to accept the result
<i>n</i>	Buffer size
<i>format</i>	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

<i>str</i>	C string with the representation of a floating-point number.
<i>endptr</i>	Reference to an already allocated object of type <code>char *</code> , whose value is set by the function to the next character in <code>str</code> 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

<i>buffer</i>	C string to accept the result
<i>n</i>	Buffer size
<i>format</i>	Format string
<i>ap</i>	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::eBUAlternatImage](#) }
- 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

<i>jawsMako</i>	A smart pointer to the JawsMako instance.
<i>implementation</i>	The custom transformation implementation.
<i>abort</i>	An abort callback to allow your transformation to be aborted.
<i>dependsOnClipBounds</i>	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.
<i>dependsOnGroupSpace</i>	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.
<i>dependsOnRenderingIntent</i>	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

<i>dependsOnTransformScale</i>	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.
<i>dependsOnTransformOffset</i>	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 <i>dependsOnTransformOffset</i> is set, then <i>dependsOnTransformScale</i> is typically also set.
<i>dependsOnBrushUsage</i>	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.
<i>dependsOnEdgeMode</i>	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.
<i>dependsOnUncoloredTilingBrush</i>	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

<i>jawsMako</i>	A smart pointer to the JawsMako instance.
<i>implementation</i>	The custom transformation implementation.
<i>abort</i>	An abort callback to allow your transformation to be aborted.
<i>dependsOnClipBounds</i>	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.
<i>dependsOnGroupSpace</i>	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.
<i>dependsOnRenderingIntent</i>	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.
<i>dependsOnTransform</i>	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 <i>dependsOnTransformScale</i> and <i>dependsOnTransformOffset</i> set to true in the alternate form of the creator below.
<i>dependsOnBrushUsage</i>	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.
<i>dependsOnEdgeMode</i>	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.
<i>dependsOnUncoloredTilingBrush</i>	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::JawsRenderer](#)

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](#) = 0 , [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

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

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

- class [JawsMako::JawsRenderer::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::JawsRenderer::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::JawsRenderer::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 rendered. 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::JawsRenderer::CThresholdHalftone](#)

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

- class [JawsMako::JawsRenderer::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](#) { [eBPCUnset](#) = -1 , [eBPCDefault](#) = 0 , [eBPCOn](#) = 1 , [eBPCOff](#) = 2 }

Black point compensation enumeration.

- enum [eRenderingIntent](#) {
[eRenderingIntentUnset](#) = -1 , [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` , `eNumColor`↔
SpaceTypes = 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

<code>eDeviceRGB</code>	Device RGB color space.
<code>eDeviceGray</code>	Device Gray color space.
<code>eDeviceCMYK</code>	Device CMYK color space.
<code>esRGB</code>	XPS sRGB color space with input range scaled to between 0 and 1.
<code>esGray</code>	Gray with sRGB gamma color space.
<code>escRGB</code>	XPS scRGB color space.
<code>eICCBased</code>	Color space with an ICC profile.
<code>eIndexed</code>	A color space analogous to the PostScript language/PDF Indexed color space,
<code>eDeviceN</code>	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
<code>eLAB</code>	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
<code>eDeviceCMY</code>	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 `IPageLayoutNodes`.
- 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 BoxTpl< 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 BoxTpl< 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

eBorderStyle

```
enum JawsMako::CAnnotationBorder::eBorderStyle
```

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 eBorderStyle & inType = eBTSolid,
    const CEDLVector< float > & inDash = CEDLVector<float>() ) [inline]
```

Constructor.

Parameters

<i>inWidth</i>	Border width. Default width is 1 PDF unit, or 1/72nd of an inch
<i>inType</i>	Border type. Default is eBTSolid
<i>inDash</i>	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:

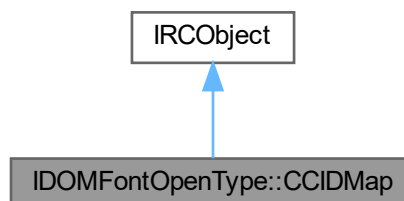
- xamlgenerator.h

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:

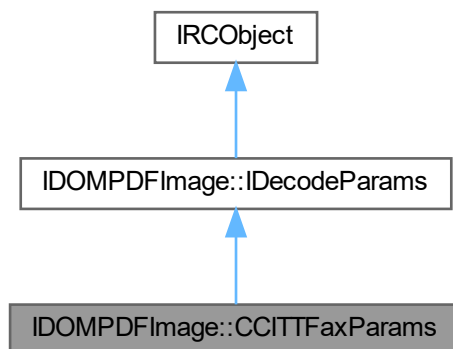
- [idomfont.h](#)

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 [IRObject](#)

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

<i>str</i>	String containing the hexadecimal value
------------	---

CClassID() [2/3]

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

Copy to another [CClassID](#).

Parameters

<i>other</i>	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

<i>dw0</i>	First unsigned, 32-bit integer
<i>dw1</i>	Second unsigned, 32-bit integer
<i>dw2</i>	Third unsigned, 32-bit integer
<i>dw3</i>	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

<i>id</i>	CClassID with which to compare
-----------	--

Returns

bool True if equal, false if not

operator=()

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

operator =

Parameters

<i>other</i>	CClassID
--------------	----------

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

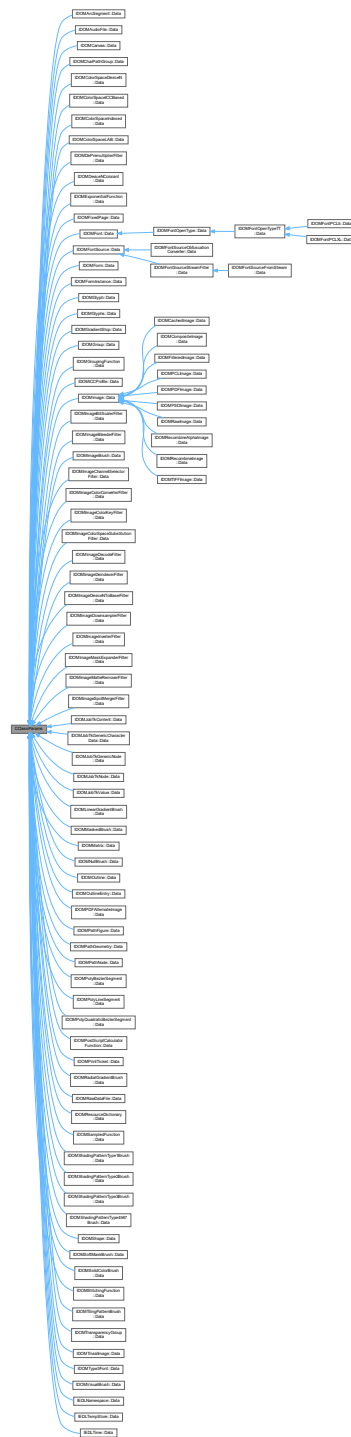
- [objclassid.h](#)

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:

- `iedobject.h`

7.8 JawsMako::JawsRenderer::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::JawsRenderer::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::JawsRenderer::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::JawsRenderer::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 apply 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:

- [jawsmake.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 frame buffer.
- uint32 **bufferOfs**
Offset 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

<i>other</i>	The cluster to compare.
<i>eps</i>	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 clusters 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

<i>other</i>	The indices glyph to compare.
<i>eps</i>	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 5 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<FPPoint> IDOMShadingPatternType4567Brush::CMeshEntry::points
```

A variable number of points. The interpretation of these points depends on the shading type and (for all but Type 5 shades) the flag entry.

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

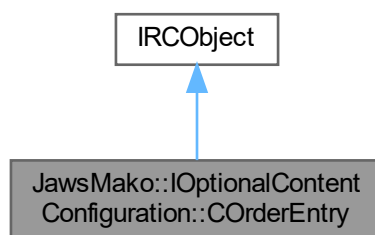
- [idombrush.h](#)

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

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

<i>objectNum</i>	The object number to use
<i>generation</i>	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 be 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

<i>memoryLimit</i>	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.
<i>diskLimit</i>	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.
<i>blockSize</i>	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, $(\text{diskLimit} + \text{memoryLimit}) / \text{blockSize}$ 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 , **elsComplex** = 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 RectTpl< TItem > &sourceRect, const RectTpl< 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 ignoreDXDY=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

<code>_xx</code>	xx component
<code>_xy</code>	xy component
<code>_yx</code>	yx component
<code>_yy</code>	yy component
<code>_dx</code>	dx component
<code>_dy</code>	dy component

CTransformMatrix() [2/3]

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

Copy constructor.

Parameters

<i>m</i>	Transform matrix
----------	------------------

CTransformMatrix() [3/3]

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

Creates a matrix that transforms from one rectangle to another.

Parameters

<i>sourceRect</i>	Source rectangle
<i>destRect</i>	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

<i>array</i>	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 PointTpl< 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

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

decompose()

```
template<typename TItem >
void CTransformMatrix< TItem >::decompose (
    PointTpl< 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

<i>translate</i>	Reference to receive the translation component in x and y.
<i>rotationAngle</i>	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).
<i>shearAngle</i>	Reference to receive the shear angle in radians.
<i>scale</i>	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

<i>matrix</i>	Matrix to compare current instance to
<i>ignoreDXDY</i>	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

<i>ignoreDXDY</i>	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

<i>point</i>	The point
<i>ignoreDXDY</i>	Ignore DX & DY for this operation

Returns

std::pair<bool, [PointTpl](#)<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 (
    PointTpl< TItem > & result,
    const PointTpl< TItem > & point,
    bool ignoreDXDY = false ) const [inline]
```

Transform a point by the inverse of the matrix.

Parameters

<i>result</i>	PointTpl <TItem> to accept result
<i>point</i>	The point
<i>ignoreDXDY</i>	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

<i>m</i>	Transform matrix
----------	------------------

postMul()

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

Postmultiply by given matrix.

Parameters

<i>matrix</i>	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

<i>matrix</i>	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

<i>radians</i>	Angle in radians
----------------	------------------

scale()

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

Scale.

Parameters

<i>xscale</i>	X-axis scaling factor
<i>yscale</i>	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

<i>_xx</i>	xx component
<i>_xy</i>	xy component
<i>_yx</i>	yx component
<i>_yy</i>	yy component
<i>_dx</i>	dx component
<i>_dy</i>	dy component

setDX()

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

Sets dx component.

Parameters

<i>x</i>	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

<i>x</i>	value to set
----------	--------------

transform() [1/2]

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

Transform a point.

Parameters

<i>point</i>	The point before transformation
<i>ignoreDXDY</i>	Ignore DX & DY for this operation

Returns

CTransformMatrix<TItem> Resulting matrix

transform() [2/2]

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

Transform a point.

Parameters

<i>result</i>	The point after transformation
<i>point</i>	The point before transformation
<i>ignoreDXDY</i>	Ignore DX & DY for this operation

transformRect()

```
template<typename TItem >
void CTransformMatrix< TItem >::transformRect (
    RectImpl< TItem > & rect,
    bool ignoreDXDY = false ) const [inline]
```

Transform a rectangle.

Transform a rectangle

Parameters

<i>rect</i>	Rectangle to transform
<i>ignoreDXDY</i>	Ignore DX & DY for this operation

translate()

```
template<typename TItem >
void CTransformMatrix< TItem >::translate (
    TItem dx,
    TItem dy ) [inline]
```

Translate.

Parameters

<i>dx</i>	x-axis transformation value
<i>dy</i>	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 attempt will be made to determine the complete state by concatenating 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 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.
- **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 Render↔Transforms 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

<i>brush</i>	The brush whose state we wish to apply.
<i>brushUsage</i>	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

<i>node</i>	The node whose state we wish to apply.
<i>updateTransform</i>	If true, update the transform entry, and also the related black point compensation entry.
<i>updateClip</i>	If true, update the clipBounds entry
<i>updateRenderingIntent</i>	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.
- **InputStreamPtr 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:

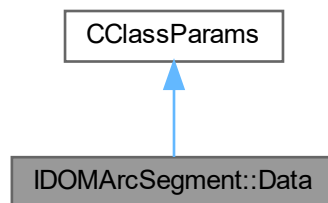
- [interactive.h](#)

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:

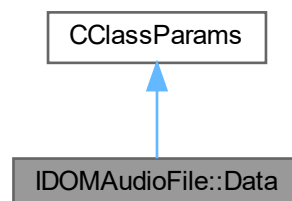
- [idompathgeometry.h](#)

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:

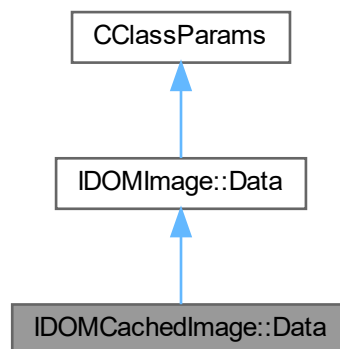
- [idomresources.h](#)

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:

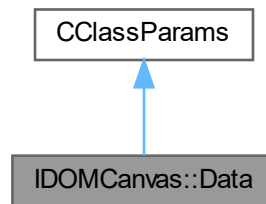
- [idomimageresource.h](#)

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:

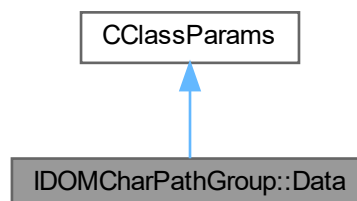
- idomcanvas.h

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:

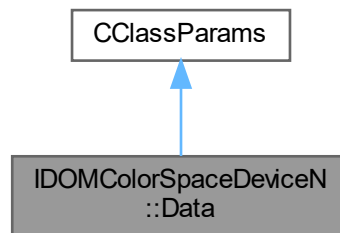
- idomcharpathgroup.h

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:

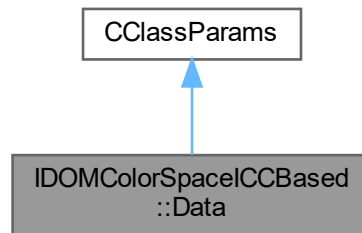
- idomcolorspace.h

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:

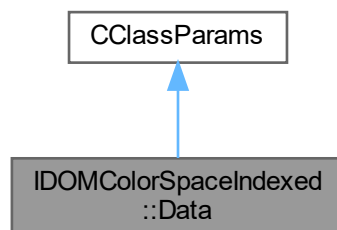
- idomcolorspace.h

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:

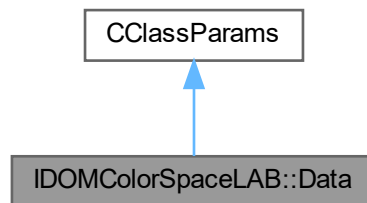
- idomcolorspace.h

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:

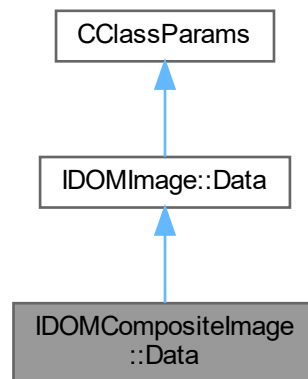
- idomcolorspace.h

7.50 IDOMCompositImage::Data Class Reference

Initialization data.

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

Inheritance diagram for IDOMCompositelImage::Data:



7.50.1 Detailed Description

Initialization data.

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

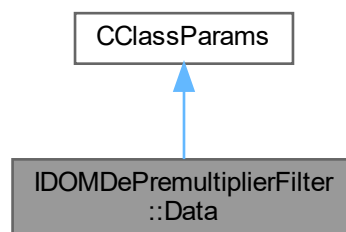
- idomimageresource.h

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:

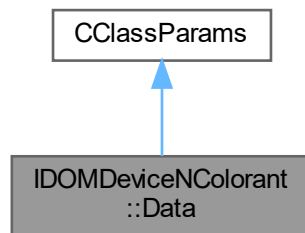
- idomimageresource.h

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:

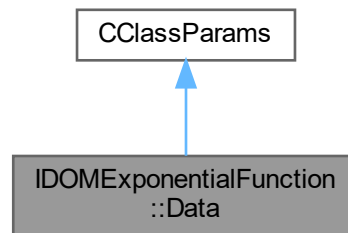
- idomcolorspace.h

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:

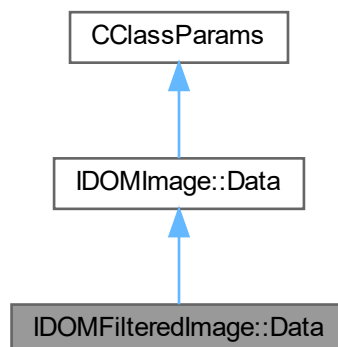
- `idomfunction.h`

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:

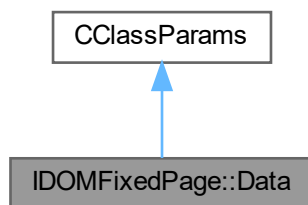
- idomimageresource.h

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:

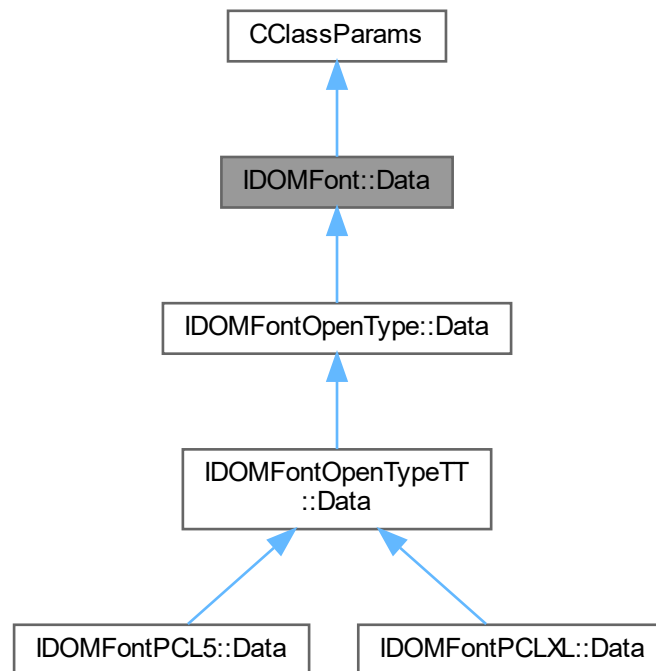
- idompage.h

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:

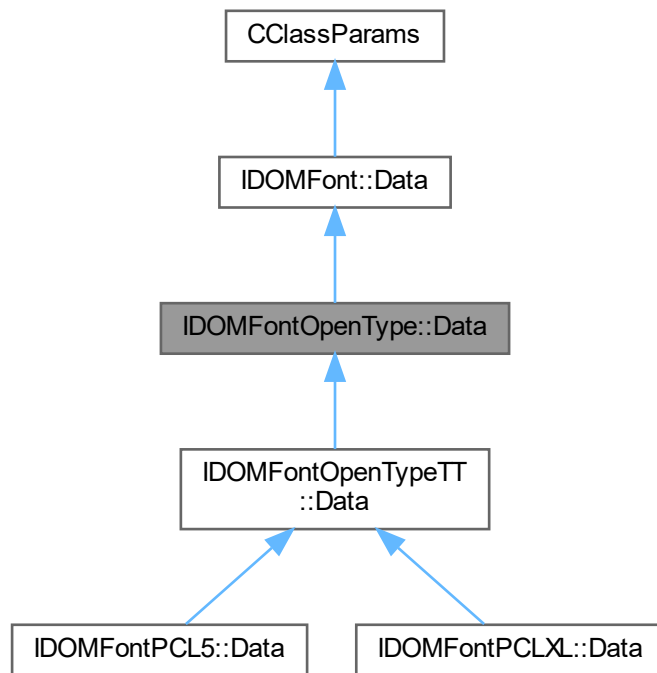
- [idomfont.h](#)

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:

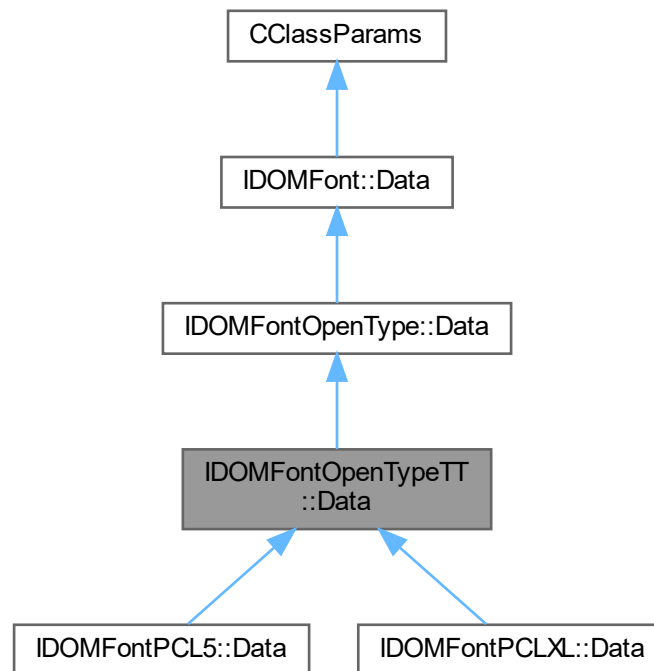
- [idomfont.h](#)

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:

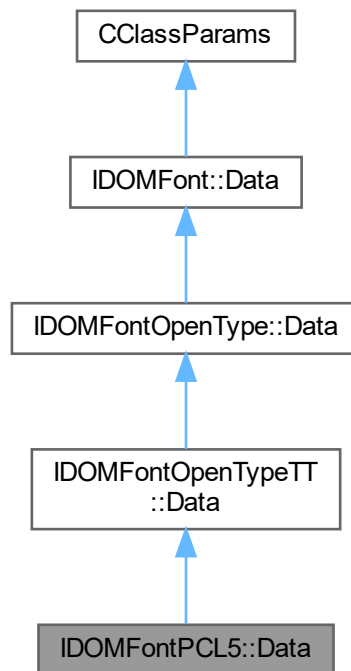
- [idomfont.h](#)

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:

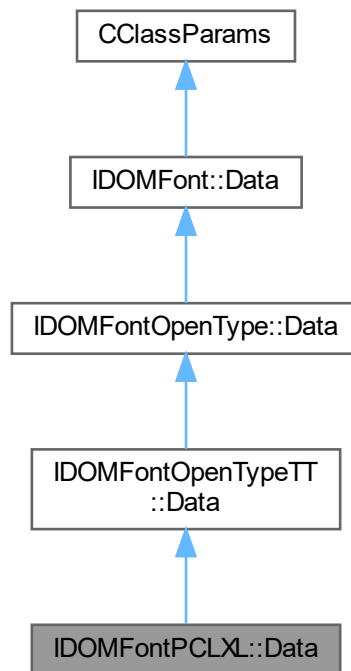
- [idomfontpcl.h](#)

7.60 IDOMFontPCLXL::Data Class Reference

Initialization data.

```
#include <idomfontpcl.h>
```

Inheritance diagram for IDOMFontPCLXL::Data:



7.60.1 Detailed Description

Initialization data.

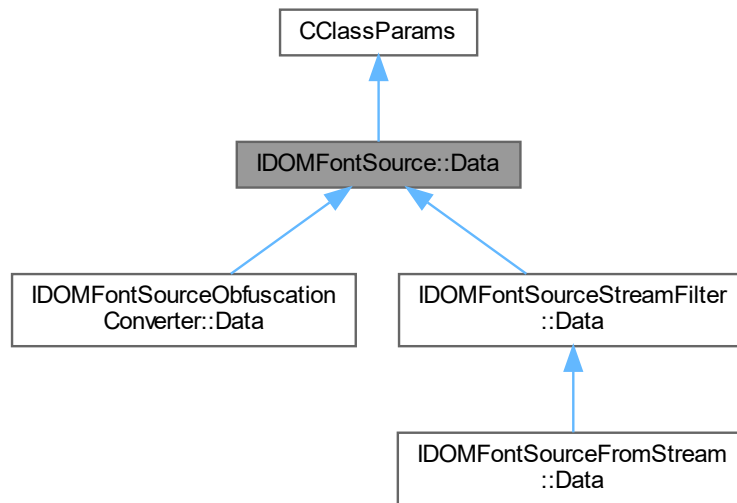
The documentation for this class was generated from the following file:

- [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:

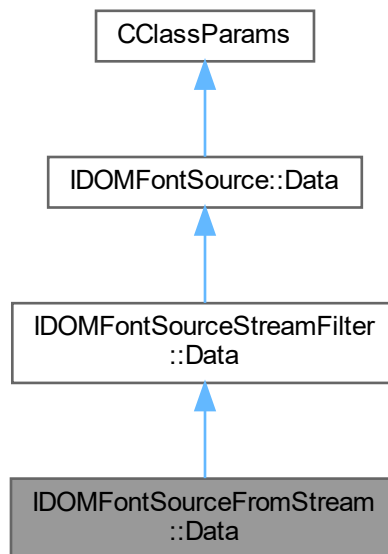
- [idomfont.h](#)

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:

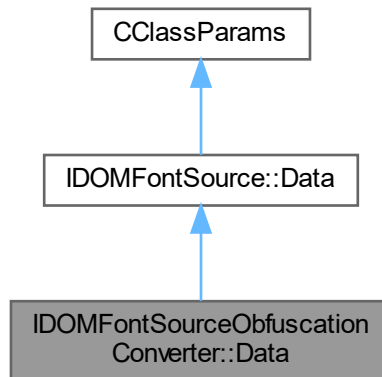
- [idomfont.h](#)

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:

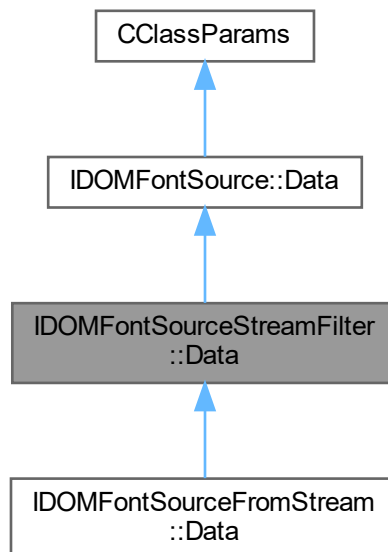
- [idomfont.h](#)

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:

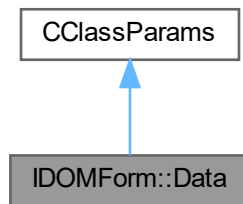
- [idomfont.h](#)

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:

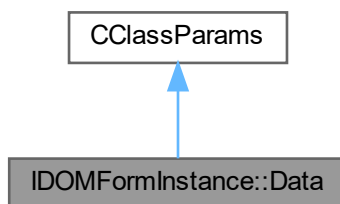
- [idomform.h](#)

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:

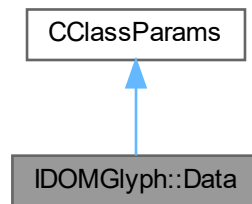
- [idomform.h](#)

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:

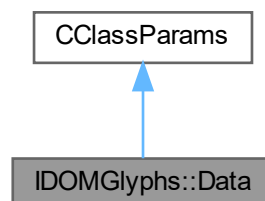
- idomglyph.h

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:

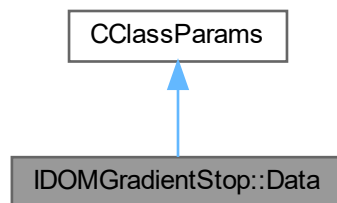
- [idomglyphs.h](#)

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:

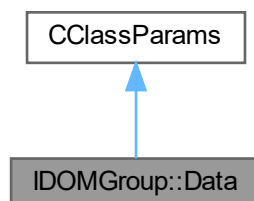
- [idombrush.h](#)

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:

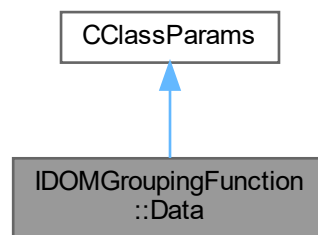
- idomgroup.h

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:

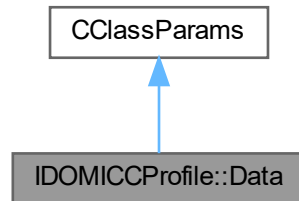
- idomfunction.h

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:

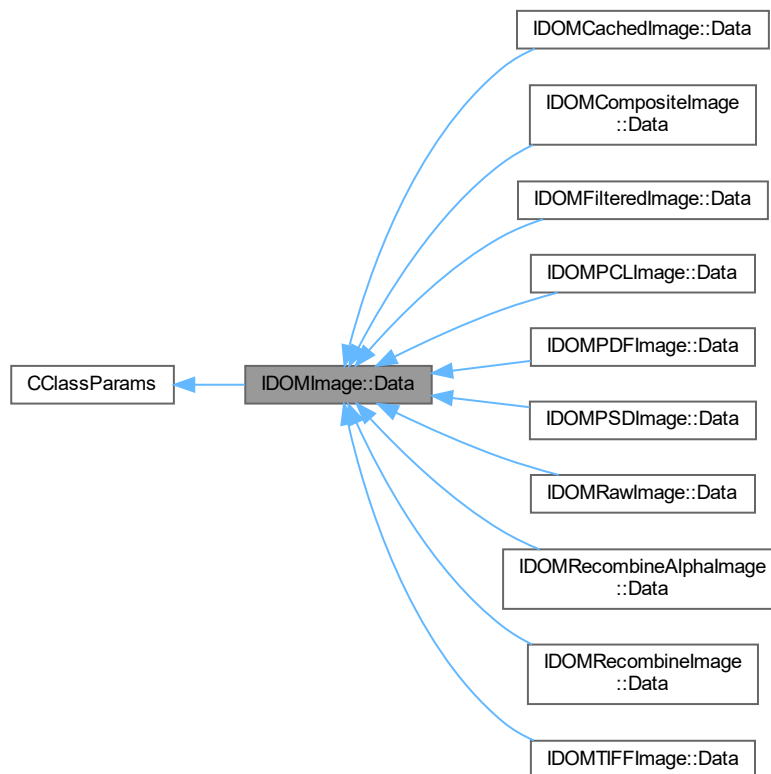
- [idomresources.h](#)

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:

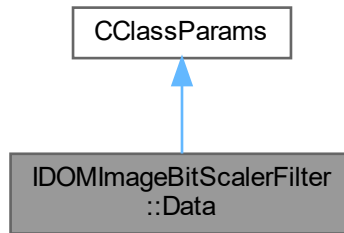
- idomimageresource.h

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:

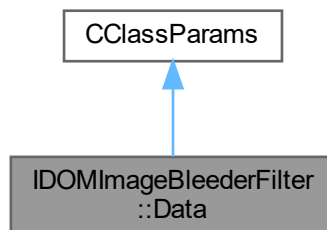
- idomimageresource.h

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:

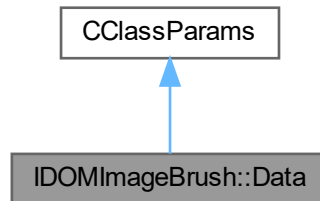
- idomimageresource.h

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:

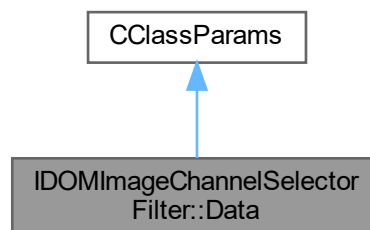
- [idombrush.h](#)

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:

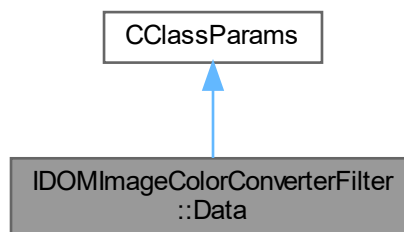
- idomimageresource.h

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:

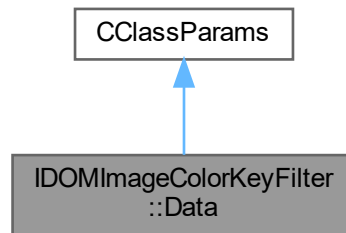
- idomimageresource.h

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:

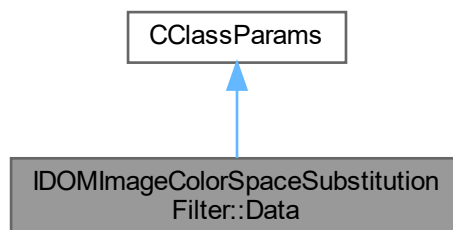
- idomimageresource.h

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:

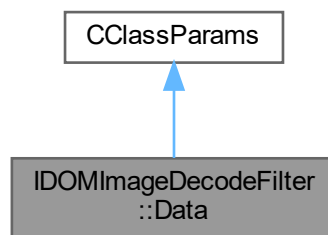
- idomimageresource.h

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:

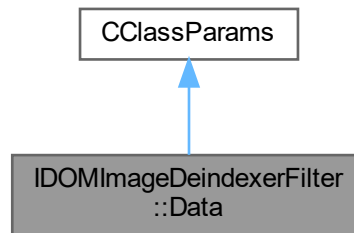
- idomimageresource.h

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:

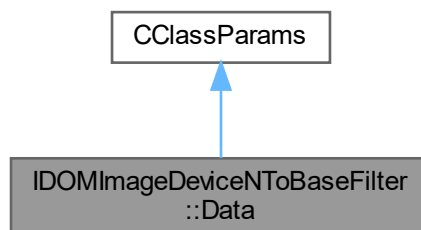
- idomimageresource.h

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:

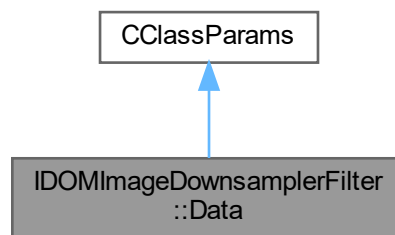
- idomimageresource.h

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:

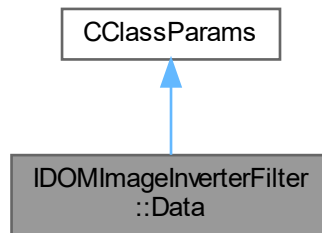
- idomimageresource.h

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:

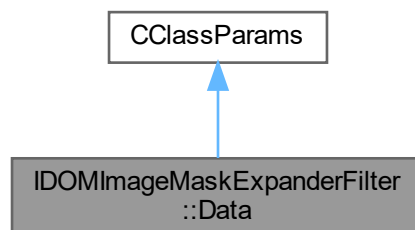
- idomimageresource.h

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:

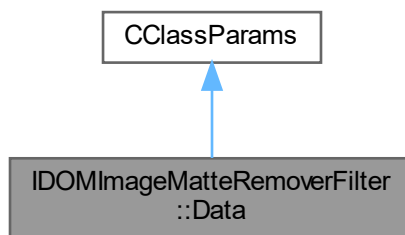
- idomimageresource.h

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:

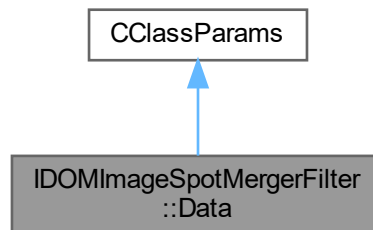
- idomimageresource.h

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:

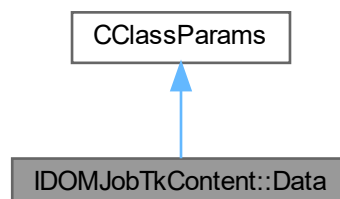
- idomimageresource.h

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:

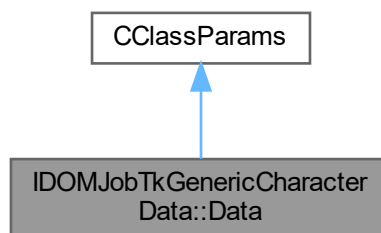
- idomjobtk.h

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:

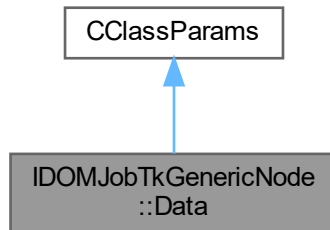
- idomjobtk.h

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:

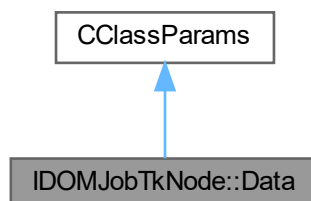
- idomjobtk.h

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:

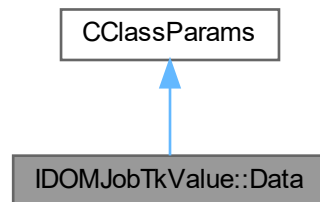
- idomjobtk.h

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:

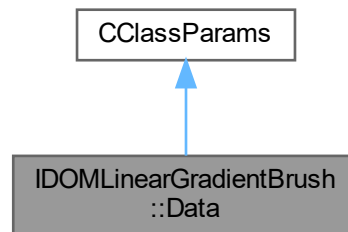
- idomjobtk.h

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:

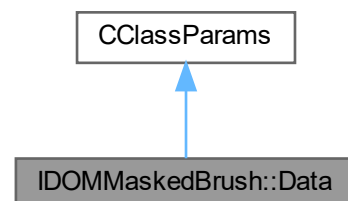
- [idombrush.h](#)

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:

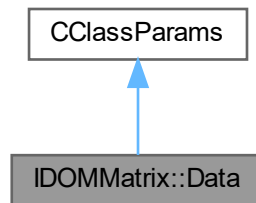
- [idombrush.h](#)

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:

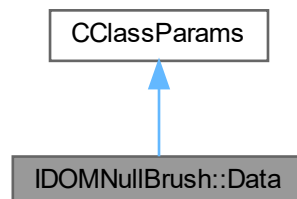
- [idomresources.h](#)

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:

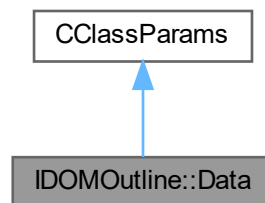
- [idombrush.h](#)

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:

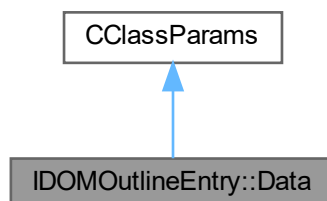
- [idomoutline.h](#)

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:

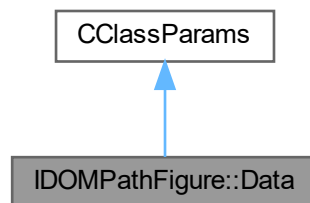
- idomoutline.h

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:

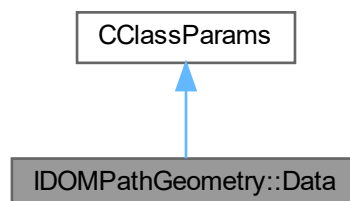
- idompathgeometry.h

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:

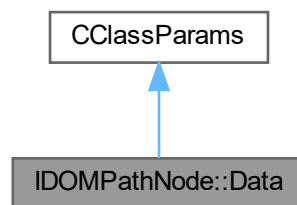
- idompathgeometry.h

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:

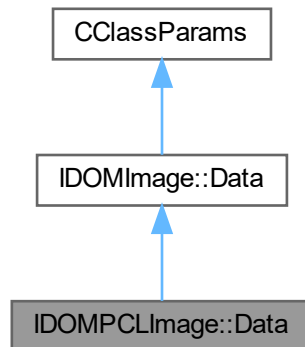
- idompath.h

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:

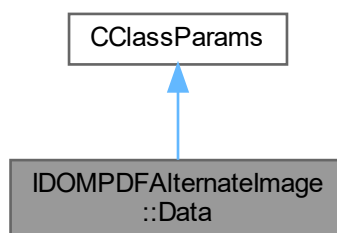
- idomimageresource.h

7.104 IDOMPDFAlternatImage::Data Class Reference

Initialization data.

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

Inheritance diagram for IDOMPDFAlternatImage::Data:



7.104.1 Detailed Description

Initialization data.

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

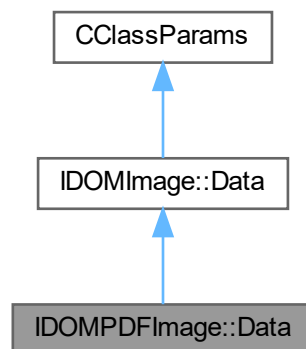
- idomimageresource.h

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:

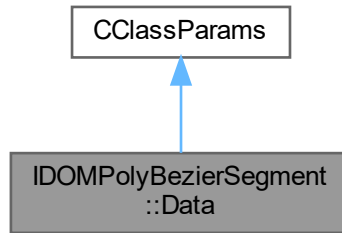
- idomimageresource.h

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:

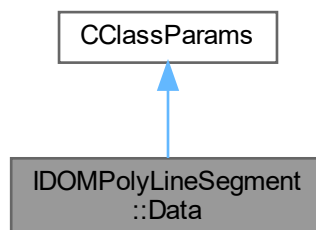
- idompathgeometry.h

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:

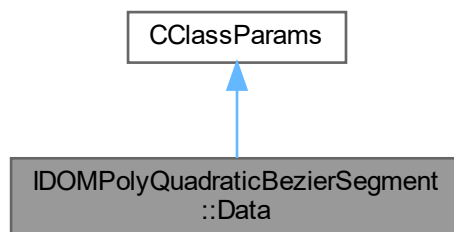
- idompathgeometry.h

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:

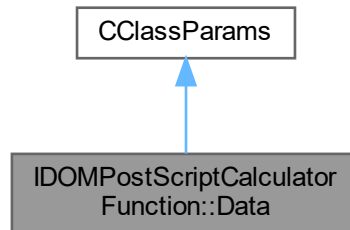
- idompathgeometry.h

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:

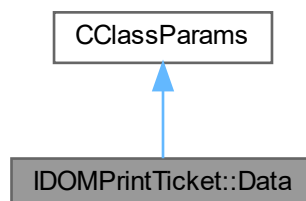
- idomfunction.h

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:

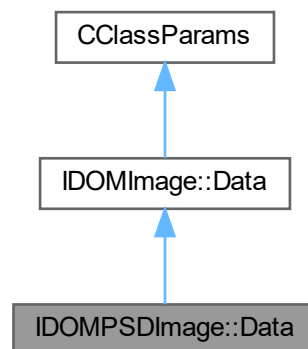
- [idomresources.h](#)

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:

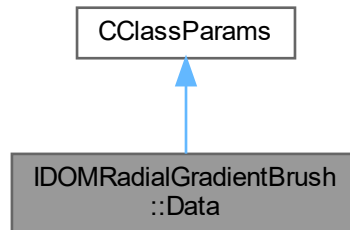
- [idomimageresource.h](#)

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:

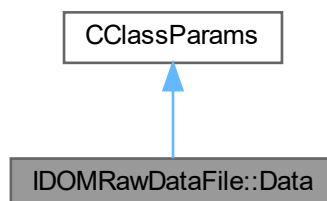
- [idombrush.h](#)

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:

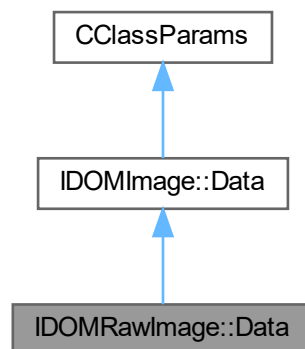
- [idomresources.h](#)

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:

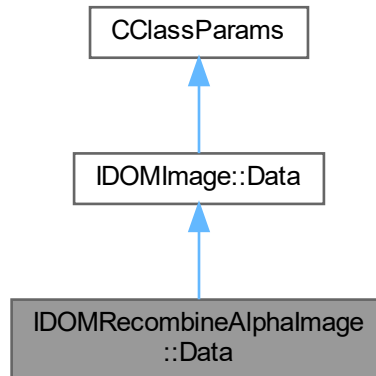
- [idomimageresource.h](#)

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:

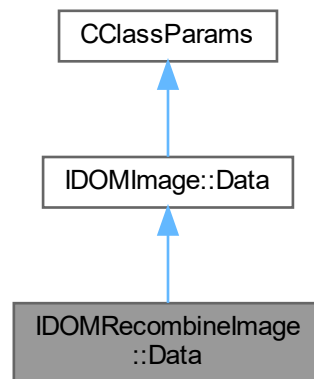
- `idomimageresource.h`

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:

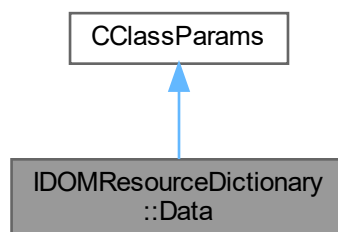
- idomimageresource.h

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:

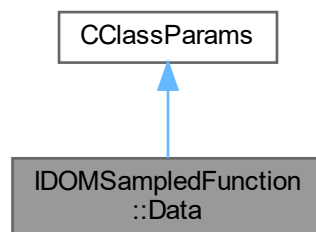
- [idomresources.h](#)

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:

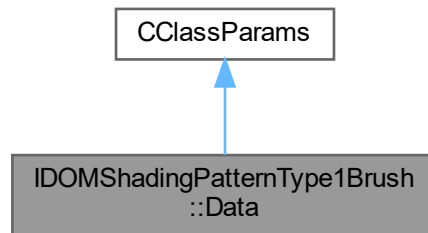
- [idomfunction.h](#)

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:

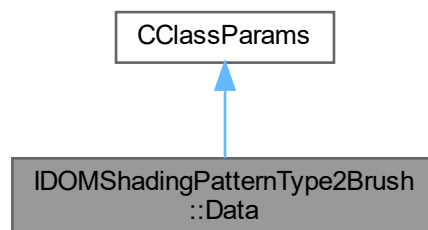
- [idombrush.h](#)

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:

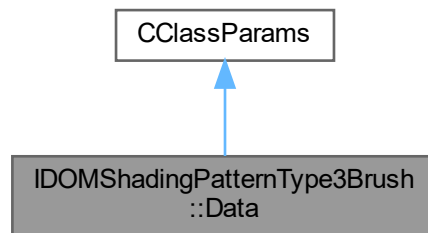
- [idombrush.h](#)

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:

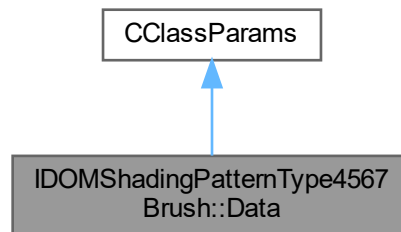
- [idombrush.h](#)

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:

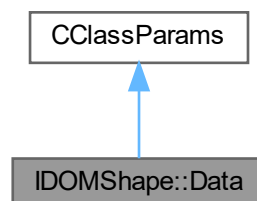
- [idombrush.h](#)

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:

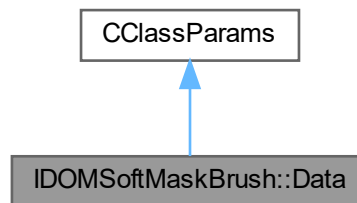
- [idomshape.h](#)

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:

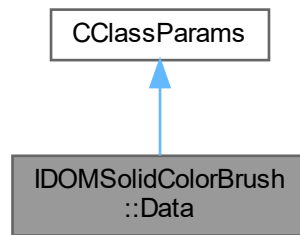
- [idombrush.h](#)

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:

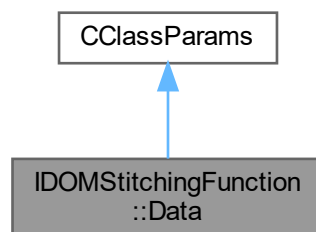
- [idombrush.h](#)

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:

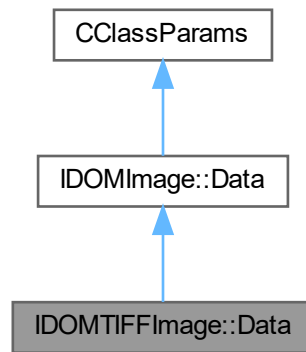
- [idomfunction.h](#)

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:

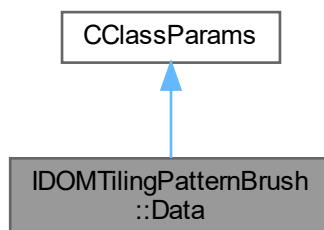
- idomimageresource.h

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:

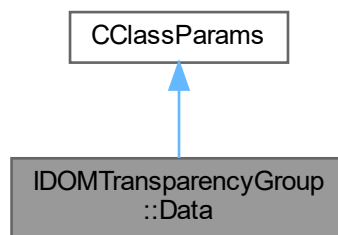
- [idombrush.h](#)

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:

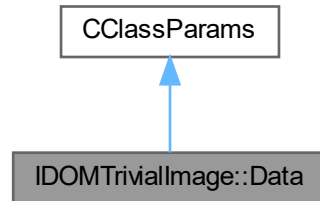
- [idomgroup.h](#)

7.130 IDOMTrivialImage::Data Class Reference

Initialization data.

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

Inheritance diagram for IDOMTrivialImage::Data:



7.130.1 Detailed Description

Initialization data.

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

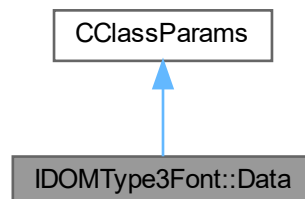
- idomimageresource.h

7.131 IDOMType3Font::Data Class Reference

Initialization data.

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

Inheritance diagram for IDOMType3Font::Data:



7.131.1 Detailed Description

Initialization data.

```
@defgroup IDOMType3FontData IDOMType3Font initialization data
@ingroup IFont
```

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

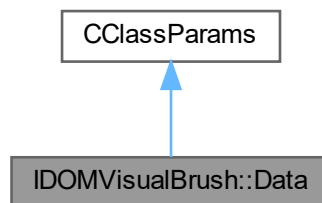
- [idomfont.h](#)

7.132 IDOMVisualBrush::Data Class Reference

Initialization data.

```
#include <idombrush.h>
```

Inheritance diagram for IDOMVisualBrush::Data:



7.132.1 Detailed Description

Initialization data.

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

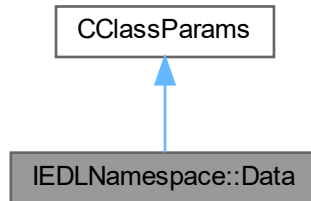
- [idombrush.h](#)

7.133 IEDLNamespace::Data Class Reference

Initialization data.

```
#include <edlqname.h>
```

Inheritance diagram for IEDLNamespace::Data:



7.133.1 Detailed Description

Initialization data.

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

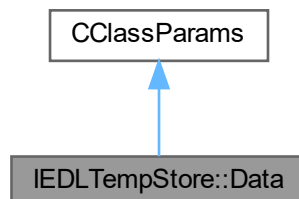
- [edlqname.h](#)

7.134 IEDLTempStore::Data Class Reference

Initialization data.

```
#include <iedltempstore.h>
```

Inheritance diagram for IEDLTempStore::Data:



7.134.1 Detailed Description

Initialization data.

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

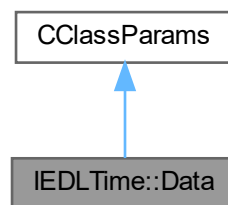
- [iedltempstore.h](#)

7.135 IEDLTime::Data Class Reference

Initialization data.

```
#include <edltime.h>
```

Inheritance diagram for IEDLTime::Data:



7.135.1 Detailed Description

Initialization data.

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

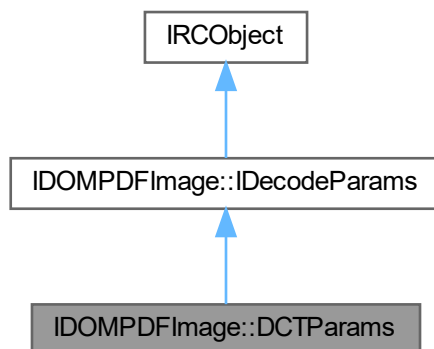
- [edltime.h](#)

7.136 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.

7.136.1 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

7.137 EDLIFStream Class Reference

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

```
#include <edlstream.h>
```

Inherits std::ifstream.

7.137.1 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

7.138 EDLOFStream Class Reference

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

```
#include <edlstream.h>
```

Inherits std::ofstream.

7.138.1 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

7.139 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.

7.139.1 Detailed Description

Implementation of qualified name class.

7.139.2 Constructor & Destructor Documentation

EDLQName() [1/2]

```
EDLQName::EDLQName (
    IEDLNamespacePtr ptrNamespace,
    const EDLString & name ) [inline]
```

constructor of [EDLQName](#)

Parameters

<i>ptrNamespace</i>	Smart pointer to the namespace interface
<i>name</i>	The name value

EDLQName() [2/2]

```
EDLQName::EDLQName (
    IEDLNamespacePtr ptrNamespace,
    const EDLSysString & name ) [inline]
```

constructor of [EDLQName](#)

Parameters

<i>ptrNamespace</i>	Smart pointer to the namespace interface
<i>name</i>	The name value

7.139.3 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

<i>another</i>	EDLQName
----------------	--------------------------

operator==()

```
bool EDLQName::operator== (
    const EDLQName & another ) const [inline]
```

operator ==

Parameters

<i>another</i>	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

<i>name</i>	The name value
-------------	----------------

Returns

bool True

setName() [2/2]

```
bool EDLQName::setName (  
    const EDLSysString & sysname ) [inline]
```

Sets member name of [EDLQName](#).

Parameters

<i>sysname</i>	The name value
----------------	----------------

Returns

bool True

setNamespace()

```
bool EDLQName::setNamespace (  
    const IEDLNamespacePtr & ns ) [inline]
```

Sets namespace.

Parameters

<i>ns</i>	Smart pointer to the namespace interface
-----------	--

Returns

bool True

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

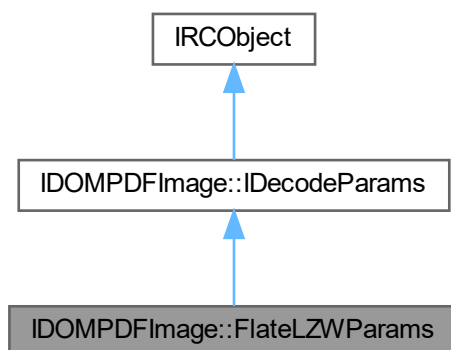
- [edlqname.h](#)

7.140 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.

7.140.1 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:

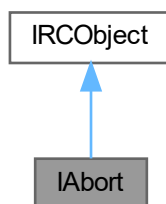
- `idomimageresource.h`

7.141 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.

7.141.1 Detailed Description

A simple class to signal an abort in processing.

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

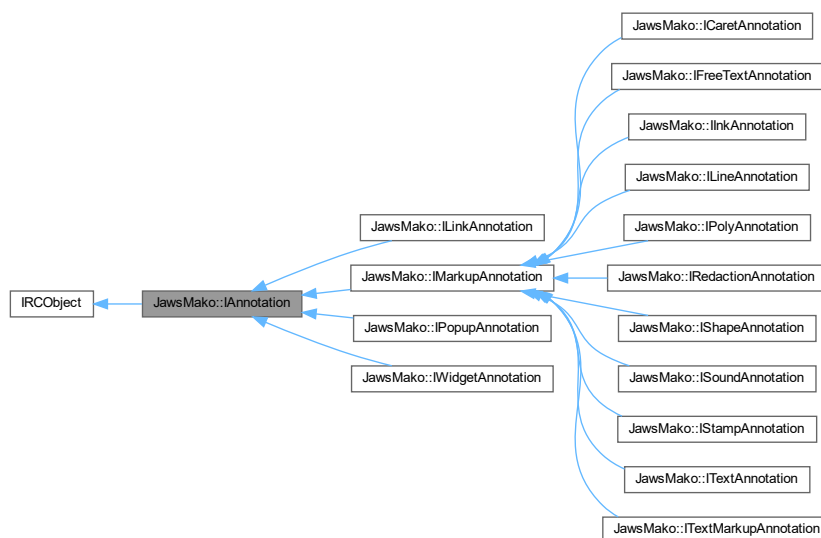
- iabort.h

7.142 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 [IRCOject](#)

- virtual void [addRef](#) () const =0
Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool [decRef](#) () const =0
Decreases the reference count of the actual object pointed to. When the reference count falls to 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 [IRCOject](#)

- virtual [~IRCOject](#) ()
Virtual destructor.

7.142.1 Detailed Description

An interface class for an annotation.

7.142.2 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.

7.142.3 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

<i>appearance</i>	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

<i>usage</i>	The annotation usage
<i>state</i>	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

<i>reference</i>	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

<i>rotate</i>	The angle to rotate. Must be 0, 90, 180 or 270 degrees.
<i>pageWidth</i>	The width of the parent page.
<i>pageHeight</i>	The height of the parent page.

setBorder()

```
virtual void JawsMako::IAnnotation::setBorder (
    const CAnnotationBorder & border ) [pure virtual]
```

Set the annotation's border.

Parameters

<i>border</i>	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

<i>color</i>	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.
--------------	--

setContentts()

```
virtual void JawsMako::IAnnotation::setContentts (
    const U8String & contents ) [pure virtual]
```

Set the Contents entry in UTF-8.

Parameters

<i>contents</i>	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

<i>flags</i>	The annotation flags
--------------	----------------------

setModificationTime()

```
virtual void JawsMako::IAnnotation::setModificationTime (
    const IEDLTimePtr & modificationTime ) [pure virtual]
```

Set the Modification date and time of the annotation.

Parameters

<i>modificationTime</i>	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

<code>rect</code>	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

<code>state</code>	The annotation state, or an empty string if the state should be removed
--------------------	---

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

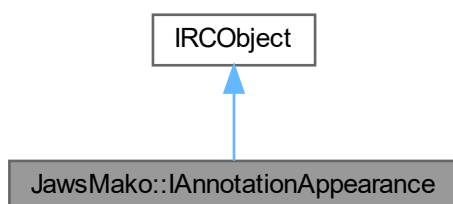
- [interactive.h](#)

7.143 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 [IRCOObject](#)

- virtual void [addRef](#) () const =0

Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.

- virtual bool [decRef](#) () const =0

Decreases the reference count of the actual object pointed to. When the reference count falls to 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 [IRCOObject](#)

- virtual [~IRCOObject](#) ()

Virtual destructor.

7.143.1 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.

7.143.2 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

<i>jawsMako</i>	The JawsMako instance
<i>form</i>	The graphical contents as an IDOMForm
<i>usage</i>	The usage for this appearance
<i>state</i>	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

<i>annotRect</i>	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:

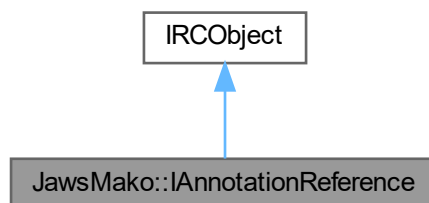
- [interactive.h](#)

7.144 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.

7.144.1 Detailed Description

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

7.144.2 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

<i>other</i>	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](#)

7.145 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.

7.145.1 Detailed Description

Utility class for dealing with XML representations of annotations and forms.

7.145.2 Member Function Documentation

createAnnotationReferenceFromTag()

```
static JAWSMako_API IAnnotationReferencePtr JawsMako::IAnnotationUtils::createAnnotationReferenceFromTag (
    const U8String & tag ) [static]
```

Generate an [IAnnotationReference](#) from an Annotation Tag attribute found in the XML.

Parameters

<i>tag</i>	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

<i>jawsMako</i>	The JawsMako instance
<i>document</i>	The document to generate the XML stream for
<i>resources</i>	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:

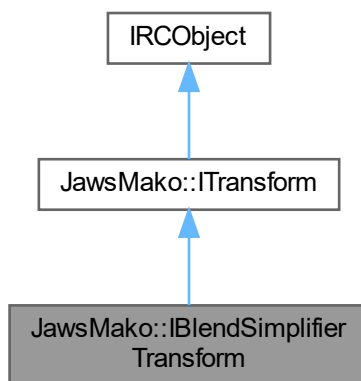
- [interactive.h](#)

7.146 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 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 transformAnnotations=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 [IRCOObject](#)

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

Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.

- virtual bool **decRef** () const =0

Decreases the reference count of the actual object pointed to. When the reference count falls to 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 [IRCOObject](#)

- virtual \sim [IRCOObject](#) ()

Virtual destructor.

7.146.1 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.

7.146.2 Member Function Documentation

create()

```
static JAWSMAKO_API IBlendSimplifierTransformPtr JawsMako::IBlendSimplifierTransform::create (
    const IJawsMakoPtr & jawsMako,
    const IAbortPtr & abort = IAbortPtr() ) [static]
```

Create the transform.

Parameters

<i>jawsMako</i>	The JawsMako instance.
<i>abort</i>	An abort callback handler.

Returns

The new instance.

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

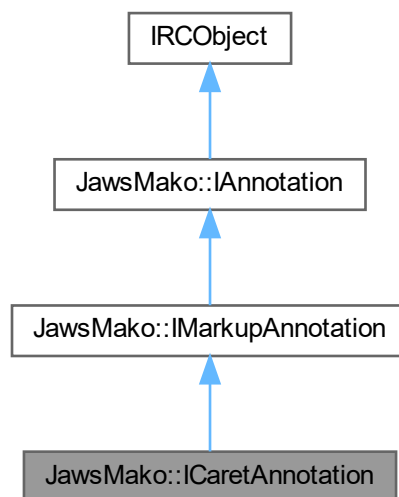
- [transforms.h](#)

7.147 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 **IRCOject**

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

7.147.1 Detailed Description

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

7.147.2 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:

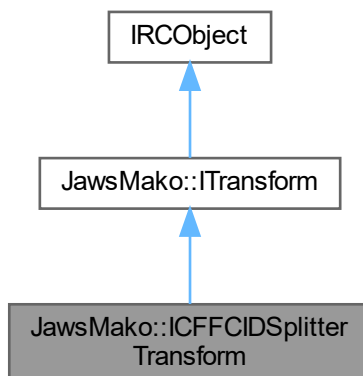
- [interactive.h](#)

7.148 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 ICFIDSplitterTransformPtr **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 transformAnnotations=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 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.

7.148.2 Member Function Documentation

create()

```
static JAWSMAKO_API ICFFCIDSplitterTransformPtr JawsMako::ICFFCIDSplitterTransform::create (
    const IJawsMakoPtr & jawsMako,
    const IAbortPtr & abort = IAbortPtr() ) [static]
```

Create the transform.

Parameters

<i>jawsMako</i>	The JawsMako instance.
<i>abort</i>	An abort callback handler.

Returns

The new instance.

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

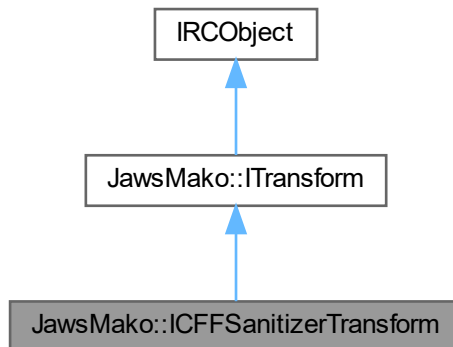
- [transforms.h](#)

7.149 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 transformAnnotations=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.

7.149.1 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 viewers in XPS, SVG, etc.

7.149.2 Member Function Documentation**create()**

```
static JAWSMAKO_API ICFFSanitizerTransformPtr JawsMako::ICFFSanitizerTransform::create (
    const IJawsMakoPtr & jawsMako,
    const IAbortPtr & abort = IAbortPtr() ) [static]
```

Create the transform.

Parameters

<i>jawsMako</i>	The JawsMako instance.
<i>abort</i>	An abort callback handler.

Returns

The new instance.

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

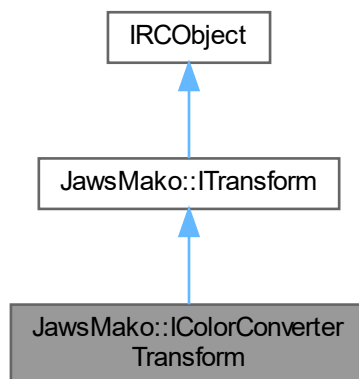
- [transforms.h](#)

7.150 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 `setShouldConvertForSpaceType(IDOMColorSpace::eICCBased, true)` however will cause all possibilities to be true. Likewise calling `setShouldConvertForSpaceType(IDOMColorSpace::eICCBased, false)` will cause all possibilities to be false.
- virtual void **setConvertICCSpacesWithProfileVersionGreaterThan** (uint8 majorVersion, uint8 minorVersion)=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 `IDOMColorSpaceDeviceN::CColorantInfoVect &spotsToMerge`=`IDOMColorSpaceDeviceN::CColorantInfoVect()`, const `CSpotColorNames &spotsToRetain`=`CSpotColorNames()`, const `CSpotColorNames &spotsToDrop`=`CSpotColorNames()`, 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.

7.150.1 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 [IColorComplexColorSimplifierTransform](#) can be applied after this transform if required.

7.150.2 Member Function Documentation

create()

```
static JAWSMAKO_API IColorConverterTransformPtr JawsMako::IColorConverterTransform::create (
    const IJawsMakoPtr & jawsMako,
    const IAbortPtr & abort = IAbortPtr() ) [static]
```

Create the transform.

Parameters

<i>jawsMako</i>	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 (IDMSolidColorBrush) 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

<i>preserveForText</i>	Whether or not to apply black preservation for text.
<i>preserveForOther</i>	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

<i>convertProcess</i>	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.
<i>mergeSpots</i>	If false, no spots will be merged (and #spotsToMerge is ignored).
<i>spotsToMerge</i>	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.
<i>spotsToRetain</i>	The names of colorants to retain even if they are subject to mergeSpots and/or spotsToMerge.
<i>spotsToDrop</i>	The names of colorants to drop entirely. If not provided, no spot components will be dropped.
<i>knockoutDroppedSpots</i>	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

<i>profile</i>	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

<i>targetSpace</i>	The target color space to be used.
--------------------	------------------------------------

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

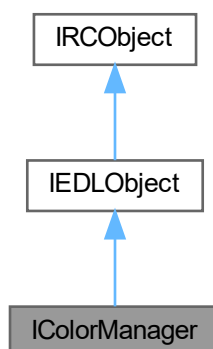
- [transforms.h](#)

7.151 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 *inColors, 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 *inColors, 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 profile used for the given ICC color space.
- virtual uint32 [getProfileClassForICCBasedSpace](#) (const IDOMColorSpaceICCBasedPtr &space)=0
Get the profile class of the profile used for the given ICC color space.
- virtual uint32 [getProfileConnectionSpaceForICCBasedSpace](#) (const IDOMColorSpaceICCBasedPtr &space)=0
Get the profile connection space (PCS) type for the profile used for the given ICC color space.
- virtual bool [getProfileHasSupportForIntentForICCBasedSpace](#) (const IDOMColorSpaceICCBasedPtr &space, [eRenderingIntent](#) intent, bool input)=0
Determine if the profile used for the given ICC color space has support for the given intent for the given usage. Matrix shaper style profiles support every intent. LUT-based profiles may have explicit support for some intents.

- 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.

7.151.1 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.

7.151.2 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

<i>numColors</i>	The number of colors to convert.
<i>hasAlpha</i>	Set to true if the color data contains alpha.
<i>sourceSpace</i>	The color space describing the input samples.
<i>destSpace</i>	The desired color space for the output samples. May be NULL if destSpace is a Device Link profile.
<i>intent</i>	The desired rendering intent for the conversion.
<i>bpc</i>	The desired handling of black point compensation. If in doubt, use eBPCDefault.
<i>inColors</i>	Pointer to the input sample buffer.
<i>outColors</i>	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

<i>numColors</i>	The number of colors to convert.
<i>hasAlpha</i>	Set to true if the color data contains alpha.
<i>sourceSpace</i>	The color space describing the input samples.
<i>destSpace</i>	The desired color space for the output samples. May be NULL if destSpace is a Device Link profile.
<i>intent</i>	The desired rendering intent for the conversion.
<i>bpc</i>	The desired handling of black point compensation. If in doubt, use eBPCDefault.
<i>inColors</i>	Pointer to the input sample buffer.
<i>outColors</i>	Pointer to the output sample buffer.
<i>inBPS</i>	The bits per sample of the input samples. Currently 1, 2, 4, 8, 12 and 16 bits are supported.
<i>outBPS</i>	The desired bits per sample of the output samples. Currently 1, 2, 4, 8, 12 and 16 bits are supported.

createCalibratedGrayProfile()

```
virtual IDOMICCPProfilePtr 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

<i>whitePoint</i>	A three entry vector describing the white point of the color space, in the CIE XYZ color space.
<i>gamma</i>	Optional. The gamma for the color space, 1.0 if not provided
<i>blackPoint</i>	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

IDOMICCPProfilePtr A smart pointer to the resulting profile.

createCalibratedRGBProfile()

```
virtual IDOMICCPProfilePtr 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

<i>whitePoint</i>	A three entry vector describing the white point of the color space, in the CIE XYZ color space.
<i>gamma</i>	Optional. A vector containing the desired gamma for the color space, one value for R, G and B respectively. 1 1 1 is used if not provided.
<i>matrix</i>	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.
<i>blackPoint</i>	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

<i>whitePoint</i>	A three entry vector describing the white point of the color space, in the CIE XYZ color space.
<i>gamma</i>	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).
<i>blackPoint</i>	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

<i>whitePoint</i>	A three entry vector describing the white point of the color space, in the CIE XYZ color space.
<i>blackPoint</i>	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

<i>sourceSpace</i>	The desired input color space.
<i>destSpace</i>	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

<i>sourceProfile</i>	The desired input color profile.
<i>destProfile</i>	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

<i>pFactory</i>	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

eBlackPointCompensation 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

<i>space</i>	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

<i>index</i>	The index of DeviceLink ICC profile which has already been set.
<i>sourceSpace</i>	The input color space associated with the DeviceLink ICC profile.
<i>destSpace</i>	The output color space associated with the DeviceLink ICC profile.
<i>deviceLink</i>	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

<i>csaMemory</i>	A reference to a simple buffer to store the given csa.
<i>space</i>	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

<i>space</i>	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

<i>profile</i>	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

<i>csaMemory</i>	A reference to a simple buffer to store the generated csa.
<i>space</i>	The ICC Based color space to process.
<i>intent</i>	The intent for which the CSA should be generated.

getProfileClassForICCBasedSpace()

```
virtual uint32 IColorManager::getProfileClassForICCBasedSpace (
    const IDOMColorSpaceICCBasedPtr & space ) [pure virtual]
```

Get the profile class of the profile used for the given ICC color space.

Parameters

<i>space</i>	The ICC color space to interrogate.
--------------	-------------------------------------

Returns

uint32 The ICC profile signature, using the enumeration as per the ICC specification.

getProfileColorSpaceForICCBasedSpace()

```
virtual uint32 IColorManager::getProfileColorSpaceForICCBasedSpace (
    const IDOMColorSpaceICCBasedPtr & space ) [pure virtual]
```

Get the color space type for an ICCBased color space.

Parameters

<i>space</i>	the ICC space to interrogate
--------------	------------------------------

Returns

uint32 The ICC color space signature, using the enumeration as per the ICC specification

getProfileConnectionSpaceForICCBasedSpace()

```
virtual uint32 IColorManager::getProfileConnectionSpaceForICCBasedSpace (
    const IDOMColorSpaceICCBasedPtr & space ) [pure virtual]
```

Get the profile connection space (PCS) type for the profile used for the given ICC color space.

Parameters

<i>space</i>	The ICC color space to interrogate. For device link profiles this will be the canonical output space.
--------------	---

Returns

uint32 The ICC PCS 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

<i>space</i>	The color space to get the profile for.
--------------	---

Returns

IDOMICCProfilePtr A smart pointer to the profile, if applicable, otherwise NULL.

getProfileHasSupportForIntentForICCBasedSpace()

```
virtual bool IColorManager::getProfileHasSupportForIntentForICCBasedSpace (
    const IDOMColorSpaceICCBasedPtr & space,
    eRenderingIntent intent,
    bool input ) [pure virtual]
```

Determine if the profile used for the given ICC color space has support for the given intent for the given usage. Matrix shaper style profiles support every intent. LUT-based profiles may have explicit support for some intents.

Parameters

<i>space</i>	The ICC color space to interrogate.
<i>intent</i>	The intent to query.
<i>input</i>	Set to true if querying for use as an input, false if querying for use as an output.

Returns

bool True if the given intent is supported by the profile for the given direction.

getProfileNameForICCBasedSpace()

```
virtual void IColorManager::getProfileNameForICCBasedSpace (
    const IDOMColorSpaceICCBasedPtr & space,
    EDLSysString & name ) [pure virtual]
```

Get the name of the profile for an ICCBased color space.

Parameters

<i>space</i>	The ICC color space to interrogate.
<i>name</i>	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 profile used for the given ICC color space.

Parameters

<i>space</i>	The ICC color space to interrogate.
<i>majorVersion</i>	Reference to receive the major version
<i>minorVersion</i>	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

<i>space</i>	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 will not be used.

Parameters

<i>defaultBpc</i>	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

<i>space</i>	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

<i>space</i>	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

<i>space</i>	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

<i>mapGrayDirectly</i>	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

<i>numColors</i>	The number of colors to test.
<i>sourceSpace</i>	The color space describing the input samples.
<i>destSpace</i>	The desired destination color space.
<i>intent</i>	The desired rendering intent for the test.
<i>bpc</i>	The desired handling of black point compensation. If in doubt, use eBPCDefault.
<i>inColors</i>	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

<i>sourceSpace</i>	The desired input color space.
<i>destSpace</i>	The desired output color space.
<i>deviceLink</i>	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

<i>sourceProfile</i>	The desired input color profile.
<i>destProfile</i>	The desired output color profile.
<i>deviceLink</i>	The desired DeviceLink ICC profile.

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

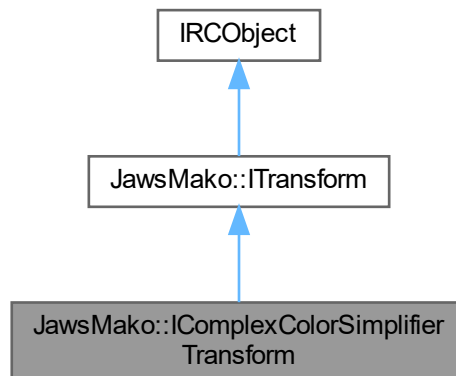
- icolormanager.h

7.152 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 transformAnnotations=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 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.

7.152.2 Member Function Documentation

create()

```
static JAWSMako_API IComplexColorSimplifierTransformPtr JawsMako::IComplexColorSimplifier↔
Transform::create (
    const IJawsMakoPtr & jawsMako,
    const IAbortPtr & abort = IAbortPtr() ) [static]
```

Create the transform.

Parameters

<i>jawsMako</i>	The JawsMako instance.
<i>abort</i>	An abort callback handler.

Returns

The new instance.

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

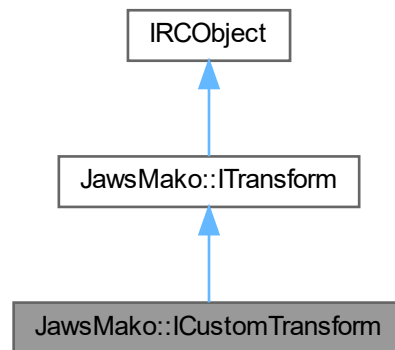
- [transforms.h](#)

7.153 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 dependsOnGroupSpace=true, bool dependsOnRenderingIntent=true, bool dependsOnTransform=true, bool dependsOnBrushUsage=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 transformAnnotations=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 [IRCOject](#)

- virtual void [addRef](#) () const =0
Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool [decRef](#) () const =0
Decreases the reference count of the actual object pointed to. When the reference count falls to 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 [IRCOject](#)

- virtual [~IRCOject](#) ()
Virtual destructor.

7.153.1 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::Implementation](#). 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 [Implementation](#), 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` (`IDOMImageBrush`) 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:

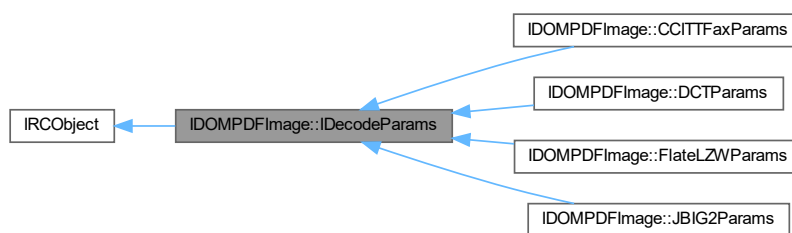
- [customtransform.h](#)

7.154 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.

7.154.1 Detailed Description

Abstract interface for per-image decoding filter parameters.

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

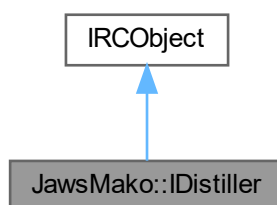
- `idomimageresource.h`

7.155 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` ¶m, 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 `setMonochromeImageCompression` (`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 `setMonochromeImageDownsampling` (`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 an 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 &font)=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](#) ([IPSCCommentMonitor](#) *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 [IRCOObject](#)

- virtual void [addRef](#) () const =0
Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool [decRef](#) () const =0
Decreases the reference count of the actual object pointed to. When the reference count falls to 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 [IRCOObject](#)

- virtual [~IRCOObject](#) ()
Virtual destructor.

7.155.1 Detailed Description

An instance of the JawsMako Distiller class.

7.155.2 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 <code>setColorQFactor()</code>

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.

7.155.3 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

<i>font</i>	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

<i>fonts</i>	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

<i>jawsMako</i>	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

<i>inputStream</i>	The input PostScript stream.
<i>outputStream</i>	The output stream to write the PDF to.
<i>progressMonitor</i>	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

<i>font</i>	The input font file name.
<i>names</i>	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

<i>fontName</i>	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

<i>ascii85EncodePages</i>	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

<i>compression</i>	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

<i>method</i>	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

<i>resolution</i>	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

<i>quality</i>	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

<i>qfactor</i>	The QFactor to use when compressing with eJQUser.
----------------	---

setCommentMonitor()

```
virtual void JawsMako::IDistiller::setCommentMonitor (
    IPSCCommentMonitor * commentMonitor ) [pure virtual]
```

Set a comment monitor.

Sets a [IPSCCommentMonitor](#) 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

<i>commentMonitor</i>	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 999999999 (~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

<i>compressObjects</i>	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

<i>compressPages</i>	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

<i>convert</i>	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

<i>cropToBBox</i>	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

<i>defaultPanoseStyle</i>	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

<i>embed</i>	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

<i>embed</i>	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

<i>embed</i>	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

<i>ownerPassword</i>	The owner password to use.
<i>userPassword</i>	The user password to use.
<i>permissions</i>	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

<i>epilog</i>	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

<i>path</i>	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

<i>compression</i>	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

<i>method</i>	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

<i>resolution</i>	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

<i>quality</i>	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

<i>qfactor</i>	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

<i>halftone</i>	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

<i>hiResBBox</i>	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

<i>jobTicket</i>	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

<i>linearize</i>	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

<i>compression</i>	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

<i>method</i>	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

<i>resolution</i>	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

<i>opi</i>	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

<i>overprint</i>	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

<i>overprintMode</i>	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

<i>pageHigh</i>	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

<i>pageLow</i>	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

<i>pageLow</i>	The lower page limit.
<i>pageHigh</i>	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

<i>panose</i>	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

<i>param</i>	The parameter to change
<i>value</i>	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

<i>version</i>	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

<i>prolog</i>	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

<i>resolution</i>	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

<i>path</i>	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

<i>subset</i>	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

<i>thumbnails</i>	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

<i>transfers</i>	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

<i>trimToBBox</i>	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

<code>useDeviceDependentColor</code>	Use convert?
--------------------------------------	--------------

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

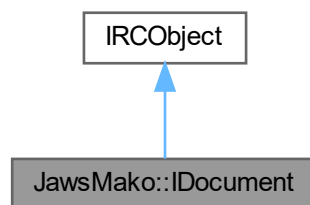
- [distiller.h](#)

7.156 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, an exception will result.*
 - virtual void **findTarget** (DOMid targetId, uint32 &pageNum)=0
 - Find the page containing 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 it 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 it 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.

7.156.1 Detailed Description

A document from an [IDocumentAssembly](#), allowing for high level document and page management, and providing on-demand lazy loading of page markup.

7.156.2 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

<i>embeddedData</i>	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

<i>namedDestination</i>	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

<i>page</i>	smart pointer to the page to be inserted.
<i>sourceDocument</i>	(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

<i>jawsMako</i>	The JawsMako instance.
<i>progressMonitor</i>	A smart pointer to an IPProgressMonitor 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

<i>page</i>	smart pointer to the page to be inserted.
<i>index</i>	The position in the document to insert the page, 0 being the first position.
<i>sourceDocument</i>	(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

<i>farReference</i>	The far reference to resolve.
<i>store</i>	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

<i>index</i>	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

<i>reference</i>	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

<i>index</i>	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

<i>namedDestinations</i>	The vector of named destinations
--------------------------	----------------------------------

setOutline()

```
virtual void JawsMako::IDocument::setOutline (
    const IDOMOutlinePtr & outline ) [pure virtual]
```

Set the document outline, if present.

Parameters

<i>outline</i>	The desired document outline, or NULL to set no outline.
----------------	--

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

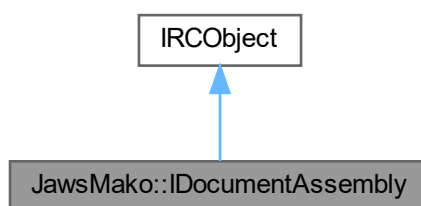
- [jawsmake.h](#)

7.157 JawsMako::IDocumentAssembly Class Reference

A self contained collection of IDocuments.

```
#include <jawsmake.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 containing 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. Currently 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 IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())

Create a new, empty document assembly.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual **~IRCObject** ()

Virtual destructor.

7.157.1 Detailed Description

A self contained collection of IDocuments.

7.157.2 Member Function Documentation

appendDocument()

```
virtual void JawsMako::IDocumentAssembly::appendDocument (
    const IDocumentPtr & document ) [pure virtual]
```

Append a document to the assembly.

Parameters

<i>document</i>	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

<i>jawsMako</i>	The JawsMako instance.
<i>progressMonitor</i>	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

<i>index</i>	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

<i>document</i>	smart pointer to the document to be inserted.
<i>index</i>	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

<i>index</i>	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

<i>thumbnail</i>	Smart pointer to new thumbnail image.
------------------	---------------------------------------

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

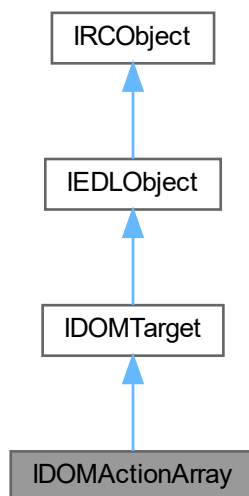
- [jawsmao.h](#)

7.158 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 `getTargetType` 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.

7.158.1 Detailed Description

[IDOMActionArray](#) interface.

7.158.2 Member Function Documentation

addAction()

```
virtual bool IDOMActionArray::addAction (  
    const IDOMTargetPtr & ptrAction ) [pure virtual]
```

Append a action to the action collection.

Parameters

<i>ptrAction</i>	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:

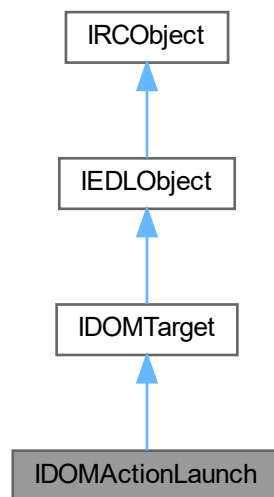
- [idomtarget.h](#)

7.159 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 ¶ms)=0
Sets the Windows-specific launch parameters.
- virtual EDLString [getMacParameters](#) () const =0
Retrieves the Mac OS-specific launch parameters.
- virtual void [setMacParameters](#) (const EDLString ¶ms)=0
Sets the Mac OS-specific launch parameters.
- virtual EDLString [getUnixParameters](#) () const =0
Retrieves the UNIX-specific launch parameters.
- virtual void [setUnixParameters](#) (const EDLString ¶ms)=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 [getTargetType](#) 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.

7.159.1 Detailed Description

`IDOMActionLaunch` interface.

7.159.2 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 `getTargetType` 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

<i>file</i>	File specification
-------------	--------------------

setMacParameters()

```
virtual void IDOMActionLaunch::setMacParameters (
    const EDLString & params ) [pure virtual]
```

Sets the Mac OS-specific launch parameters.

Parameters

<i>params</i>	The launch parameters
---------------	-----------------------

setNewWindow()

```
virtual void IDOMActionLaunch::setNewWindow (
    bool bNewWindow ) [pure virtual]
```

Setter for NewWindow flag.

Parameters

<i>bNewWindow</i>	New value of the NewWindow flag
-------------------	---------------------------------

setUnixParameters()

```
virtual void IDOMActionLaunch::setUnixParameters (
    const EDLString & params ) [pure virtual]
```

Sets the UNIX-specific launch parameters.

Parameters

<i>params</i>	The launch parameters
---------------	-----------------------

setWinParameters()

```
virtual void IDOMActionLaunch::setWinParameters (
    const EDLString & params ) [pure virtual]
```

Sets the Windows-specific launch parameters.

Parameters

<i>params</i>	The launch parameters
---------------	-----------------------

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

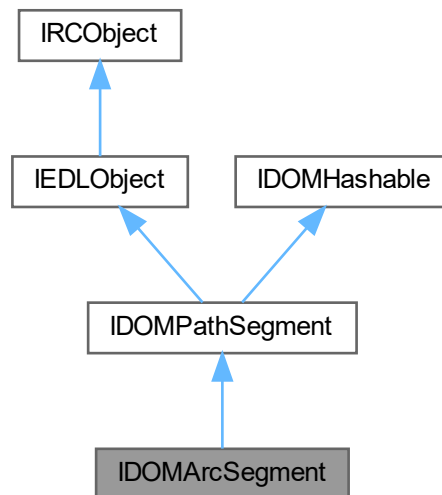
- idomtarget.h

7.160 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.

7.160.1 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.

7.160.2 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

<i>factory</i>	Pointer to the class factory to be used.
<i>startPoint</i>	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

<i>factory</i>	The factory to use.
<i>point</i>	The end point of the arc.
<i>radiusX</i>	The desired x radius of the arc.
<i>radiusY</i>	The desired y radius of the arc.
<i>rotationAngle</i>	The rotation angle.
<i>isLargeArc</i>	Whether or not the large arc should be used.
<i>isStroked</i>	Should the segment be stroked, if used in a stroking path.
<i>sweepDirection</i>	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

<i>isLA</i>	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

<i>pt</i>	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

<i>radiusX</i>	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

<i>radiusY</i>	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

<i>rA</i>	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

<i>sd</i>	eSweepDirection The direction in which the arc is drawn from the start point.
-----------	---

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

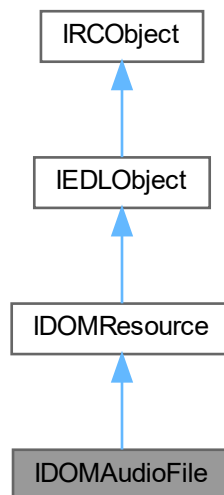
- idompathgeometry.h

7.161 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 `InputStreamPtr getStream () const =0`
Retrieves the resource stream.
- virtual void `setStream (const InputStreamPtr &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.

7.161.1 Detailed Description

[IDOMAudioFile](#) interface.

7.161.2 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:

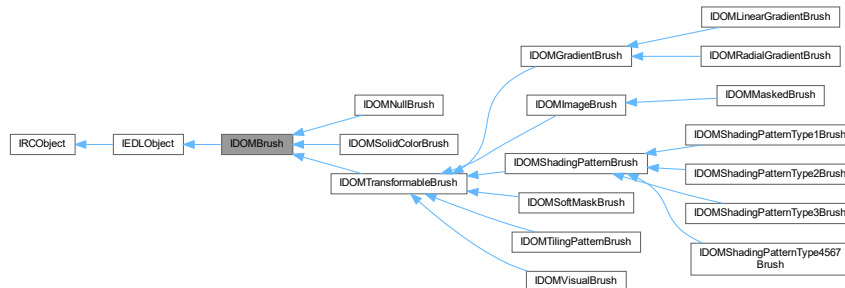
- [idomresources.h](#)

7.162 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.

7.162.1 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 multiplicative way with the opacity value.

Note

Overprint information is available via APIs in [IDOMPathNode](#) and [IDOMGlyphs](#) interfaces.

7.162.2 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

<i>pFactory</i>	The factory to use.
<i>nodeTransform</i>	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

<i>opc</i>	The opacity value
------------	-------------------

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

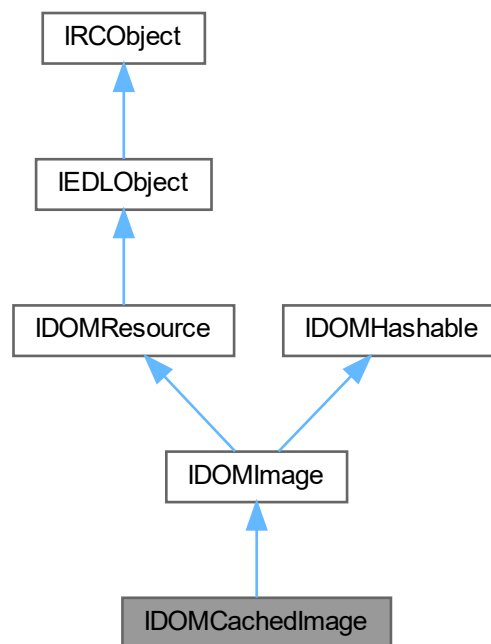
- [idombrush.h](#)

7.163 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 [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 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 &sourceImage, 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.

7.163.1 Detailed Description

Interface to a class that when overlaid 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.

7.163.2 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

<i>pFactory</i>	The EDL Class Factory.
<i>sourceImage</i>	The image to use.
<i>cacheSamples</i>	Set to true if image samples should be cached in addition to metadata. Note that unless <i>forceCache</i> 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.
<i>alwaysCache</i>	Set this (and <i>cacheSamples</i>) to true if samples should always be cached, subject to cache size limits. Consulted only if <i>alwaysCache</i> 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

<i>stream</i>	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

<i>stream</i>	A smart pointer to the stream
---------------	-------------------------------

Implements [IDOMResource](#).

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

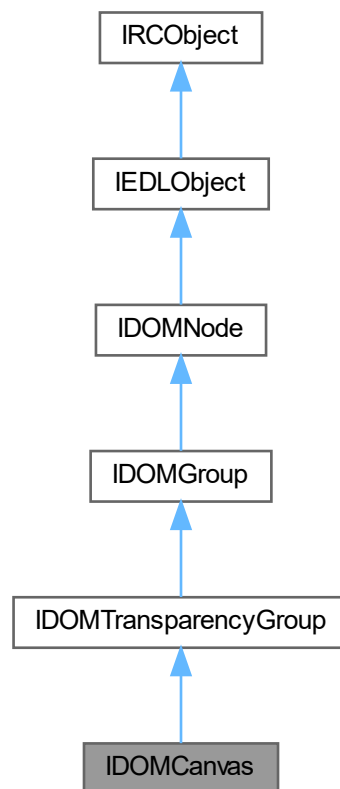
- idomimageresource.h

7.164 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.*
 - virtual [eRenderingIntent getRenderingIntent](#) () const =0
 - Get the rendering intent for the node if present. The rendering intent is used whenever the node requires color conversion to a different color space. Set eRenderingIntentUnset to unset the rendering intent, in which a default will be chosen based on the color space in use. This setting is honored only for rendering, color conversion, PDF or PostScript output.*
 - virtual void [setRenderingIntent](#) ([eRenderingIntent](#) intent)=0
 - Set or clear the rendering intent for the node. The rendering intent is used whenever the node requires color conversion to a different color space. Set eRenderingIntentUnset to unset the rendering intent, in which a default will be chosen based on the color space in use. This setting is honored only for rendering, color conversion, PDF or PostScript output.*
 - virtual [eBlackPointCompensation getBlackPointCompensation](#) () const =0
 - Get the black point compensation for the node. The black point compensation setting is used whenever the node requires color conversion to a different color space and affects how dark regions are handled. This setting is honored only for rendering, color conversion, PDF or PostScript output.*
 - virtual void [setBlackPointCompensation](#) ([eBlackPointCompensation](#) compensation)=0
 - Set the black point compensation for the node. The black point compensation setting is used whenever the node requires color conversion to a different color space and affects how dark regions are handled. This setting is honored only for rendering, color conversion, PDF or PostScript output.*

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 (getNodeTypeId) 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 (getNodeTypeId) 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

Simplified 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 descendIntoForms=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 [IRCOobject](#)

- virtual void [addRef](#) () const =0

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

Decreases the reference count of the actual object pointed to. When the reference count falls to 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.

7.164.1 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.

7.164.2 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

<i>helptext</i>	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

<i>propname</i>	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

<i>em</i>	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

<i>lang</i>	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

<i>target</i>	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

<i>ptrResourceDictionary</i>	Smart pointer to the resource dictionary
------------------------------	--

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

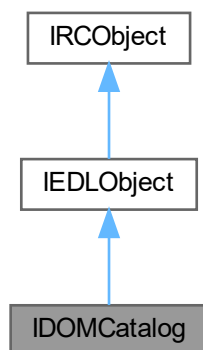
- idomcanvas.h

7.165 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 pair 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.

7.165.1 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.

7.165.2 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

<i>uri</i>	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

<i>id1</i>	A numeric identifier.
<i>id2</i>	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

<i>index</i>	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

<i>id1</i>	A numeric identifier.
<i>id2</i>	A second numeric identifier (optional. Presently not used, and set to zero.)
<i>createIfNotExist</i>	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

<i>uri</i>	URI of the resource whose DOM ID is required.
<i>createIfNotExist</i>	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

<i>id</i>	The DOM ID of the IEDLObject to retrieve.
<i>ptrObject</i>	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

<i>id</i>	DOM ID of the resource
<i>uri</i>	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 pair in the catalog, under the ID specified by id.

Parameters

<i>id</i>	The ID to register the object under.
<i>id1</i>	A numeric identifier.
<i>id2</i>	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

<i>id</i>	The ID to register the object under.
<i>ptrObject</i>	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

<i>id</i>	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:

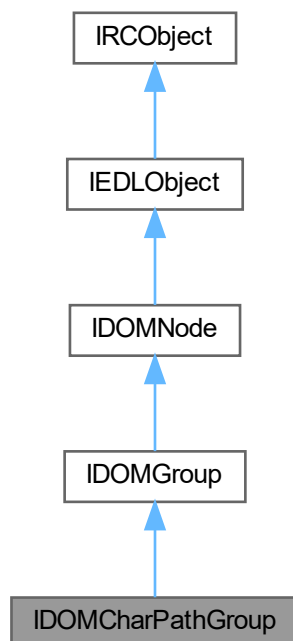
- idomcatalog.h

7.166 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
- Simplified 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 descendIntoForms=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 [IRCOject](#)

- virtual void [addRef](#) () const =0

Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.

- virtual bool [decRef](#) () const =0

Decreases the reference count of the actual object pointed to. When the reference count falls to 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.

7.166.1 Detailed Description

[IDOMCharPathGroup](#) interface Interface to DOM node representing Group Elements that consist of stroked text.

7.166.2 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

<i>charPathType</i>	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

<i>clippedGroup</i>	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

<i>strokePath</i>	Smart pointer to stroke path of the charpath
-------------------	--

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

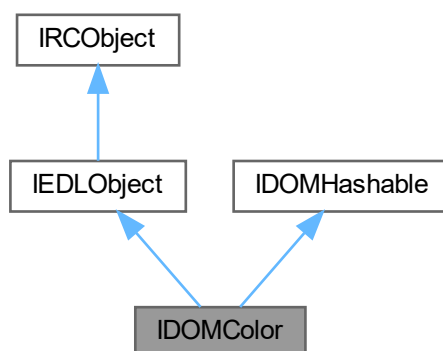
- idomcharpathgroup.h

7.167 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 colors. 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 colors. 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 colors. 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.

7.167.1 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.

7.167.2 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

<i>pFactory</i>	The EDL Class factory to use.
<i>space</i>	The color space to use. This must take one color component.
<i>alpha</i>	The alpha to use.
<i>component</i>	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

<i>pFactory</i>	The EDL Class factory to use.
<i>space</i>	The color space to use. This must take two color components.
<i>alpha</i>	The alpha to use.
<i>component1</i>	The first color components to use.
<i>component2</i>	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

<i>pFactory</i>	The EDL Class factory to use.
-----------------	-------------------------------

Parameters

<i>space</i>	The color space to use. This must take three color components.
<i>alpha</i>	The alpha to use.
<i>component1</i>	The first color components to use.
<i>component2</i>	The second color components to use.
<i>component3</i>	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

<i>pFactory</i>	The EDL Class factory to use.
<i>space</i>	The color space to use. This must take four color components.
<i>alpha</i>	The alpha to use.
<i>component1</i>	The first color components to use.
<i>component2</i>	The second color components to use.
<i>component3</i>	The third color components to use.
<i>component4</i>	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

<i>pFactory</i>	The EDL Class factory to use.
<i>space</i>	The color space to use. This must take five color components.
<i>alpha</i>	The alpha to use.
<i>component1</i>	The first color components to use.
<i>component2</i>	The second color components to use.
<i>component3</i>	The third color components to use.
<i>component4</i>	The fourth color components to use.
<i>component5</i>	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

<i>pFactory</i>	The EDL Class factory to use.
<i>space</i>	The color space to use. This must take six color components.
<i>alpha</i>	The alpha to use.
<i>component1</i>	The first color components to use.
<i>component2</i>	The second color components to use.
<i>component3</i>	The third color components to use.
<i>component4</i>	The fourth color components to use.
<i>component5</i>	The fifth color components to use.
<i>component6</i>	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

<i>pFactory</i>	The EDL Class factory to use.
<i>space</i>	The color space to use. This must take between 7 and 32 color components.
<i>alpha</i>	The alpha to use.
<i>component1</i>	The first color component to use.
<i>component2</i>	The second color component to use.
<i>component3</i>	The third color component to use.
<i>component4</i>	The fourth color component to use.
<i>component5</i>	The fifth color component to use.
<i>component6</i>	The sixth color component to use.
<i>component7</i>	The seventh color component to use.
<i>component8</i>	The eighth color component to use (if applicable).
<i>component9</i>	The ninth color component to use (if applicable).
<i>component10</i>	The tenth color component to use (if applicable).
<i>component11</i>	The eleventh color component to use (if applicable).
<i>component12</i>	The twelfth color component to use (if applicable).
<i>component13</i>	The thirteenth color component to use (if applicable).

Parameters

<i>component14</i>	The fourteenth color component to use (if applicable).
<i>component15</i>	The fifteenth color component to use (if applicable).
<i>component16</i>	The sixteenth color component to use (if applicable).
<i>component17</i>	The seventeenth color component to use (if applicable).
<i>component18</i>	The eighteenth color component to use (if applicable).
<i>component19</i>	The nineteenth color component to use (if applicable).
<i>component20</i>	The twentieth color component to use (if applicable).
<i>component21</i>	The twenty-first color component to use (if applicable).
<i>component22</i>	The twenty-second color component to use (if applicable).
<i>component23</i>	The twenty-third color component to use (if applicable).
<i>component24</i>	The twenty-fourth color component to use (if applicable).
<i>component25</i>	The twenty-fifth color component to use (if applicable).
<i>component26</i>	The twenty-sixth color component to use (if applicable).
<i>component27</i>	The twenty-seventh color component to use (if applicable).
<i>component28</i>	The twenty-eighth color component to use (if applicable).
<i>component29</i>	The twenty-ninth color component to use (if applicable).
<i>component30</i>	The thirtieth color component to use (if applicable).
<i>component31</i>	The thirty-first color component to use (if applicable).
<i>component32</i>	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

<i>pFactory</i>	The EDL Class factory to use.
<i>space</i>	The color space to use.
<i>alpha</i>	The alpha to use.
<i>components</i>	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

<i>pFactory</i>	The EDL Class factory to use.
<i>space</i>	The color space to use.
<i>alpha</i>	The alpha to use.
<i>components</i>	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 colors. Default parameters will yield an opaque DeviceCMYK white.

Parameters

<i>pFactory</i>	The EDL Class factory to use.
<i>c</i>	The cyan level to use.
<i>m</i>	The magenta level to use.
<i>y</i>	The yellow level to use.
<i>k</i>	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 colors. Default parameters will yield an opaque DeviceGray black.

Parameters

<i>pFactory</i>	The EDL Class factory to use.
<i>gray</i>	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 colors. Default parameters will yield an opaque DeviceRGB black.

Parameters

<i>pFactory</i>	The EDL Class factory to use.
<i>r</i>	The red level to use.
<i>g</i>	The green level to use.
<i>b</i>	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

<i>component</i>	Index of the component.
------------------	-------------------------

Returns

float Component value

setAlpha()

```
virtual void IDOMColor::setAlpha (
    float a ) [pure virtual]
```

Sets the alpha channel value.

Parameters

<i>a</i>	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 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

<i>colorSpace</i>	Smart pointer to the desired new color space
<i>setDefaultColor</i>	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, it 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 color will be given zero alpha and the default components for the destination color space will be set.

Parameters

<i>colorSpace</i>	Smart pointer to the desired new color space.
<i>intent</i>	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

<i>bpc</i>	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 <code>eBPCDefault</code> .
<i>pFactory</i>	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

<i>component</i>	Index of the component.
<i>value</i>	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, it will be converted to the Alternate or Underlying color space before color conversion takes place.

Parameters

<i>pFactory</i>	The EDL class factory.
<i>colorSpace</i>	Smart pointer to the color space to test against.
<i>intent</i>	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

<i>bpc</i>	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

7.168 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.

7.168.1 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.

7.168.2 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

<i>colorSpace</i>	The color space to compare with.
<i>exact</i>	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

<i>component</i>	The index of the component whose range is being queried
<i>low</i>	Reference parameter to receive the low bound of the component's value range.
<i>high</i>	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

<i>colorSpace</i>	The color space to compare with.
<i>intent</i>	The rendering intent to use.
<i>bpc</i>	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:

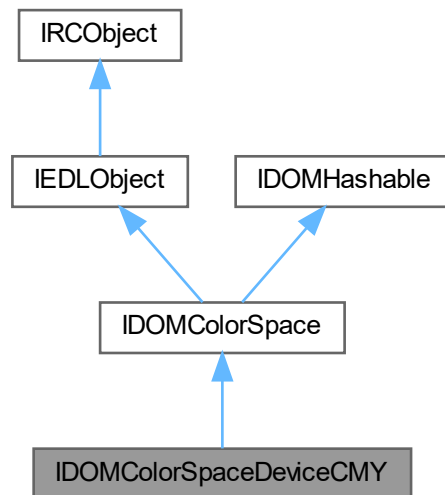
- idomcolorspace.h

7.169 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.

7.169.1 Detailed Description

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

7.169.2 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

<code>pFactory</code>	The EDL Class factory to use.
-----------------------	-------------------------------

Returns

[IDOMColorSpaceDeviceCMYPtr](#) The new color space.

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

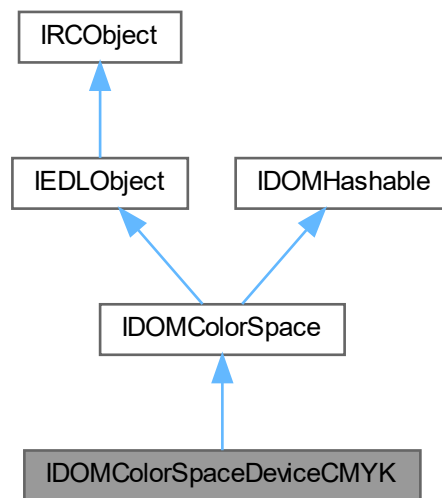
- `idomcolorspace.h`

7.170 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 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

Represents the default CMYK color space.

7.170.2 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

<i>pFactory</i>	The EDL Class factory to use.
-----------------	-------------------------------

Returns

[IDOMColorSpaceDeviceCMYKPtr](#) The new color space.

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

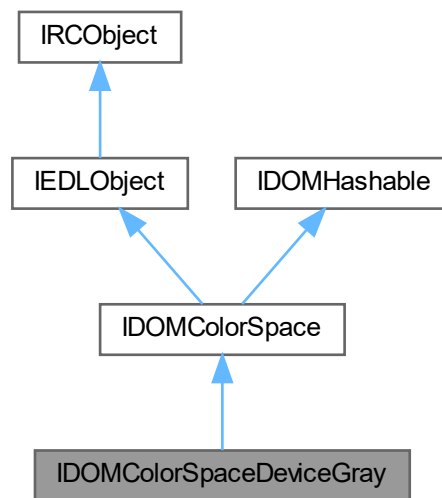
- [idomcolorspace.h](#)

7.171 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 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.

7.171.1 Detailed Description

[IDOMColorSpaceDeviceGray](#) interface.

7.171.2 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

<i>pFactory</i>	The EDL Class factory to use.
-----------------	-------------------------------

Returns

[IDOMColorSpaceDeviceGrayPtr](#) The new color space.

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

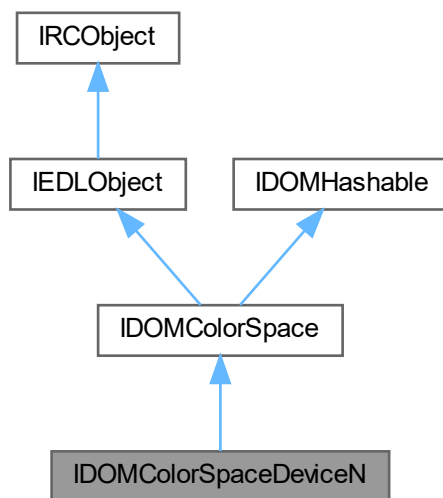
- [idomcolorspace.h](#)

7.172 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 &colorSpace=IDOMColorSpacePtr()) const =0

Convenience method to populate a CColorantInfoVect from the Device*N* 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 CEDLRawStringVect &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 CDeviceNColorantVect &colorants, const IDOMColorSpacePtr &alternate, const IDOMFunctionPtr &tintTransform, const IDOMColorSpacePtr &processColorSpace=IDOMColorSpacePtr(), const CEDLRawStringVect &processComponents=CEDLRawStringVect(), const CEDLRawStringVect &printingOrder=CEDLRawStringVect(), bool isNChannel=false)`
Create a DeviceN Color Space. Throws an `IEDLError` on failure.
- static `EDL_API IDOMColorSpaceDeviceNPtr create (IEDLClassFactory *pFactory, const CColorantInfoVect &colorants, const IDOMColorSpacePtr &alternate, const IDOMColorSpacePtr &processColorSpace=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.

7.172.1 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

7.172.2 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

<i>pFactory</i>	The EDL Class factory to use.
<i>colorants</i>	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.
<i>alternate</i>	The alternate color space.
<i>processColorSpace</i>	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

<i>pFactory</i>	The EDL Class factory to use.
<i>colorants</i>	A vector of smart pointers to IDOMDeviceNColorant objects
<i>alternate</i>	The alternate color space.
<i>tintTransform</i>	A function that maps colors in the space to the alternate color space.
<i>processColorSpace</i>	The PDF process color space associated with this DeviceN color space. Optional.
<i>processComponents</i>	A vector containing a list of process components. Optional
<i>printingOrder</i>	A vector containing a list defining the printing order. Optional
<i>isNChannel</i>	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

<i>pFactory</i>	The EDL Class factory to use.
<i>colorants</i>	A vector of colorant names for the device inks
<i>alternate</i>	The alternate color space.
<i>tintTransform</i>	A function that maps colors in the space to the alternate color space.
<i>processColorSpace</i>	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

<i>component</i>	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

<i>factory</i>	The EDL Class factory to use.
<i>colorantSpace</i>	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:

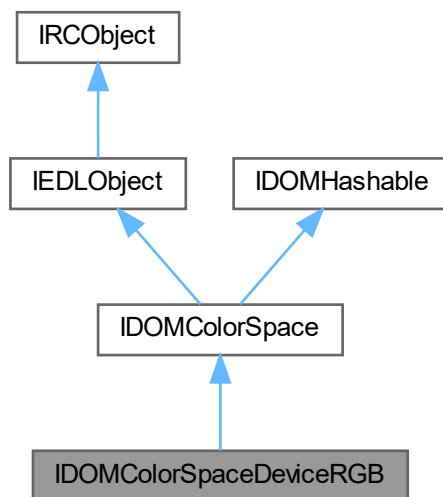
- idomcolorspace.h

7.173 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.

7.173.1 Detailed Description

[IDOMColorSpaceDeviceRGB](#) interface.

7.173.2 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

<code>pFactory</code>	The EDL Class factory to use.
-----------------------	-------------------------------

Returns

[IDOMColorSpaceDeviceRGB](#) The new color space.

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

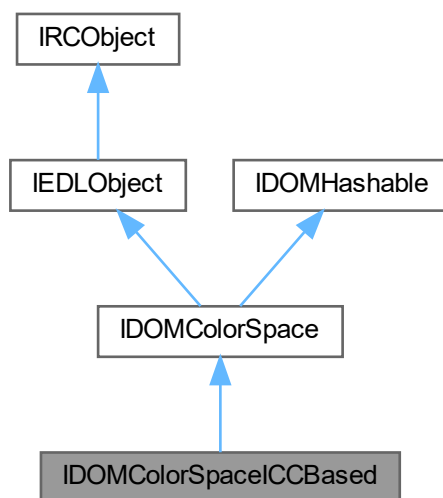
- `idomcolorspace.h`

7.174 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 `IDOMICCProfilePtr` &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.

7.174.1 Detailed Description

Represents a color space described by an ICC profile.

7.174.2 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

<i>pFactory</i>	The class factory.
<i>profile</i>	The profile to use
<i>alternate</i>	The alternate color space, if any. If not supplied and the profile has 1, 3, or 4 components, a Device color 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

<i>profile</i>	A reference to the new ICC profile.
----------------	-------------------------------------

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

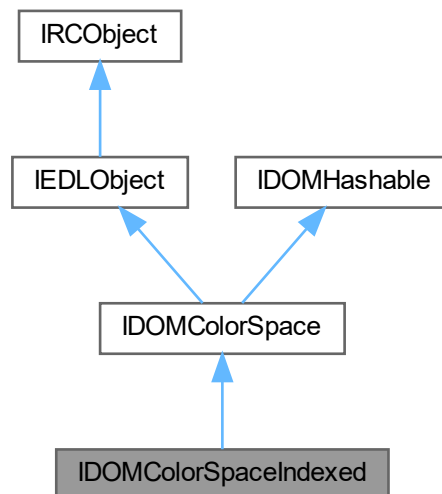
- idomcolorspace.h

7.175 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 IDOMColorSpacePtr &underlying, const IDOMFunctionPtr &mappingFunction)
Create an Indexed Color Space. Throws an [IEDLError](#) on failure.
- static EDL_API IDOMColorSpaceIndexedPtr [create](#) ([IEDLClassFactory](#) *pFactory, const IDOMColorSpacePtr &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.

7.175.1 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

7.175.2 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

<i>pFactory</i>	The EDL Class factory to use.
<i>underlying</i>	The underlying color space.
<i>table</i>	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

<i>pFactory</i>	The EDL Class factory to use.
<i>underlying</i>	The underlying color space.
<i>mappingFunction</i>	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:

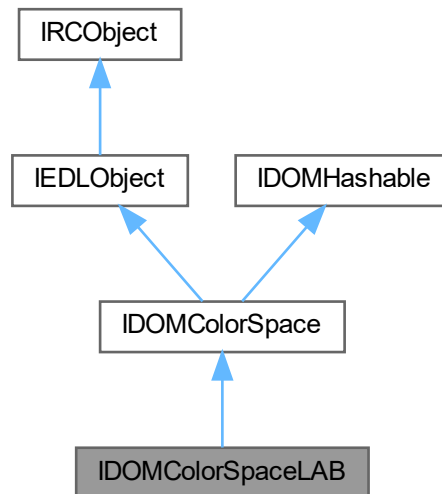
- idomcolorspace.h

7.176 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.

7.176.1 Detailed Description

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

7.176.2 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

<i>pFactory</i>	The EDL Class factory to use.
<i>whitePointX</i>	The CIE X value for the white point.
<i>whitePointY</i>	The CIE Y value for the white point.
<i>whitePointZ</i>	The CIE V value for the white point.
<i>blackPointX</i>	The CIE X value for the black point.
<i>blackPointY</i>	The CIE Y value for the black point.
<i>blackPointZ</i>	The CIE V value for the black point.
<i>rangeALo</i>	The low bound of the A parameter
<i>rangeAHi</i>	The high bound of the A parameter
<i>rangeBLo</i>	The low bound of the B parameter
<i>rangeBHi</i>	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

<i>x</i>	Reference to receive the black point X value
<i>y</i>	Reference to receive the black point Y value
<i>z</i>	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

<i>lowA</i>	Reference to receive the low bound of the A parameter
<i>highA</i>	Reference to receive the high bound of the A parameter
<i>lowB</i>	Reference to receive the low bound of the B parameter
<i>highB</i>	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:

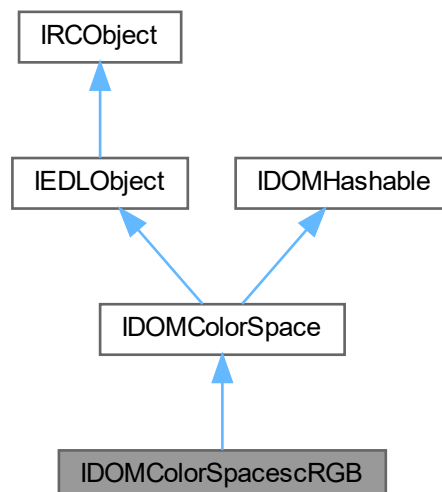
- idomcolorspace.h

7.177 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 [IEDLObject](#).

Public Member Functions inherited from [IRCOBJECT](#)

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

7.177.2 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

<i>pFactory</i>	The EDL Class factory to use
-----------------	------------------------------

Returns

IDOMColorSpacescRGBPtr The new color space

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

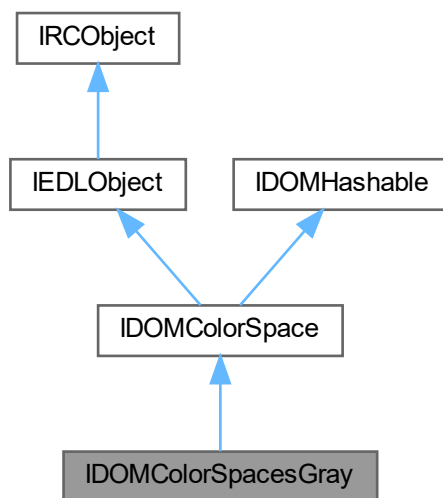
- idomcolorspace.h

7.178 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 `IEDLObject`.

Public Member Functions inherited from [IRCOBJECT](#)

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

7.178.2 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

<i>pFactory</i>	The EDL Class factory to use.
-----------------	-------------------------------

Returns

IDOMColorSpacesGrayPtr The new color space.

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

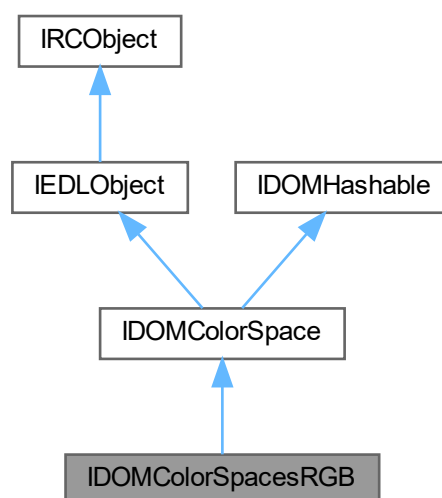
- idomcolorspace.h

7.179 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 [IEDLObject](#).

Public Member Functions inherited from [IRCOBJECT](#)

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

7.179.1 Detailed Description

Represents the RGB color space.

7.179.2 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

<i>pFactory</i>	The EDL Class factory to use.
-----------------	-------------------------------

Returns

IDOMColorSpacesRGBPtr The new color space.

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

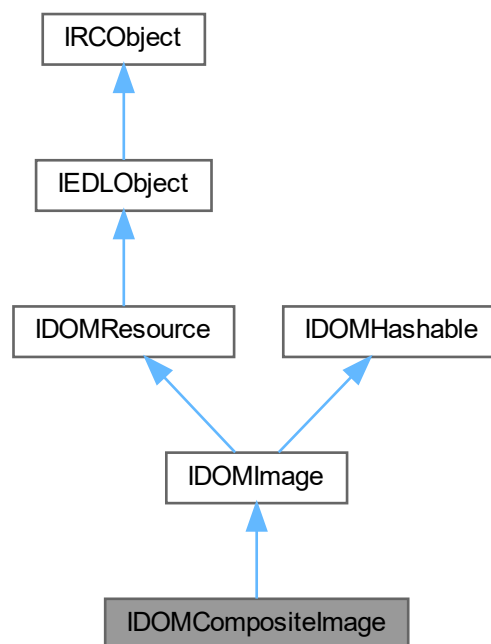
- idomcolorspace.h

7.180 IDOMCompositelImage 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 IDOMCompositelImage:

**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 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 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.

7.180.1 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.

7.180.2 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

<i>pFactory</i>	The EDL Class Factory
<i>componentImages</i>	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

<i>stream</i>	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

<i>stream</i>	A smart pointer to the stream
---------------	-------------------------------

Implements [IDOMResource](#).

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

- idomimageresource.h

7.181 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>
```

7.181.1 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:

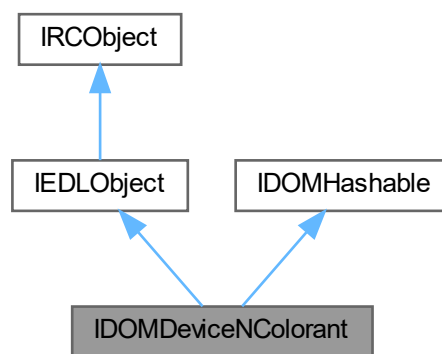
- idomimageresource.h

7.182 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.

7.182.1 Detailed Description

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

7.182.2 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

<i>solidity</i>	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:

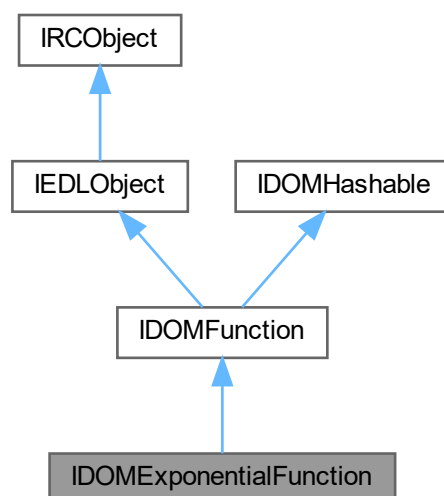
- idomcolorspace.h

7.183 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.

7.183.1 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.

7.183.2 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

<i>outputNum</i>	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

<i>outputNum</i>	The 0-indexed input number
------------------	----------------------------

Returns

float The C1 value

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

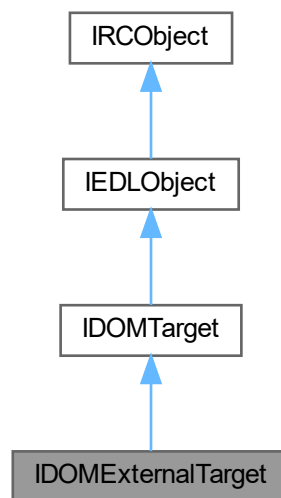
- idomfunction.h

7.184 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.

7.184.1 Detailed Description

[IDOMExternalTarget](#) interface.

This interface describes the existence of links to documents and resources that are not part of the present document.

7.184.2 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

<i>isMap</i>	The new value of IsMap flag.
--------------	------------------------------

setTargetUri()

```
virtual void IDOMExternalTarget::setTargetUri (
    const EDLSysString & uri ) [pure virtual]
```

Sets the target URI.

Parameters

<i>uri</i>	The new target URI.
------------	---------------------

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

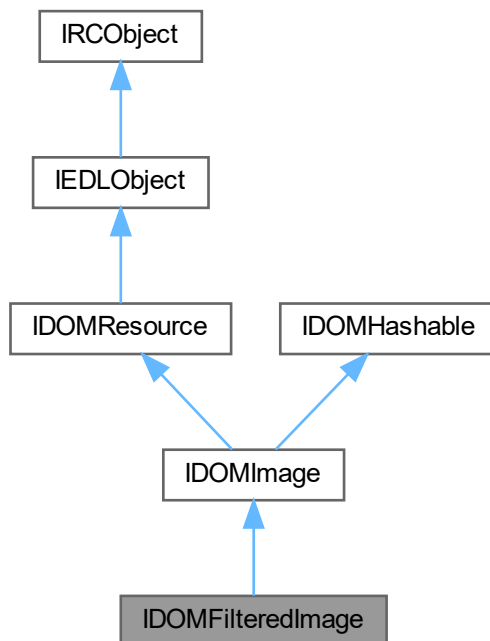
- idomtarget.h

7.185 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 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 `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 &sourceImage, const IDOMImageFilterPtr &filter)`
Simplified creator for a filtered image with a single filter.
- static `EDL_API IDOMFilteredImagePtr create (IEDLClassFactory *pFactory, const IDOMImagePtr &sourceImage, 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.

7.185.1 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.

7.185.2 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

<i>pFactory</i>	The EDL Class Factory
<i>sourceImage</i>	The source image.
<i>filters</i>	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

<i>pFactory</i>	The EDL Class Factory
<i>sourceImage</i>	The source image.
<i>filter</i>	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

<i>index</i>	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

<i>stream</i>	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

<i>filter</i>	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

<i>stream</i>	A smart pointer to the stream
---------------	-------------------------------

Implements [IDOMResource](#).

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

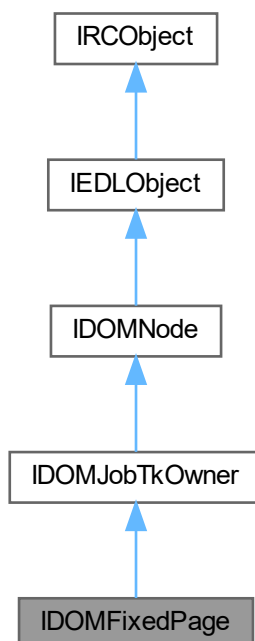
- idomimageresource.h

7.186 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 box 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 box 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
Simplified 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 descendIntoForms=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.

7.186.1 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.

7.186.2 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

<i>pFactory</i>	The EDL class factory
<i>width</i>	Description for width (defaults to 210mm, width of A4)
<i>height</i>	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 box 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 if 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 thumbnail 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

<i>bb</i>	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 box 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

<i>bb</i>	The new content box settings
-----------	------------------------------

setCropBox()

```
virtual void IDOMFixedPage::setCropBox (
    const FRect & cropBox ) [pure virtual]
```

Sets the fixed page's crop box.

Parameters

<i>cropBox</i>	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

<i>height</i>	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

<i>lang</i>	The new language setting.
-------------	---------------------------

setPageGroup()

```
virtual void IDOMFixedPage::setPageGroup (
    const IDOMTransparencyGroupPtr & pageGroup ) [pure virtual]
```

Set the page transparency group.

Parameters

<i>pageGroup</i>	A reference to the desired transparency group, or NULL.
------------------	---

setResourceDictionary()

```
virtual void IDOMFixedPage::setResourceDictionary (
    const IDOMResourceDictionaryPtr & ptrResources ) [pure virtual]
```

Sets the resource dictionary.

Parameters

<i>ptrResources</i>	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

<i>ptrThumbnail</i>	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

<i>trimBox</i>	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

<i>width</i>	New fixed page width
--------------	----------------------

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

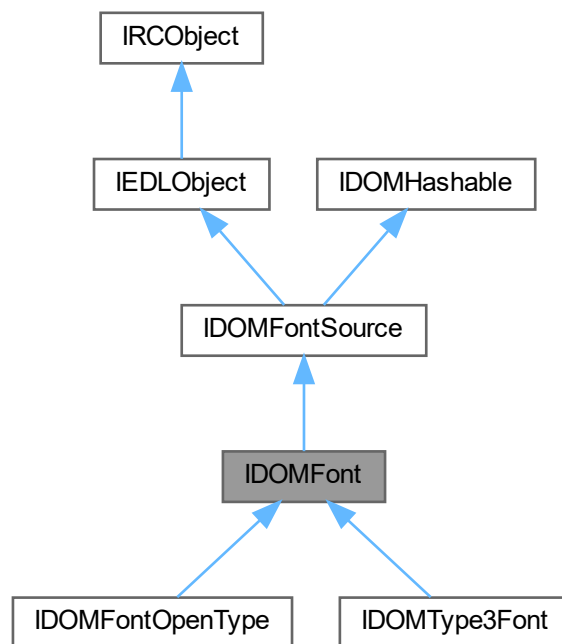
- idompage.h

7.187 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.

7.187.1 Detailed Description

[IDOMFont](#) Base Class.

7.187.2 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

<i>characterMap</i>	The character map to populate.
<i>fontIndex</i>	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

<i>fontSource</i>	The new font source
-------------------	---------------------

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

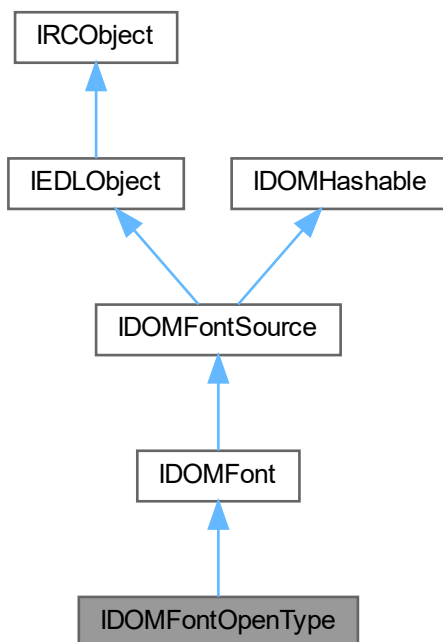
- [idomfont.h](#)

7.188 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 embedding.
- virtual void **setEmbedded** (bool embedded)=0
Sets whether of not the font is flagged for embedding.
- 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 **striplInstructions**=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 [IEDLObject](#).

Public Member Functions inherited from [IRCObject](#)

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

7.188.1 Detailed Description

[IDOMFontOpenType](#) interface.

7.188.2 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

<i>pFactory</i>	EDL class factory to use.
<i>stream</i>	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

<i>ptrSession</i>	A pointer to the EDL session.
<i>fontName</i>	The desired font name
<i>fontIndex</i>	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

<i>ptrSession</i>	A pointer to the EDL session.
<i>usedGlyphs</i>	A vector of glyph IDs that should be retained in the subset font.
<i>usedUnicode</i>	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

<i>ptrSession</i>	A pointer to the EDL session.
<i>regenerateStream</i>	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 embeddedding.

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

<i>fontIndex</i>	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

<i>fontIndex</i>	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

<i>fontIndex</i>	The index of the font - only used for TTC fonts.
<i>stripInstructions</i>	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

<i>fontIndex</i>	The index of the desired font (if a TrueType collection)
<i>pFactory</i>	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

<i>fontIndex</i>	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

<i>pFactory</i>	A pointer to an EDL Class factory.
<i>fontIndex</i>	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

<i>cidMap</i>	The CID map to set. Ownership is taken by the font.
---------------	---

setEmbedded()

```
virtual void IDOMFontOpenType::setEmbedded (
    bool embedded ) [pure virtual]
```

Sets whether or not the font is flagged for embedding.

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

<i>embedded</i>	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

<i>session</i>	A pointer to the session.
<i>fontIndex</i>	The index of the font - only used for TTC fonts.
<i>all</i>	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

<i>session</i>	A pointer to the session.
<i>fontIndex</i>	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](#)

7.189 IDOMFontOTFTrueType Class Reference

Opentype Font.

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

7.189.1 Detailed Description

Opentype Font.

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

- [idomfont.h](#)

7.190 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](#).

7.190.1 Detailed Description

[IDOMFontPCL5](#) (PCL5 TrueType) derived from an OpenType font source.

7.190.2 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

<i>fontName</i>	The font name returned
-----------------	------------------------

Returns

bool True on success

getFontTrueTypeGlyphAccessor()

```
virtual bool IDOMFontPCL5::getFontTrueTypeGlyphAccessor (
    IFontPCL5TrueTypeGlyphAccessorPtr & accessor ) [pure virtual]
```

Creates a TrueType glyph accessor.

Parameters

<i>accessor</i>	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

<i>enumerator</i>	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

<i>enumerator</i>	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

<i>symbolSetID</i>	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

<i>symbolSet</i>	The font symbol set
------------------	---------------------

Returns

bool True on success

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

- [idomfontpcl.h](#)

7.191 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](#).

7.191.1 Detailed Description

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

7.191.2 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

<i>enumerator</i>	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

<i>fontName</i>	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

<i>accessor</i>	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:

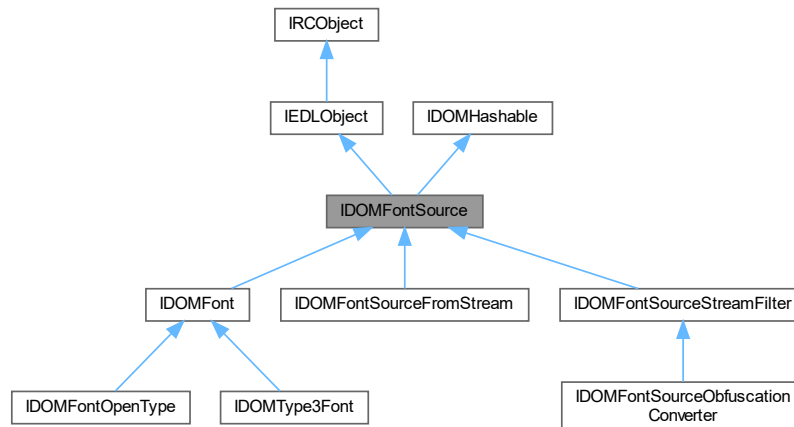
- [idomfontpcl.h](#)

7.192 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.

7.192.1 Detailed Description

The font source for the class [IDOMFont](#). This class describes the different ways fonts are constructed.

7.192.2 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:

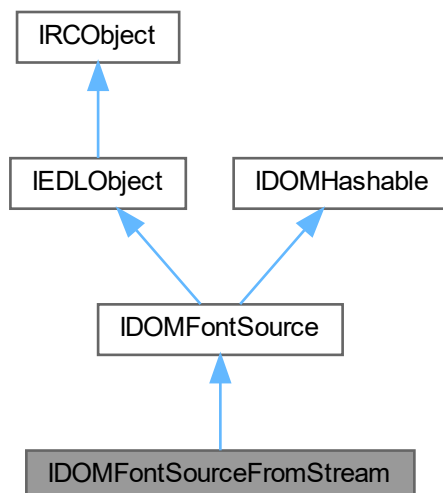
- [idomfont.h](#)

7.193 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 `InputStreamPtr` [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.

7.193.1 Detailed Description

The source for [IDOMFont](#) when sourced from an existing stream.

7.193.2 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:

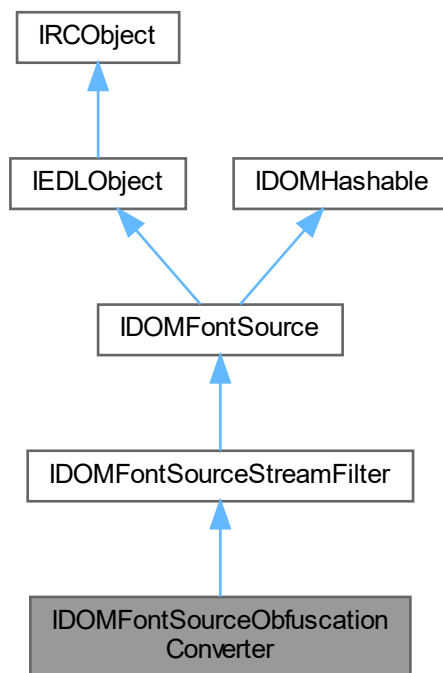
- [idomfont.h](#)

7.194 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 **IEDLObject**.*

Public Member Functions inherited from **IRCObject**

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

7.194.1 Detailed Description

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

7.194.2 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:

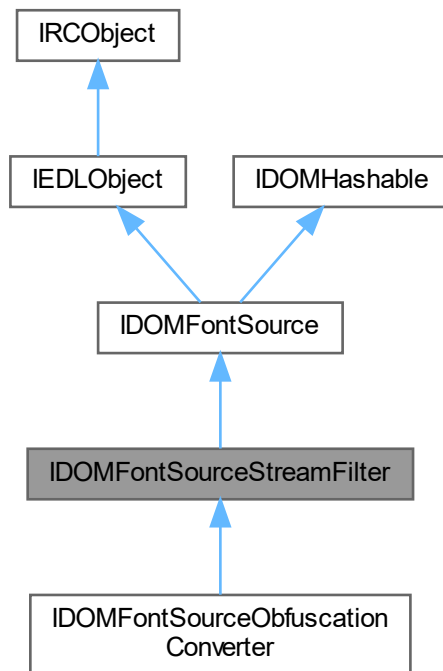
- [idomfont.h](#)

7.195 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 [IRCOject](#)

- virtual void [addRef](#) () const =0
Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool [decRef](#) () const =0
Decreases the reference count of the actual object pointed to. When the reference count falls to 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.

7.195.1 Detailed Description

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

7.195.2 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:

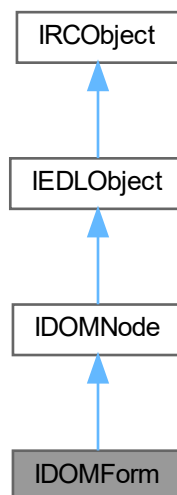
- [idomfont.h](#)

7.196 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 unique 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 bCheckComplete=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

Simplified 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 descendIntoForms=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.

7.196.1 Detailed Description

Interface to DOM node representing PDF-style Form XObjects.

7.196.2 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

<i>pFactory</i>	The EDL class factory
<i>matrix</i>	The desired form matrix
<i>bounds</i>	The desired form bounds
<i>propertiesDictionary</i>	Any additional PDF properties

Returns

IDOMFormPtr A smart pointer to the [IDOMForm](#)

getFormId()

```
virtual DOMid IDOMForm::getFormId ( ) const [pure virtual]
```

Obtain the unique 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

<i>bounds</i>	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

<i>matrix</i>	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

<i>propertiesDictionary</i>	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

<i>elementRef</i>	The structure element reference to set
-------------------	--

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

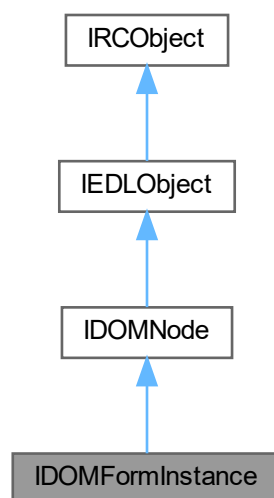
- [idomform.h](#)

7.197 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

- virtual `eBlendMode getBlendMode ()` const =0
Gets the Form associated with this instance.
- 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 bCheckComplete=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
Simplified 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 descendIntoForms=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.

7.197.1 Detailed Description

[IDOMFormInstance](#) interface. This describes an instance of an [IDOMForm](#) in a DOM tree.

7.197.2 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

<i>pFactory</i>	The EDL class factory.
<i>form</i>	The form that this instance will use.
<i>renderTransform</i>	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

<i>blendMode</i>	The desired blend mode
------------------	------------------------

setForm()

```
virtual void IDOMFormInstance::setForm (
    const IDOMFormPtr & form ) [pure virtual]
```

Sets the Form associated with this instance.

Parameters

<i>form</i>	Reference to the desired form.
-------------	--------------------------------

setOpacity()

```
virtual void IDOMFormInstance::setOpacity (
    float opacity ) [pure virtual]
```

Set the form alpha/opacity.

Parameters

<i>opacity</i>	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

<i>ptrOpacityMask</i>	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

<i>renderTransform</i>	Reference to the desired transform
------------------------	------------------------------------

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

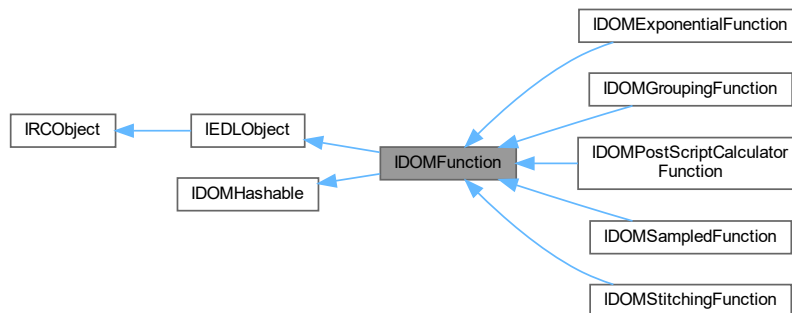
- [idomform.h](#)

7.198 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.

7.198.1 Detailed Description

Base class for PDF/PS Style functions All function instances throw [IEDLError](#) exceptions on failure.

7.198.2 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

<i>inputValues</i>	An array of floats that are input into the function. The size of the array must be the same as the required number of <i>inputValues</i>
<i>outputValues</i>	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 <i>inputValues</i>

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

<i>inputValues</i>	An array of integers that are input into the function. The size of the array must be the same as the required number of <i>inputValues</i>
<i>outputValues</i>	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 <i>inputValues</i>

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

<i>inputNum</i>	The 0-indexed input number
<i>low</i>	A reference to receive the low domain bound for the given <i>inputNum</i>
<i>high</i>	A reference to receive the high domain bound for the given <i>inputNum</i>

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

<i>outputNum</i>	The 0-indexed output number
<i>low</i>	A reference to receive the low range bound for the given outputNum
<i>high</i>	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:

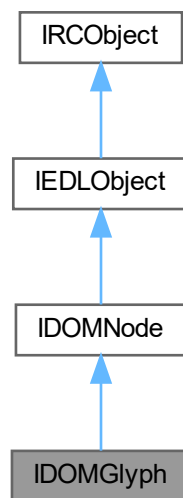
- idomfunction.h

7.199 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 bCheckComplete=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

Simplified 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 descendIntoForms=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](#) ¶ms)
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.

7.199.1 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

7.199.2 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

<i>pFactory</i>	The factory to use.
<i>hasCustomAdvance</i>	Whether or not a custom value has been set for the glyph's advance.
<i>advanceX</i>	The x-component of the advance of the glyph.
<i>advanceY</i>	The y-component of the advance of the glyph.
<i>uOffset</i>	The UOffset value for the glyph.
<i>vOffset</i>	The VOffset value for the glyph.
<i>unicode</i>	The Unicode for the glyph, or an empty string.
<i>glyphName</i>	The glyph name, or an empty string.
<i>bounds</i>	The glyph bounding box.
<i>colored</i>	Indicates if this glyph contains color information.
<i>originalAdvanceX</i>	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 (
    IDLClassFactory * pFactory,
    const IDOMGlyph::Data & params ) [static]
```

Simplified creator for a glyph object Throws an [IEDLError](#) on failure.

Parameters

<i>pFactory</i>	The factory to use.
<i>params</i>	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

<i>glyphName</i>	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

<i>codePoint</i>	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

<i>bColored</i>	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

<i>bHasCustomAdvance</i>	Set to true if the glyph has a custom advance
--------------------------	---

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

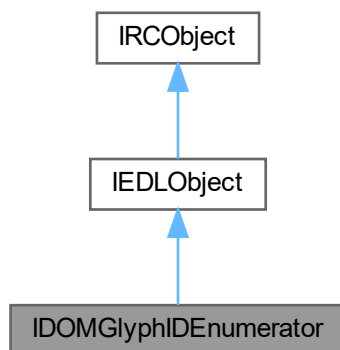
- idomglyph.h

7.200 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 [IRCOject](#)

- virtual void [addRef](#) () const =0
Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool [decRef](#) () const =0
Decreases the reference count of the actual object pointed to. When the reference count falls to 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 [IRCOject](#)

- virtual [~IRCOject](#) ()
Virtual destructor.

7.200.1 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.

7.200.2 Member Function Documentation

add()

```
virtual void IDOMGlyphIDEnumerator::add (  
    IDOMGlyph::GlyphID glyphID ) [pure virtual]
```

Adds a glyph to the end of the list.

Parameters

<i>glyphID</i>	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.

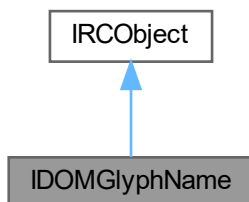
The documentation for this class was generated from the following file:

- idomglyph.h

7.201 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.

7.201.1 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:

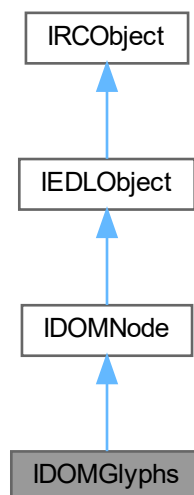
- `idomglyph.h`

7.202 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 invisible (see [#setIsInvisible\(\)](#)).
- 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.
 - virtual bool `getFillOverprints` () const =0
Retrieves whether or not the fill should overprint. Overprinting is honored only for rendering, PDF and PostScript output. It will also be honored should any output need to render as part of producing the output.
 - virtual void `setFillOverprints` (bool overprint=true)=0
Sets whether or not the fill should overprint. Overprinting is honored only for rendering, PDF and PostScript output. It will also be honored should any output need to render as part of producing the output.
 - virtual bool `getOverprintMode` () const =0
Retrieves whether or not overprint should obey PDF/PS overprint mode semantics. PDF and PostScript support a manner of overprinting for CMYK scenarios where an individual CMYK component may overprint if that channel is zero and the rendering target is also the same CMYK color space (excluding some transparency group scenarios). Setting overprint mode to true enables this feature.
 - virtual void `setOverprintMode` (bool mode=true)=0
Sets whether or not overprint should obey PDF/PS overprint mode semantics. PDF and PostScript support a manner of overprinting for CMYK scenarios where an individual CMYK component may overprint if that channel is zero and the rendering target is also the same CMYK color space (excluding some transparency group scenarios). Setting overprint mode to true enables this feature.
 - virtual `eRenderingIntent` `getRenderingIntent` () const =0
Get the rendering intent for the node if present. The rendering intent is used whenever the node requires color conversion to a different color space. Set `eRenderingIntentUnset` to unset the rendering intent, in which a default will be chosen based on the color space in use. This setting is honored only for rendering, color conversion, PDF or PostScript output.

- virtual void [setRenderingIntent](#) ([eRenderingIntent](#) intent)=0
Set or clear the rendering intent for the node. The rendering intent is used whenever the node requires color conversion to a different color space. Set [eRenderingIntentUnset](#) to unset the rendering intent, in which a default will be chosen based on the color space in use. This setting is honored only for rendering, color conversion, PDF or PostScript output.
- virtual [eBlackPointCompensation](#) [getBlackPointCompensation](#) () const =0
Get the black point compensation for the node. The black point compensation setting is used whenever the node requires color conversion to a different color space and affects how dark regions are handled. This setting is honored only for rendering, color conversion, PDF or PostScript output.
- virtual void [setBlackPointCompensation](#) ([eBlackPointCompensation](#) compensation)=0
Set the black point compensation for the node. The black point compensation setting is used whenever the node requires color conversion to a different color space and affects how dark regions are handled. This setting is honored only for rendering, color conversion, PDF or PostScript output.

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 bCheckComplete=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
Simplified 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 descendIntoForms=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 [IDOMPathGeometryPtr](#) &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.

7.202.1 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](#).

7.202.2 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

<i>pFactory</i>	The factory to use.
<i>position</i>	The desired position for the glyphs.
<i>unicodeString</i>	The text to be used.
<i>size</i>	The font size to use, in ems.
<i>brush</i>	The fill brush to use.
<i>font</i>	The font to use.
<i>fontIndex</i>	The index of the font to use, if the font is within a truetype collection.
<i>styleSimulations</i>	The desired style simulations effect.
<i>renderTransform</i>	The render transform 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.

getBlackPointCompensation()

```
virtual eBlackPointCompensation IDOMGlyphs::getBlackPointCompensation ( ) const [pure virtual]
```

Get the black point compensation for the node. The black point compensation setting is used whenever the node requires color conversion to a different color space and affects how dark regions are handled. This setting is honored only for rendering, color conversion, PDF or PostScript output.

Returns

eBlackPointCompensation The black point compensation, or eBPCUnset if not present.

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.

getFillOverprints()

```
virtual bool IDOMGlyphs::getFillOverprints ( ) const [pure virtual]
```

Retrieves whether or not the fill should overprint. Overprinting is honored only for rendering, PDF and PostScript output. It will also be honored should any output need to render as part of producing the output.

Returns

bool True if the fill should overprint, false otherwise.

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.

getOverprintMode()

```
virtual bool IDOMGlyphs::getOverprintMode ( ) const [pure virtual]
```

Retrieves whether or not overprint should obey PDF/PS overprint mode semantics. PDF and PostScript support a manner of overprinting for CMYK scenarios where an individual CMYK component may overprint if that channel is zero and the rendering target is also the same CMYK color space (excluding some transparency group scenarios). Setting overprint mode to true enables this feature.

Returns

bool true if the overprint mode should be enabled, false otherwise.

getRenderingIntent()

```
virtual eRenderingIntent IDOMGlyphs::getRenderingIntent ( ) const [pure virtual]
```

Get the rendering intent for the node if present. The rendering intent is used whenever the node requires color conversion to a different color space. Set `eRenderingIntentUnset` to unset the rendering intent, in which a default will be chosen based on the color space in use. This setting is honored only for rendering, color conversion, PDF or PostScript output.

Returns

eRenderingIntent The rendering intent, or `eRenderingIntentUnset` if not present.

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

<i>bidiLevel</i>	The new <code>bidiLevel</code> value.
------------------	---------------------------------------

setBlackPointCompensation()

```
virtual void IDOMGlyphs::setBlackPointCompensation (
    eBlackPointCompensation compensation ) [pure virtual]
```

Set the black point compensation for the node. The black point compensation setting is used whenever the node requires color conversion to a different color space and affects how dark regions are handled. This setting is honored only for rendering, color conversion, PDF or PostScript output.

Parameters

<i>compensation</i>	The desired black point compensation setting, or <code>eBPCUnset</code> to clear.
---------------------	---

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

<i>blendMode</i>	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

<i>caretStops</i>	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

<i>ptrClip</i>	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

<i>clusters</i>	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

<i>name</i>	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

<i>ptrFill</i>	Smart pointer to the new fill brush.
----------------	--------------------------------------

setFillOverprints()

```
virtual void IDOMGlyphs::setFillOverprints (
    bool overprint = true ) [pure virtual]
```

Sets whether or not the fill should overprint. Overprinting is honored only for rendering, PDF and PostScript output. It will also be honored should any output need to render as part of producing the output.

Parameters

<i>overprint</i>	The desired overprint.
------------------	------------------------

setFont()

```
virtual void IDOMGlyphs::setFont (
    const IDOMFontPtr & font ) [pure virtual]
```

Sets the font object to be used to display the text.

Parameters

<i>font</i>	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

<i>index</i>	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

<i>emSize</i>	The new font rendering em size value.
---------------	---------------------------------------

setIndices()

```
virtual void IDOMGlyphs::setIndices (
    const EDLSysString & indices ) [pure virtual]
```

Sets the glyphs indices.

Parameters

<i>indices</i>	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

<i>isCharPath</i>	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

<i>isInvisible</i>	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

<i>isSideways</i>	The new <code>IsSideways</code> 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

<i>target</i>	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 invisible (see #setIsInvisible()).

Parameters

<i>opc</i>	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

<i>ptrOpacityMask</i>	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

<code>originX</code>	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

<code>originY</code>	The new y-coordinate for the first glyph in the text run.
----------------------	---

setOverprintMode()

```
virtual void IDOMGlyphs::setOverprintMode (
    bool mode = true ) [pure virtual]
```

Sets whether or not overprint should obey PDF/PS overprint mode semantics. PDF and PostScript support a manner of overprinting for CMYK scenarios where an individual CMYK component may overprint if that channel is zero and the rendering target is also the same CMYK color space (excluding some transparency group scenarios). Setting overprint mode to true enables this feature.

Parameters

<code>mode</code>	The desired overprint mode
-------------------	----------------------------

setRenderingIntent()

```
virtual void IDOMGlyphs::setRenderingIntent (
    eRenderingIntent intent ) [pure virtual]
```

Set or clear the rendering intent for the node. The rendering intent is used whenever the node requires color conversion to a different color space. Set `eRenderingIntentUnset` to unset the rendering intent, in which a default will be chosen based on the color space in use. This setting is honored only for rendering, color conversion, PDF or PostScript output.

Parameters

<code>intent</code>	The desired rendering intent, or <code>eRenderingIntentUnset</code> to clear.
---------------------	---

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

<i>matrix</i>	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 `eStyleSimulationsNone`, 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

<i>styleSimulations</i>	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

<i>unicodeString</i>	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:

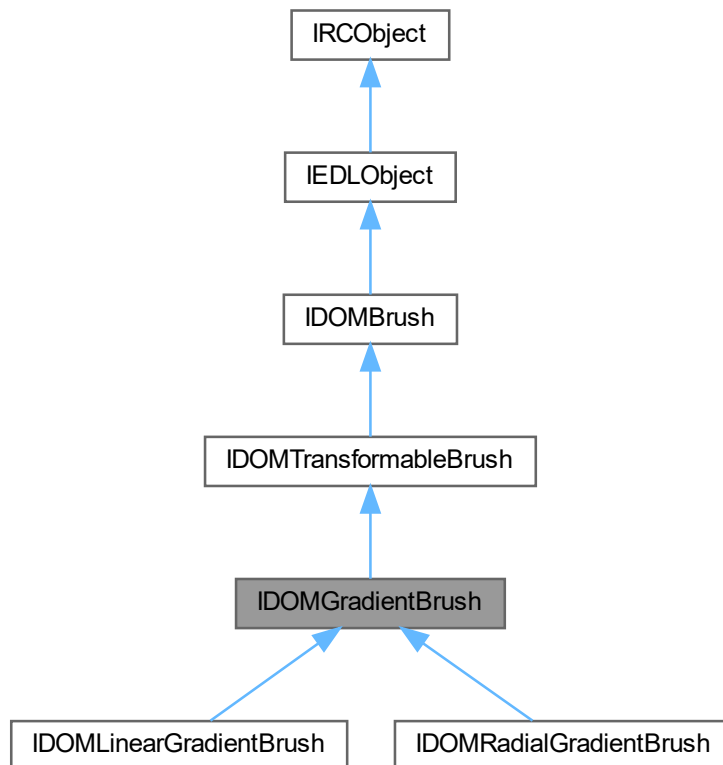
- idomglyphs.h

7.203 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.

7.203.1 Detailed Description

A common interface for both [IDOMLinearGradient](#) and [IDOMRadialGradient](#). Provides straightforward access to common attributes.

7.203.2 Member Function Documentation

addGradientStop()

```
virtual void IDOMGradientBrush::addGradientStop (
    const IDOMGradientStopPtr & ptrGradientStop ) [pure virtual]
```

Append a gradient stop.

Parameters

<i>ptrGradientStop</i>	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

<i>pFactory</i>	The EDL class factory
<i>ptrColorSpace</i>	The desired colors pace. Pass NULL to use the color space of the first brush.
<i>intent</i>	The desired conversion intent if color conversion is required.
<i>bpc</i>	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

<i>cim</i>	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

<i>stops</i>	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

<i>sm</i>	The new spread method value.
-----------	------------------------------

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

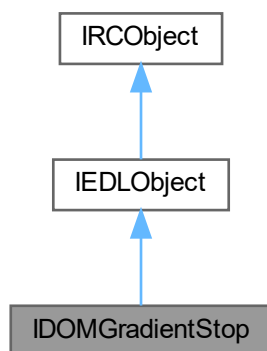
- [idombrush.h](#)

7.204 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.

7.204.1 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.

7.204.2 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

<i>pFactory</i>	The EDL Class factory to use.
<i>color</i>	The color to use.
<i>offset</i>	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

<i>color</i>	The new color value.
--------------	----------------------

setOffset()

```
virtual void IDOMGradientStop::setOffset (
    double offset ) [pure virtual]
```

Sets the offset value of the gradient stop.

Parameters

<i>offset</i>	The new offset value.
---------------	-----------------------

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

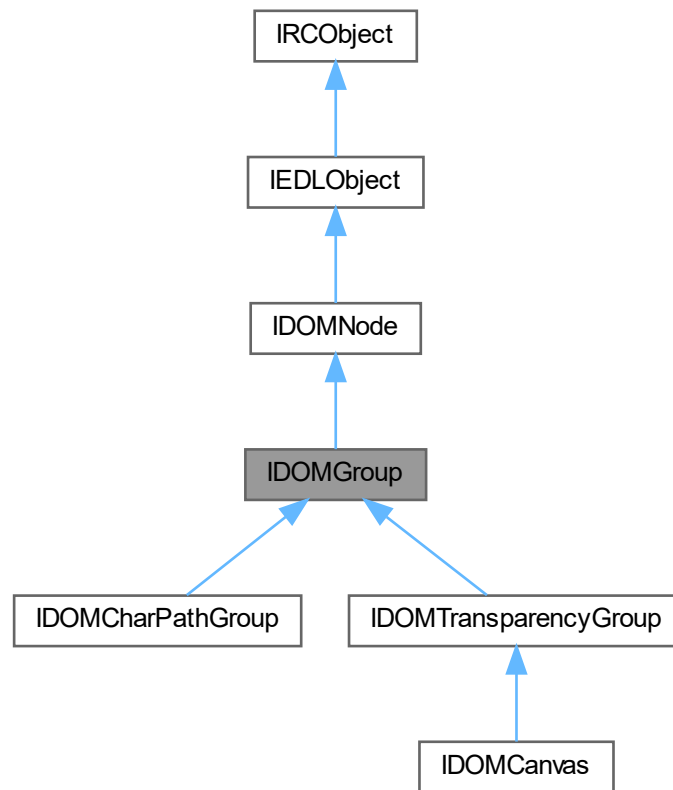
- [idombrush.h](#)

7.205 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 bCheckComplete=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
Simplified 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 descendIntoForms=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.

7.205.1 Detailed Description

Interface to DOM node representing Group Elements.

7.205.2 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

<i>pFactory</i>	The EDL class factory
<i>transform</i>	The desired render transform
<i>clip</i>	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

<i>ptrClip</i>	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

<i>details</i>	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](#) ([getNodeTypeId\(\)](#) is eDOMGroupNode) will result in an exception. Note that an [IDOMGroup](#) may have optional content details or marked content details, but not both.

Parameters

<i>details</i>	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

<i>matrix</i>	Render transform matrix
---------------	-------------------------

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

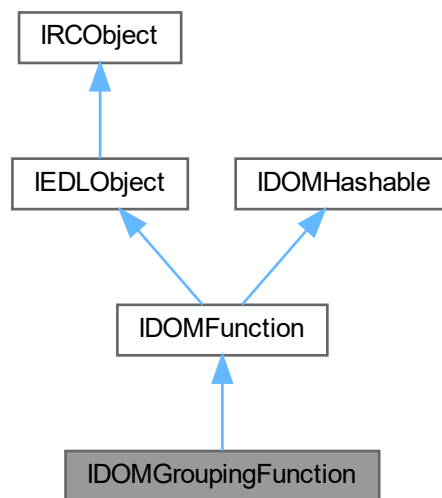
- idomgroup.h

7.206 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.

7.206.1 Detailed Description

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

This allows an array of `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.

7.206.2 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

<i>index</i>	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:

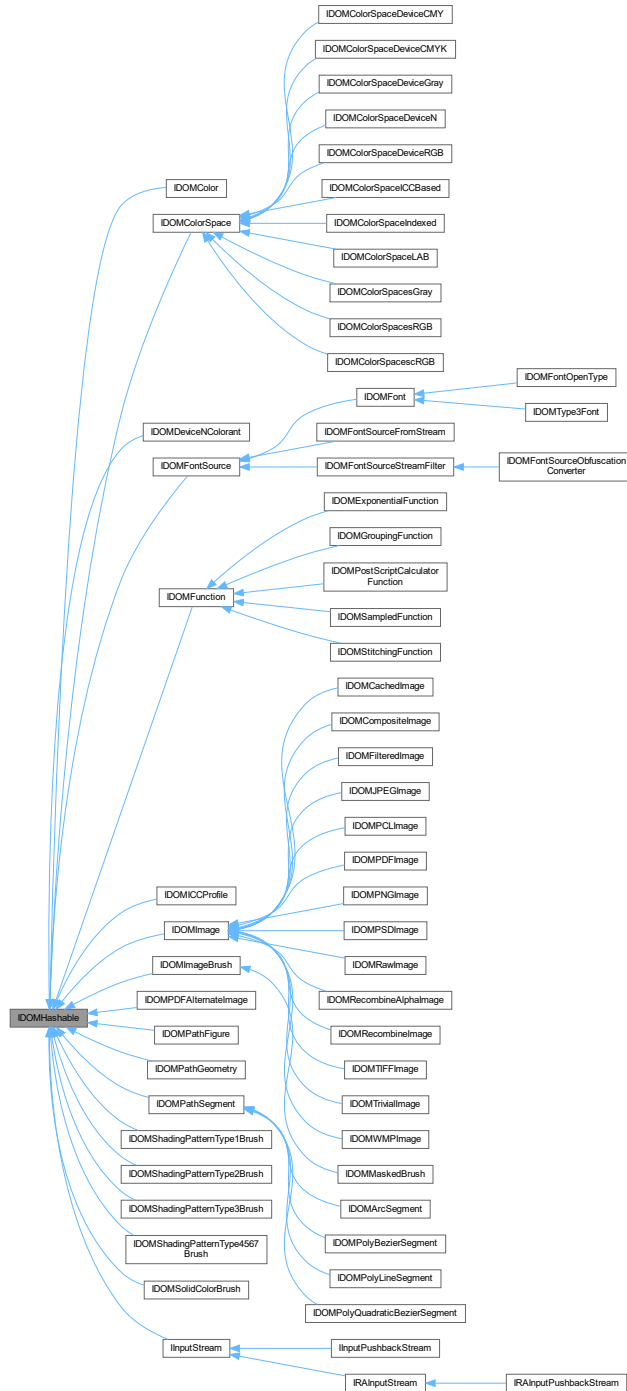
- idomfunction.h

7.207 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.

7.207.1 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.

7.207.2 Member Function Documentation

hash()

```
virtual bool IDOMHashable::hash (
    uint64 & hash ) [pure virtual]
```

Retrieve a hash for this object.

Parameters

<i>hash</i>	A reference to receive the hash.
-------------	----------------------------------

Returns

bool True on success, false if the call fails.

Implemented in [InputStream](#).

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:

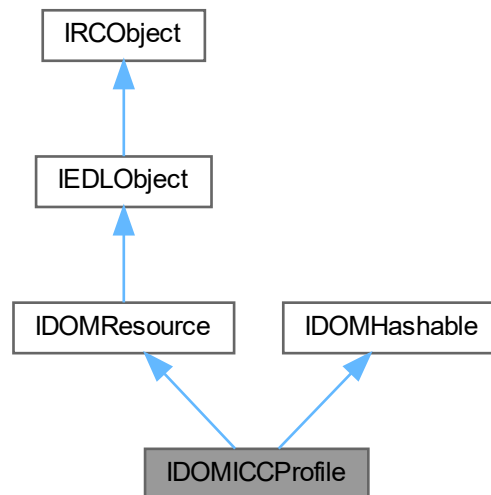
- `idomhashable.h`

7.208 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 [getProfileClass](#) ()=0
Get the profile class for this ICC profile. 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 uint32 [getProfileConnectionSpace](#) ()=0
Get the profile connection space (PCS) type for this ICC profile. For device link profiles this will be the canonical output space. Throws an exception of type [IEDLError](#) on failure.
- virtual bool [getProfileHasSupportForIntent](#) ([eRenderingIntent](#) intent, bool input)=0
Determine if the profile has support for the given intent for the given usage. Matrix shaper style profiles support every intent. LUT-based profiles may have explicit support for some intents. 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 `InputStreamPtr` [getStream](#) () const =0
Retrieves the resource stream.
- virtual void [setStream](#) (const `InputStreamPtr` &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 IDOMICCPProfilePtr` [create](#) (`IEDLClassFactory` *pFactory, const `InputStreamPtr` &stream, `uint32` length=0, const `EDLSysString` &uri=`EDLSysString`())
Creation function for an [IDOMICCPProfile](#) 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.

7.208.1 Detailed Description

[IDOMICCProfile](#) interface.

7.208.2 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

<i>pFactory</i>	The class factory.
<i>stream</i>	The ICC profile stream.
<i>length</i>	The length of the profile stream. If 0, an attempt will be made to determine the length of the stream.
<i>uri</i>	The URI that should be associated with this profile, if required.

Returns

IDOMICCProfilePtr The new profile.

getProfileClass()

```
virtual uint32 IDOMICCProfile::getProfileClass ( ) [pure virtual]
```

Get the profile class for this ICC profile. Throws an exception of type [IEDLError](#) on failure.

Returns

uint32 The ICC profile signature, using the enumeration as per the ICC specification.

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.

getProfileConnectionSpace()

```
virtual uint32 IDOMICCProfile::getProfileConnectionSpace ( ) [pure virtual]
```

Get the profile connection space (PCS) type for this ICC profile. For device link profiles this will be the canonical output space. Throws an exception of type [IEDLError](#) on failure.

Returns

uint32 The ICC PCS color space signature, using the enumeration as per the ICC specification.

getProfileHasSupportForIntent()

```
virtual bool IDOMICCProfile::getProfileHasSupportForIntent (
    eRenderingIntent intent,
    bool input ) [pure virtual]
```

Determine if the profile has support for the given intent for the given usage. Matrix shaper style profiles support every intent. LUT-based profiles may have explicit support for some intents. Throws an exception of type [IEDLError](#) on failure.

Parameters

<i>intent</i>	The intent to query.
<i>input</i>	Set to true if querying for use as an input, false if querying for use as an output.

Returns

bool True if the given intent is supported by the profile for the given direction.

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

<i>majorVersion</i>	Reference to receive the major version.
<i>minorVersion</i>	Reference to receive the minor version.

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

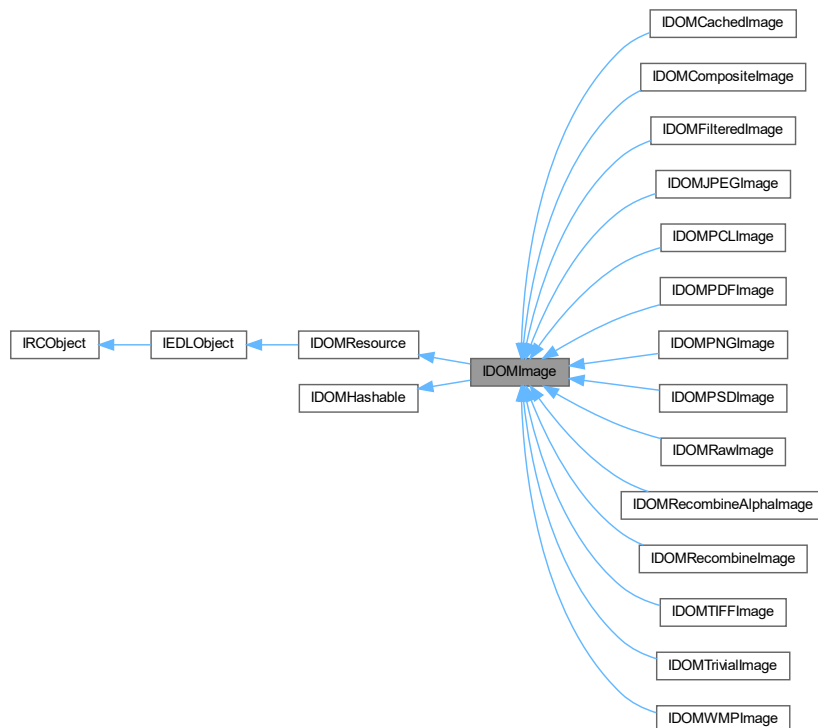
- [idomresources.h](#)

7.209 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 `ImageDecoderPtr` [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 `ImageFramePtr` [getImageFrame](#) (`IEDLClassFactory *factory`)
Fetch the image frame; convenience.
- virtual `ImageEncoderPtr` [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 `InputStreamPtr` `getStream` () const =0
Retrieves the resource stream.
- virtual void `setStream` (const `InputStreamPtr` &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.

7.209.1 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.

7.209.2 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

<i>factory</i>	Pointer to the EDL class factory
<i>imageProperties</i>	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

<i>session</i>	Pointer to the EDL session
<i>imageDest</i>	Smart pointer to the output stream
<i>imageProperties</i>	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

<i>factory</i>	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 color space. internally.

If the operation would make no changes, then the existing image is returned. Otherwise a modified clone will be made.

Parameters

<i>factory</i>	Pointer to the EDL class factory
<i>colorSpace</i>	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

7.210 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.

7.210.1 Detailed Description

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

7.210.2 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

<i>pFactory</i>	The EDL Class Factory
<i>bps</i>	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

7.211 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](#).

7.211.1 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.

7.211.2 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

<i>pFactory</i>	The EDL Class Factory
<i>bleedPixels</i>	The number of pixels to bleed the image.

Returns

IDOMImageBleederFilterPtr The new filter.

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

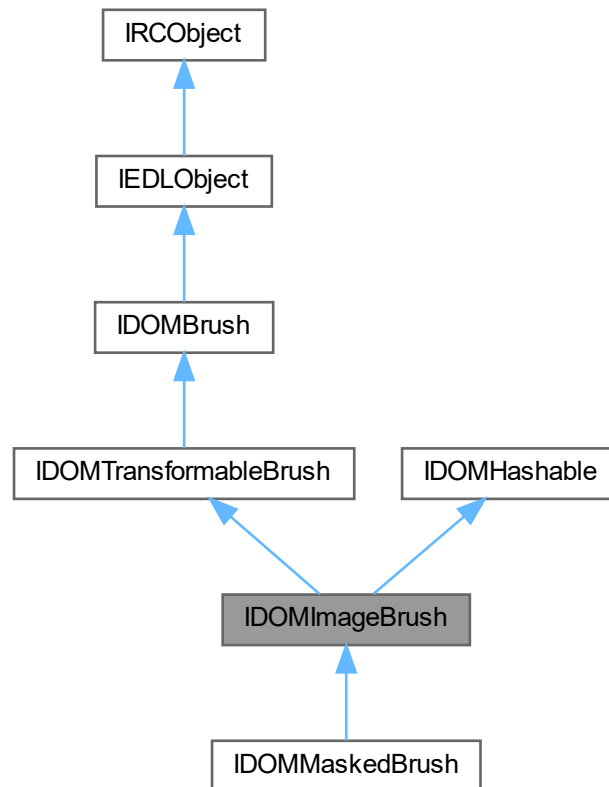
- [idomimageresource.h](#)

7.212 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 [IDOMICCPProfilePtr](#) [getICCProfile](#) () const =0
Retrieves the external ICC profile of the brush if present.
- virtual void [setICCProfile](#) (const [IDOMICCPProfilePtr](#) &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 &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.

7.212.1 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.

7.212.2 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

<i>pFactory</i>	The factory to use.
<i>image</i>	The image to use.
<i>viewBox</i>	The desired view box. If empty, the viewPort will be set to the full area of the image.
<i>viewPort</i>	The desired view port.
<i>renderTransform</i>	The desired render transform.
<i>opacity</i>	The desired brush opacity.
<i>tileMode</i>	The desired tile mode.
<i>alternateImages</i>	The PDF Alternate image brushes, if present.
<i>optionalContentDetails</i>	The optional content details. Ignored for tiling images
<i>propertiesDictionary</i>	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

<i>pFactory</i>	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

<i>pFactory</i>	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

<i>alternates</i>	The alternate images.
-------------------	-----------------------

setICCProfile()

```
virtual void IDOMImageBrush::setICCProfile (
    const IDOMICCProfilePtr & icc ) [pure virtual]
```

Retrieves the external ICC profile of the brush if present.

Parameters

<i>icc</i>	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

<i>ptrImageSource</i>	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

<i>propertiesDictionary</i>	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

<i>tm</i>	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

<i>vb</i>	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

<i>vu</i>	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

<i>vp</i>	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

<i>vu</i>	The new viewport units value
-----------	------------------------------

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

- [idombrush.h](#)

7.213 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.

7.213.1 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.

7.213.2 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

<i>pFactory</i>	The EDL Class Factory
<i>selection</i>	Which channel(s) to retain during filtering.
<i>channel</i>	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

7.214 IDOMImageColorConverterFilter Class Reference

An image filter that presents a color 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.

7.214.1 Detailed Description

An image filter that presents a color converted version of an image.

7.214.2 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

<i>pFactory</i>	The EDL Class Factory
<i>space</i>	The target color space. Must be a base space; ie no DeviceN or Indexed spaces.
<i>intent</i>	The rendering intent to use
<i>bpc</i>	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](#)

7.215 IDOMImageColorKeyFilter Class Reference

An image filter that presents a masked image where colors 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.

7.215.1 Detailed Description

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

7.215.2 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

<i>pFactory</i>	The EDL Class Factory
<i>key</i>	A vector that describes the range of colors 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

7.216 IDOMImageColorSpaceSubstitutionFilter Class Reference

An image filter that presents an identical image, just with the colorspace 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.

7.216.1 Detailed Description

An image filter that presents an identical image, just with the colorspace substituted.

7.216.2 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 IDOMImageColorSpaceSubstitutionFilter::create (
    IEDLClassFactory * pFactory,
    const IDOMColorSpacePtr & colorSpace ) [static]
```

Simplified creator for a color space substitutor filter.

Parameters

<i>pFactory</i>	The EDL Class Factory
<i>colorSpace</i>	The color space to substitute.

Returns

IDOMImageColorSpaceSubstitutionFilterPtr The new filter.

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

- idomimageresource.h

7.217 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](#).

7.217.1 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.

7.217.2 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

<i>pFactory</i>	The EDL Class Factory
<i>decode</i>	The decode vector.

Returns

IDOMImageDecodeFilterPtr The new filter.

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

- [idomimageresource.h](#)

7.218 IDOMImageDeindexerFilter Class Reference

An image filter that presents an image with an Indexed color 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.

7.218.1 Detailed Description

An image filter that presents an image with an Indexed color space as a simple eight bit image.

7.218.2 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

<i>pFactory</i>	The EDL Class Factory
-----------------	-----------------------

Returns

IDOMImageDeindexerFilterPtr The new filter.

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

- idomimageresource.h

7.219 IDOMImageDeviceNToBaseFilter Class Reference

An image filter that presents an image with a DeviceN color 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](#).

7.219.1 Detailed Description

An image filter that presents an image with a DeviceN color space as a simple image in the alternate space.

7.219.2 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

<i>pFactory</i>	The EDL Class Factory
-----------------	-----------------------

Returns

IDOMImageDeviceNToBaseFilterPtr The new filter.

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

- idomimageresource.h

7.220 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](#).

7.220.1 Detailed Description

An image filter that presents a downsampled version of an image.

7.220.2 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

<i>pFactory</i>	The EDL Class Factory
<i>xRes</i>	The desired x resolution of the result.
<i>yRes</i>	The desired y resolution of the result. If 0, the x resolution will be used.
<i>method</i>	The desired downsampling method to use.

Returns

IDOMImageDownsamplerFilterPtr The new filter.

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

- idomimageresource.h

7.221 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.

7.221.1 Detailed Description

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

7.221.2 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

<i>pFactory</i>	The EDL Class Factory
-----------------	-----------------------

Returns

IDOMImageInverterFilterPtr The new filter.

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

- idomimageresource.h

7.222 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 IDOMColorPtr &color)
Simplified creator for a mask expander filter.
- static const [CClassID](#) & [classID](#) ()
Retrieves class id of IDOMImageMaskExpanderFilter.

7.222.1 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.

7.222.2 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

<i>pFactory</i>	The EDL Class Factory
<i>color</i>	The color to use.

Returns

IDOMImageMaskExpanderFilterPtr The new filter.

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

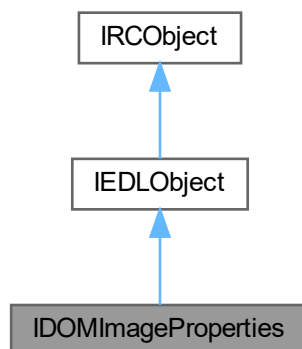
- idomimageresource.h

7.223 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.

7.223.1 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.

7.223.2 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

<i>name</i>	Property name.
<i>value</i>	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

<i>name</i>	Property name.
<i>value</i>	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

<i>name</i>	Property name.
<i>value</i>	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

<i>name</i>	Property name.
<i>value</i>	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

<i>name</i>	Property name.
<i>value</i>	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

<i>name</i>	Property name.
<i>value</i>	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

<i>name</i>	Property name.
<i>object</i>	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

<i>name</i>	Property name.
<i>value</i>	The string value for the property.

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

- idomimageresource.h

7.224 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](#).

7.224.1 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.

7.224.2 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

<i>pFactory</i>	The EDL Class Factory.
<i>inProcessSpace</i>	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.
<i>mergeSpots</i>	If false, no spots will be merged (and #spotsToMerge is ignored).
<i>spotsToMerge</i>	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.
<i>spotsToRetain</i>	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.
<i>spotsToDrop</i>	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.
<i>processConversions</i>	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

<i>pFactory</i>	The EDL Class Factory.
<i>inProcessSpace</i>	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.

Parameters

<i>outProcessSpace</i>	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 <i>inProcessSpace</i> does not match <i>outProcessSpace</i> , then process components will be converted from the <i>inProcessSpace</i> to the <i>outProcessSpace</i> before any spot merging takes place.
<i>mergeSpots</i>	If false, no spots will be merged (and <i>#spotsToMerge</i> is ignored).
<i>spotsToMerge</i>	The spots to merge. Consulted only if <i>#mergeSpots</i> 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 <i>DeviceN</i> 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 <i>outProcessSpace</i> .
<i>spotsToRetain</i>	The names of colorants to retain, even if subject to <i>mergeSpots</i> or <i>spotsToMerge</i> . 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.
<i>spotsToDrop</i>	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.
<i>intent</i>	The rendering intent to use if process components require color conversion.
<i>bpc</i>	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:

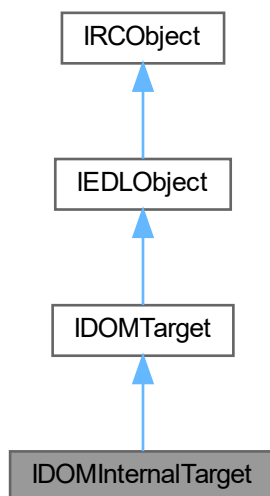
- `idomimageresource.h`

7.225 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.

7.225.1 Detailed Description

The [IDOMInternalTarget](#) interface describes the targets of hyperlinks that are in the same document but not on the current page.

7.225.2 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

<i>dom</i> ↔ <i>Id</i>	The internal target id.
---------------------------	-------------------------

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

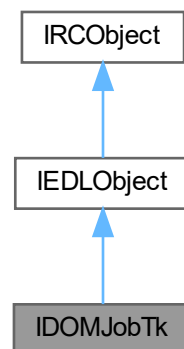
- idomtarget.h

7.226 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 &namespace)=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.

7.226.1 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.

7.226.2 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

<i>nodeType</i>	Node type of the the node's child to search for.
<i>qname</i>	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 & namespace ) [pure virtual]
```

Searches through the node's direct child set for the node type, name and namespace matching the information provided.

Parameters

<i>nodeType</i>	Type of the node's child to search for.
<i>name</i>	Name of the node's child to search for.
<i>namespace</i>	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

<i>ptrCombinedContent</i>	A smart pointer to receive the resulting combined content.
<i>addDefaultJobTk</i>	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

<i>ptrContent</i>	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

<i>ptrOwner</i>	Smart pointer to the new owner node.
-----------------	--------------------------------------

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

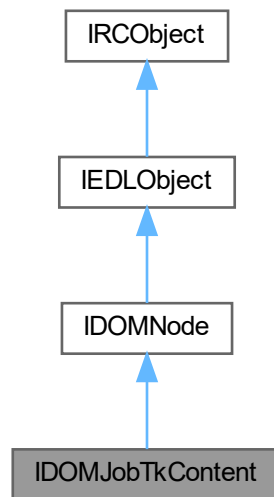
- idomjobtk.h

7.227 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 [eDOMJobTkLevelInvalid](#).
- 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 [eDOMJobTkLevelInvalid](#).
- 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 namespace 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 &slnit)=0
Fills the JobTicket using input from the initialisation string.
- virtual void [addParameterInit](#) (const IEDLNamespacePtr ¶meterNamespace, const EDLSysString ¶meterName, 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 bCheckComplete=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`
Simplified 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, GDOMNodeVect &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 descendIntoForms=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 ¶mName, const EDLSysString &initString, const EDLSysString ¶mValue="", 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.

7.227.1 Detailed Description

Represents the content element of the JobTicket.

7.227.2 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

<i>featureNamespace</i>	Smart pointer to the feature name namespace.
<i>featureName</i>	Name of feature.
<i>option</i>	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

<i>prefix</i>	Value of the new namespace prefix
<i>name</i>	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

<i>ptrNamespace</i>	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

<i>parameterNamespace</i>	Smart pointer to the parameter namespace.
<i>parameterName</i>	Name of parameter.
<i>value</i>	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

<i>paramName</i>	The parameter name.
<i>initString</i>	The init string to add to
<i>paramValue</i>	The optional parameter value.
<i>overwrite</i>	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 & namespace ) [pure virtual]
```

Constructs a qualified name based on a name and namespace provided. The namespace must be present in the namespace collection.

Parameters

<i>name</i>	Name of JobTicket content element
<i>namespace</i>	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

<i>nodeType</i>	Job ticket node type.
<i>qname</i>	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 & namespace ) [pure virtual]
```

Searches through the node's direct child set for a node matching the provided node type, name and namespace.

Parameters

<i>nodeType</i>	Job ticket node type.
<i>name</i>	Name of JobTicket node.
<i>namespace</i>	Namespace of JobTicket node.

Returns

IDOMJobTkNodePtr. Returns smart pointer to the retrieved JobTicket node.

findNamespaceByName()

```
virtual IEDLNamespacePtr IDOMJobTkContent::findNamespaceByName (
    const EDLSysString & namespace ) [pure virtual]
```

Finds a namespace in the namespace collection by name.

Parameters

<i>nmspace</i>	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 namespace in the namespace collection by prefix.

Parameters

<i>prefix</i>	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

<i>filename</i>	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

<i>sInit</i>	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

<i>xmlStream</i>	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 eDOMJobTkLevelNotValid.

Parameters

<i>level</i>	Job ticket level.
--------------	-------------------

setModified()

```
virtual void IDOMJobTkContent::setModified (
    bool modified ) [pure virtual]
```

Sets the "modified" flag.

Parameters

<i>modified</i>	New value of the "modified" flag.
-----------------	-----------------------------------

setVersion()

```
virtual void IDOMJobTkContent::setVersion (
    double version ) [pure virtual]
```

Sets the version of the JobTicket content.

Parameters

<i>version</i>	Job ticket version.
----------------	---------------------

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

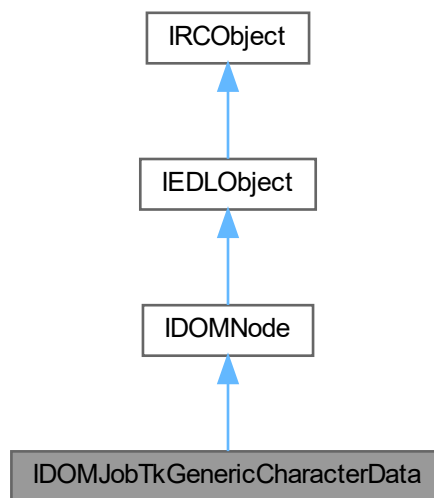
- idomjobtk.h

7.228 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 bCheckComplete=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
*Simplified 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 descendIntoForms=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 **IRCOject**

- virtual void **addRef** () const =0
Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool **decRef** () const =0
Decreases the reference count of the actual object pointed to. When the reference count falls to 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 *IDOM.JobTkGenericNode*.*

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 [IDOMJobTkGenericCharacterData](#) node.

Objects of type [IDOMJobTkGenericCharacterData](#) store CDATA associated with an [IDOMJobTkGenericNode](#). Objects of this type are stored as children of [IDOMJobTkGenericNodes](#).

7.228.2 Member Function Documentation

[appendCharacterData\(\)](#)

```
virtual void IDOMJobTkGenericCharacterData::appendCharacterData (
    const EDLString & charData ) [pure virtual]
```

Append to the node's CDATA.

Parameters

<i>charData</i>	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

<i>charData</i>	The CDATA as an EDLString
-----------------	---------------------------

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

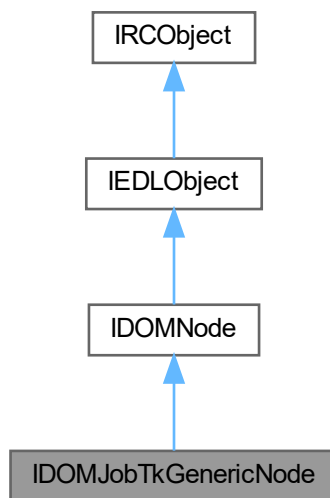
- idomjobtk.h

7.229 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 \sim [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 bCheckComplete=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

Simplified 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, GDOMNodeVect &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 descendIntoForms=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.

7.229.1 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.

7.229.2 Member Function Documentation

addAttribute()

```
virtual void IDOMJobTkGenericNode::addAttribute (
    const EDLQName & qname,
    const EDLSysString & value ) [pure virtual]
```

Adds an attribute.

Parameters

<i>qname</i>	Qualified name of the attribute
<i>value</i>	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

<i>index</i>	The index of the attribute, with 0 representing the first attribute
<i>qname</i>	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:

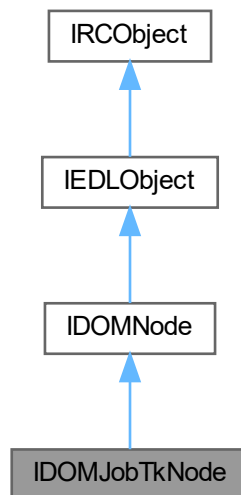
- idomjobtk.h

7.230 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 bCheckComplete=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
Simplified 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 descendIntoForms=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.

7.230.1 Detailed Description

Represents a Job Ticket Node.

7.230.2 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

<i>nodeType</i>	Job ticket node type.
<i>qname</i>	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 & namespace ) [pure virtual]
```

Searches through the node's direct child set for a node matching the provided node type, name and namespace.

Parameters

<i>nodeType</i>	Job ticket node type.
<i>name</i>	Name of JobTicket node.
<i>namespace</i>	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

<i>nodeType</i>	The node type. See <code>eDOMJobTkNodeType</code> .
-----------------	---

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

<i>qname</i>	Qualified name value
--------------	----------------------

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

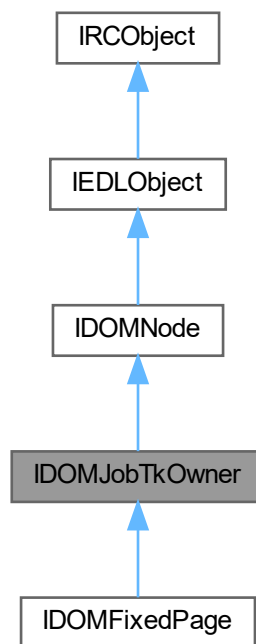
- `idomjobtk.h`

7.231 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 bCheckComplete=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
Simplified 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 descendIntoForms=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.

7.231.1 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.

7.231.2 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

<i>ptrJobTicket</i>	Smart pointer to the new JobTicket.
---------------------	-------------------------------------

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

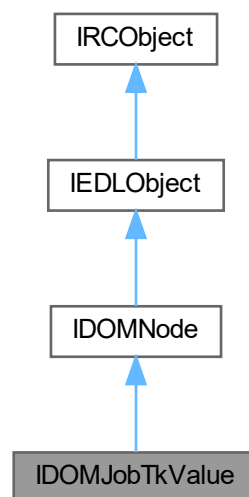
- idomjobtk.h

7.232 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 bCheckComplete=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`
Simplified 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, GDOMNodeVect &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 descendIntoForms=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.

7.232.1 Detailed Description

Represents a Job Ticket value element.

7.232.2 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

<i>value</i>	The new node value.
--------------	---------------------

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

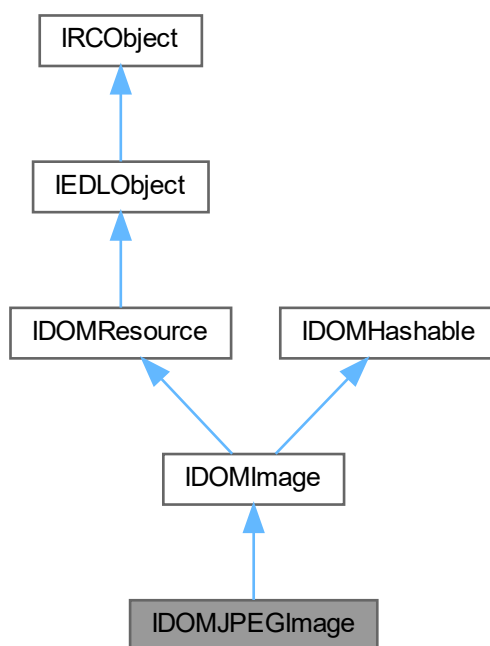
- idomjobtk.h

7.233 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, IImageFrame↔WriterPtr &frame, const IDOMColorSpacePtr &colorSpace, uint32 width, uint32 height, uint8 bitsPer↔Component=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 IOutputStreamPtr &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 color space.

- static EDL_API void [encode](#) (const ISessionPtr &pSession, const IImageFramePtr &frame, const IOutputStreamPtr &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 color 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 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.

Protected Member Functions inherited from [IRCOBJECT](#)

- virtual [~IRCOBJECT](#) ()
Virtual destructor.

7.233.1 Detailed Description

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

7.233.2 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

<i>pFactory</i>	The EDL Class factory to use.
<i>stream</i>	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

<i>session</i>	The session to use
<i>frame</i>	On exit, this is populated with a frame ready to receive image data via <code>frame->writeScanLine()</code> . Use <code>frame->flushData()</code> to complete the encoding process.
<i>colorSpace</i>	The color space to use. Must be compatible with the JPEG format.
<i>width</i>	The width of the image, in pixels.
<i>height</i>	The height of the image, in pixels.
<i>bitsPerComponent</i>	The number of bits per color component, either 8 or 16. Default is 8.
<i>xResolution</i>	The x resolution, in pixels-per-inch.
<i>yResolution</i>	The y resolution, in pixels-per-inch.
<i>quality</i>	The desired quality in the range 1 through 5, with 1 being lowest and 5 being highest. Default is 3.
<i>inStream</i>	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.
<i>outStream</i>	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 color space.

Parameters

<i>pSession</i>	The relevant EDL session
<i>image</i>	The image to be encoded
<i>stream</i>	The stream to use to store the image data.
<i>quality</i>	The desired quality in the range 1 through 5, with 1 being lowest and 5 being highest. Default is 3.
<i>allowCmyk</i>	Whether or not CMYK encoding should be allowed. CMYK JPEGs are not universally supported, and so are not generated by default.
<i>invertCmyk</i>	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 color space.

Parameters

<i>pSession</i>	The relevant EDL session
<i>frame</i>	The frame providing the source image data
<i>stream</i>	The stream to use to store the image data.
<i>quality</i>	The desired quality in the range 1 through 5, with 1 being lowest and 5 being highest. Default is 3.
<i>allowCmyk</i>	Whether or not CMYK encoding should be allowed. CMYK JPEGs are not universally supported, and so are not generated by default.
<i>invertCmyk</i>	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:

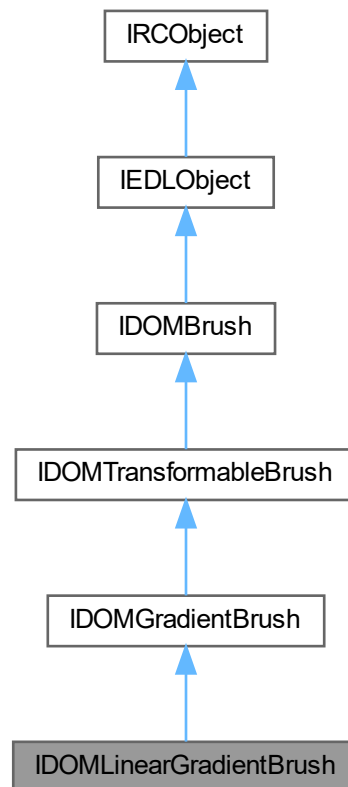
- idomimageresource.h

7.234 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.

7.234.1 Detailed Description

[IDOMLinearGradientBrush](#) interface. A linear gradient brush is used to specify a gradient along a vector.

7.234.2 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

<i>pFactory</i>	The EDL Class factory to use.
<i>startPoint</i>	The start point of the gradient
<i>endPoint</i>	The end point of the gradient
<i>stops</i>	The gradient stops
<i>opacity</i>	The alpha to use.
<i>renderTransform</i>	The render transform to use
<i>spreadMethod</i>	The desired spread method
<i>colorInterpolationMode</i>	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

<i>pFactory</i>	The EDL class factory
<i>useFirstStopColorSpace</i>	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 ignoreSpreadMethod is set, a shading pattern resembling a pad reflect mode will be generated. Further, alpha information in the gradient stops are ignored.

Parameters

<i>pFactory</i>	The EDL class factory
<i>useFirstStopColorSpace</i>	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.
<i>ignoreSpreadMethod</i>	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

<i>pt</i>	The new end point
-----------	-------------------

setStartPoint()

```
virtual void IDOMLinearGradientBrush::setStartPoint (
    const FPoint & pt ) [pure virtual]
```

Sets the start point of the linear gradient.

Parameters

<i>pt</i>	The new start point.
-----------	----------------------

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

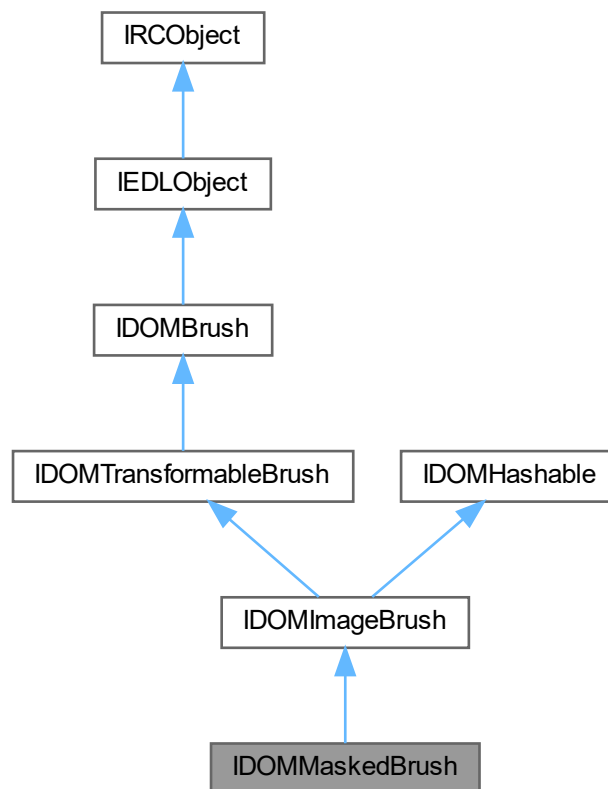
- [idombrush.h](#)

7.235 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 represents 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 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::OptionalContentDetailsPtr &optionalContentDetails=JawsMako::OptionalContentDetailsPtr(), const JawsMako::IPDFDictionaryPtr &properties=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 &alternateImages=CDOMAlternateImageVect(), const JawsMako::OptionalContentDetailsPtr &optionalContentDetails=JawsMako::OptionalContentDetailsPtr(), const JawsMako::IPDFDictionaryPtr &properties=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.

7.235.1 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.

7.235.2 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

<i>pFactory</i>	The factory to use.
<i>mask</i>	The image mask to use.
<i>brush</i>	The brush to use
<i>viewBox</i>	The desired view box. If empty, the viewPort will be set to the full area of the image.
<i>viewPort</i>	The desired view port.
<i>renderTransform</i>	The desired render transform.
<i>opacity</i>	The desired brush opacity.
<i>alternateImages</i>	A vector containing alternate images.
<i>optionalContentDetails</i>	Any optional content details to be applied to this masked case
<i>propertiesDictionary</i>	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

<i>pFactory</i>	A pointer to an EDL class factory
<i>enclosedBounds</i>	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

<i>pFactory</i>	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 represents 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

<i>pFactory</i>	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

<i>brush</i>	Smart pointer to the brush resource to be painted through the image.
--------------	--

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

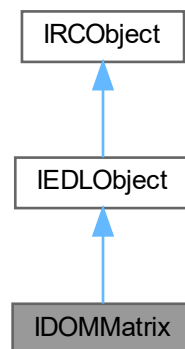
- [idombrush.h](#)

7.236 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 [IRObject](#)

- virtual [~IRObject](#) ()
Virtual destructor.

7.236.1 Detailed Description

Defines the render transform matrix.

7.236.2 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

<i>matrix</i>	Render transform matrix
---------------	-------------------------

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

- [idomresources.h](#)

7.237 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>
```

7.237.1 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:

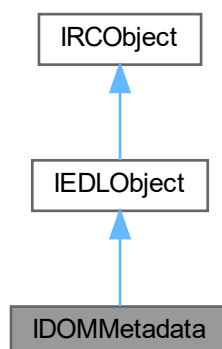
- [idomimageresource.h](#)

7.238 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.

7.238.1 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

7.238.2 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

<i>propertyType</i>	The type of the property whose value is to be retrieved.
<i>propertyName</i>	The name of the property whose value is to be retrieved.
<i>propertyValue</i>	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

<i>propertyType</i>	The type of the property to be removed.
<i>propertyName</i>	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

<i>propertyType</i>	The type of the property whose value is to be set.
<i>propertyName</i>	The name of the property whose value is to be set.
<i>propertyValue</i>	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:

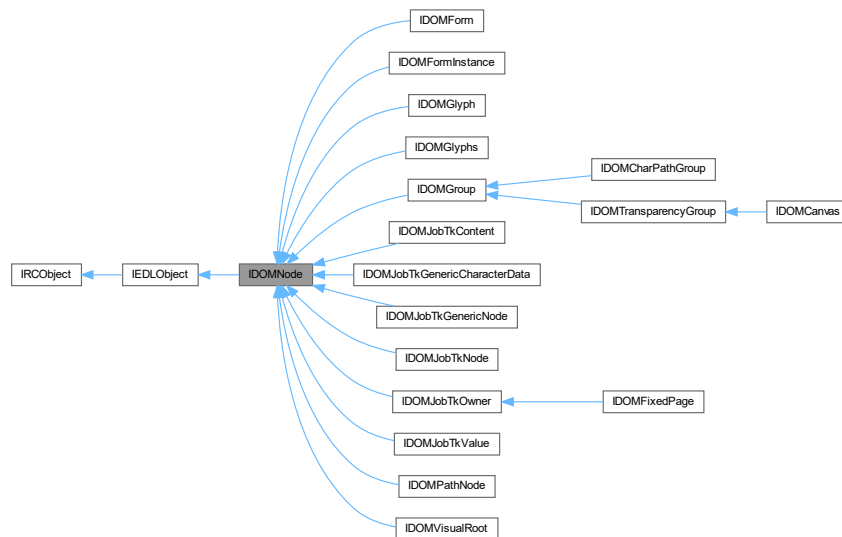
- idommetadata.h

7.239 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 bCheckComplete=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`
Simplified 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, GDOMNodeVect &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 descendIntoForms=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.

7.239.1 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.

7.239.2 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

<i>child</i>	The child node to append.
--------------	---------------------------

cloneNode()

```
virtual IDOMNodePtr IDOMNode::cloneNode (
    IEDLClassFactory * pFactory ) const [pure virtual]
```

Simplified node cloning. An exception of type [IEDLError](#) will be thrown on failure.

Parameters

<i>pFactory</i>	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

<i>pFactory</i>	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

<i>pFactory</i>	The EDL class factory.
<i>dest</i>	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

<i>pSourceNode</i>	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

<i>node</i>	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

<i>child</i>	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

<i>type</i>	the type of node to find
<i>nodes</i>	Vector to receive the found nodes.
<i>searchForms</i>	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

<i>applyTransform</i>	Controls whether or not the receiver's <code>RenderTransform</code> is applied to the bounds (if it has one) <ul style="list-style-type: none"> • 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
<i>applyClip</i>	Controls whether or not the receiver's <code>Clip</code> 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

<i>child</i>	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::getNodeTypes ( ) 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

<i>child</i>	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

<i>propertyName</i>	The name of the property.
<i>propertyValue</i>	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

<i>ptrPreviousSibling</i>	Smart pointer to the previous sibling node. If ptrPreviousSibling is NULL then child will be inserted as the first node
<i>child</i>	The child node to insert.
<i>bCheckComplete</i>	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

<i>ptrCandidate</i>	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

<i>func</i>	The function to call when this node is about to be destroyed
<i>priv</i>	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

<i>propertyName</i>	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

<i>oldChild</i>	Pointer to the child node to be replaced.
<i>newChild</i>	Pointer to the replacing node.

setDOMid()

```
virtual void IDOMNode::setDOMid (
    DOMid id ) [pure virtual]
```

Sets the node ID.

Parameters

<i>id</i>	The new DOM ID for the node.
-----------	------------------------------

setNextSibling()

```
virtual void IDOMNode::setNextSibling (
    const IDOMNodePtr & ptrNextSibling ) [pure virtual]
```

Sets the next sibling node.

Parameters

<i>ptrNextSibling</i>	Smart pointer to the new next sibling node.
-----------------------	---

setParentNode()

```
virtual void IDOMNode::setParentNode (
    const IDOMNodePtr & ptrParent ) [pure virtual]
```

Sets the parent node.

Parameters

<i>ptrParent</i>	Smart pointer to the new parent node.
------------------	---------------------------------------

setPreviousSibling()

```
virtual void IDOMNode::setPreviousSibling (
    const IDOMNodePtr & ptrPreviousSibling ) [pure virtual]
```

Sets the previous sibling node.

Parameters

<i>ptrPreviousSibling</i>	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

<i>propertyName</i>	The name of the property.
<i>propertyValue</i>	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

<i>func</i>	The function that was registered to be called when the node is destroyed.
<i>priv</i>	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

<i>func</i>	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.
<i>priv</i>	A bare pointer that is passed to func for each call.
<i>descendIntoBrushes</i>	If true, the walker will descend into composite brushes, only once per invocation.
<i>descendIntoForms</i>	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](#)

7.240 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.

7.240.1 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.

7.240.2 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

7.240.3 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

<i>position</i>	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

<i>position</i>	The bit position to set.
<i>value</i>	The bit value to set.

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

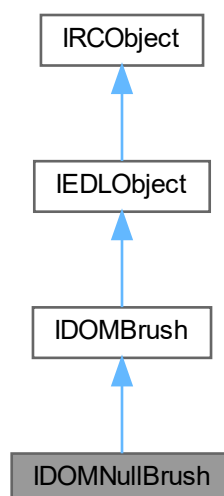
- [idomnode.h](#)

7.241 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.

7.241.1 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.

7.241.2 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

<i>pFactory</i>	The factory to use.
-----------------	---------------------

Returns

IDOMNullBrushPtr The new brush.

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

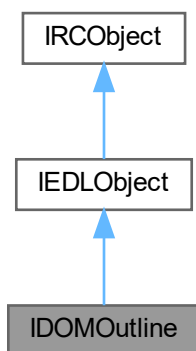
- [idombrush.h](#)

7.242 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.

7.242.1 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.

7.242.2 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

<i>pFactory</i>	The EDL factory to use
<i>language</i>	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

<i>strLanguage</i>	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

<i>strLanguage</i>	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:

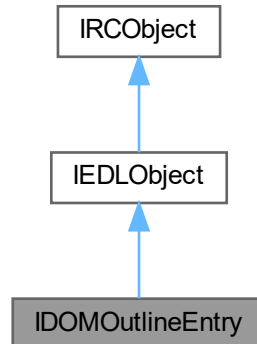
- idomoutline.h

7.243 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 &text←Color=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.

7.243.1 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.

7.243.2 Member Function Documentation

createNode()

```
static EDL_API IDOMOutlineTreeNodePtr IDOMOutlineEntry::createNode (
    IDLClassFactory * 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

<i>pFactory</i>	The EDL factory to use
<i>description</i>	The description of the outline entry
<i>expanded</i>	If the outline entry's children should be shown by default
<i>target</i>	The destination for the outline entry, or NULL for no target.
<i>textColor</i>	The color for the outline entry, or NULL for default. If provided, it must use the DeviceRGB color space.
<i>style</i>	The desired text style
<i>language</i>	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

<i>strDescription</i>	Reference parameter to receive the outline entry node description.
-----------------------	--

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

<i>strLanguage</i>	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

<i>ptrSE</i>	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

<i>ptrTarget</i>	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

<i>ptrColor</i>	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

<i>strDescription</i>	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

<i>expanded</i>	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

<i>strLanguage</i>	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

<i>ptrSE</i>	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

<i>ptrTarget</i>	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

<i>ptrColor</i>	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

<i>style</i>	The new value for the outline entry's text style.
--------------	---

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

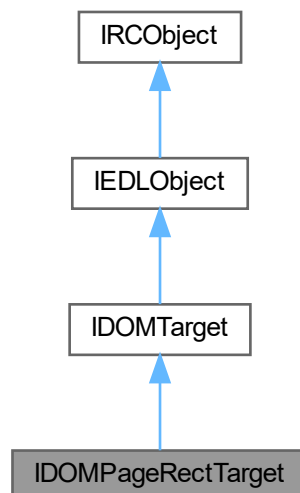
- idomoutline.h

7.244 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 [IDOMPPageRectTargetPtr](#) [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.

7.244.1 Detailed Description

[IDOMPPageRectTarget](#) nodes are used to describe hyperlinks on a page rectangle to targets on the same page.

7.244.2 Member Function Documentation

[create\(\)](#)

```
static EDL_API IDOMPPageRectTargetPtr IDOMPPageRectTarget::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

<i>pFactory</i>	The EDL factory to use
<i>pageId</i>	The DOM id of the target page
<i>fitType</i>	The fit type
<i>zoom</i>	The desired zoom level. Zero indicates that the zoom should be unchanged when navigating to this target.
<i>left</i>	The left of the target rect. For certain fit types, this will be ignored.
<i>top</i>	The top of the target rect. For certain fit types, this will be ignored.
<i>right</i>	The right of the target rect. For certain fit types, this will be ignored.
<i>bottom</i>	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

<i>bottom</i>	The new bottom coordinate value.
---------------	----------------------------------

setFitType()

```
virtual void IDOMPageRectTarget::setFitType (
    eFitType fitType ) [pure virtual]
```

Sets the fit type.

See also

[eFitType](#)

Parameters

<i>fitType</i>	The new fit type.
----------------	-------------------

setLeft()

```
virtual void IDOMPageRectTarget::setLeft (
    const doubleWithNull & left ) [pure virtual]
```

Sets the left coordinate value of the target.

Parameters

<i>left</i>	The new left coordinate value.
-------------	--------------------------------

setPageId()

```
virtual void IDOMPageRectTarget::setPageId (
    DOMid domId ) [pure virtual]
```

Sets the target page DOM id.

Parameters

<i>domId</i>	The new target page id
--------------	------------------------

setRight()

```
virtual void IDOMPageRectTarget::setRight (
    const doubleWithNull & right ) [pure virtual]
```

Sets the right coordinate value of the target.

Parameters

<i>right</i>	The new right coordinate value.
--------------	---------------------------------

setTop()

```
virtual void IDOMPageRectTarget::setTop (
    const doubleWithNull & top ) [pure virtual]
```

Sets the top coordinate value of the target.

Parameters

<i>top</i>	The new top coordinate value.
------------	-------------------------------

setZoom()

```
virtual void IDOMPageRectTarget::setZoom (
    double zoom ) [pure virtual]
```

Sets for the zoom value.

Parameters

<i>zoom</i>	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:

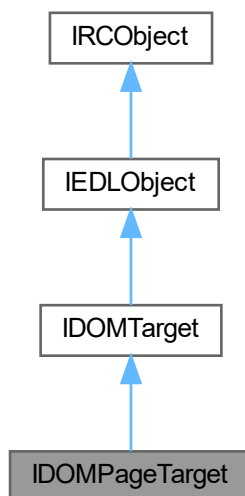
- idomtarget.h

7.245 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.

7.245.1 Detailed Description

[IDOMPageTarget](#) nodes are used to describe hyperlinks on a page to targets on the same page.

7.245.2 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

<i>page</i>	The new target page number.
-------------	-----------------------------

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

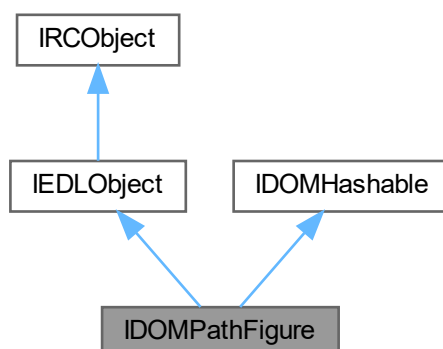
- idomtarget.h

7.246 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-that is, 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=CDOMPathSegmentVect())

Simplified creator for a path figure.

Additional Inherited Members

Protected Member Functions inherited from [IRCOBJECT](#)

- virtual `~IRCOBJECT ()`
Virtual destructor.

7.246.1 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](#)

7.246.2 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

<i>pathSegment</i>	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

<i>factory</i>	The factory to use.
<i>isClosed</i>	Should the figure be closed if stroking.
<i>isFilled</i>	Should the figure be filled, if used in a filling path.
<i>segments</i>	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-that is, 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

<i>closed</i>	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

<i>filled</i>	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

<i>sp</i>	New value of the start point of the first segment in the figure.
-----------	--

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

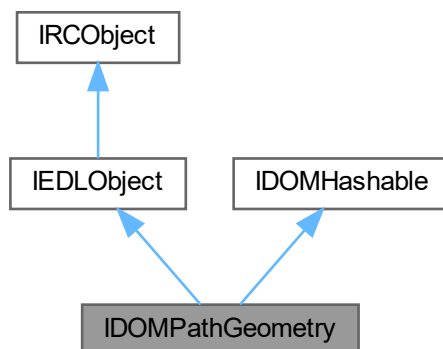
- idompathgeometry.h

7.247 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 bool [getIsRect](#) (FPoint &startPoint, FPoint &oppositePoint, bool &firstLineIsX, bool &closed)=0
Determines if the geometry is a simple rectangle, and if so, determines the start point, opposite point, and whether or not the first move in the rectangle is in x. 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 CDOMPathFigureVect &figures=CDOMPathFigureVect(), const FMatrix &renderTransform=FMatrix(), eFillRule fillRule=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.

7.247.1 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 figures.

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.

7.247.2 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

<i>pathFigure</i>	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

<i>factory</i>	The factory to use.
<i>figures</i>	The figures comprising the geometry.
<i>renderTransform</i>	The transform to apply to the geometry.
<i>fillRule</i>	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

<i>factory</i>	The factory to use.
<i>abbreviatedGeometry</i>	Path geometry string specified using XPS abbreviated geometry syntax.
<i>renderTransform</i>	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

<i>factory</i>	The factory to use.
<i>rect</i>	The rect to use.
<i>close</i>	Whether or not the rectangular path should be closed.
<i>renderTransform</i>	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

<i>factory</i>	The factory to use.
<i>rect</i>	The rect describing the bounds of the circle/ellipse.
<i>renderTransform</i>	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

<i>factory</i>	The factory to use.
<i>rect</i>	The rect describing the bounds of the polygon.
<i>numSides</i>	The desired number of sides.
<i>rotationAngle</i>	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

<i>applyTransform</i>	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

<i>pFactory</i>	A pointer to the EDL class factory.
<i>transform</i>	The external transformation applied to the geometry. Needed to determine the visible size of the geometry, and hence influences the level of curve approximation.
<i>resolution</i>	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() [1/2]

```
virtual bool IDOMPathGeometry::getIsRect (
    FPoint & startPoint,
    FPoint & oppositePoint,
    bool & firstLineIsX,
    bool & closed ) [pure virtual]
```

Determines if the geometry is a simple rectangle, and if so, determines the start point, opposite point, and whether or not the first move in the rectangle is in x. 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

<i>startPoint</i>	Reference to receive the start point of the rectangle.
<i>oppositePoint</i>	Reference to receive the opposite point of the rectangle.
<i>firstLineIsX</i>	Reference to receive whether or not the first line from the start point is in x or y.
<i>closed</i>	Reference to receive whether or not the geometry is closed.

Returns

bool. True if the geometry is a simple orthogonal rectangle.

getIsRect() [2/2]

```
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

<i>rect</i>	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

<i>factory</i>	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

<i>factory</i>	A pointer to the EDL class factory.
<i>transform</i>	The transform that should be applied before scan conversion.
<i>resolution</i>	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

<i>pFactory</i>	A pointer to the EDL class factory.
<i>simplifyQuads</i>	If true, any quadratic segments will be converted to regular cubic segments in the simplified geometry.
<i>simplifyArcs</i>	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

<i>fr</i>	Fill rule.
-----------	------------

setRenderTransform()

```
virtual void IDOMPathGeometry::setRenderTransform (
    const FMatrix & matrix ) [pure virtual]
```

Sets the render transform matrix for the path.

Parameters

<i>matrix</i>	The new render transform matrix.
---------------	----------------------------------

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

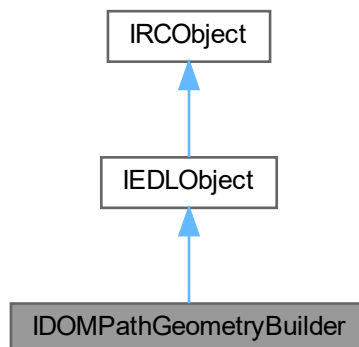
- idompathgeometry.h

7.248 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.

7.248.1 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.

7.248.2 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

<i>radius</i>	The x and y radius of the arc.
<i>rotation</i>	The rotation angle.
<i>isLargeArc</i>	Whether or not the large arc should be used.
<i>sweepDirection</i>	The direction of the sweep - clockwise or counter clockwise - between the start and end points.
<i>point</i>	The end point of the arc.
<i>absolute</i>	Whether the points are specified in absolute coordinates. The default value is true.
<i>stroked</i>	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

<i>stroked</i>	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

<i>factory</i>	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

<i>fillRule</i>	The fill rule.
<i>ignoreDegenerateSubpaths</i>	Whether to generate ignore degenerate subpaths. The default value is false.
<i>renderTransform</i>	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

<i>control0</i>	The first control point used to define the curve.
<i>control1</i>	The second control point used to define the curve.
<i>point</i>	The new end point.
<i>absolute</i>	Whether the points are specified in absolute coordinates. The default value is true.
<i>stroked</i>	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

<i>point</i>	The new end point.
<i>absolute</i>	Whether the points are specified in absolute coordinates. The default value is true.
<i>stroked</i>	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

<i>point</i>	The point to move to.
<i>absolute</i>	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

<i>control</i>	The control point used to define the curve.
<i>point</i>	The new end point.
<i>absolute</i>	Whether the points are specified in absolute coordinates. The default value is true.
<i>stroked</i>	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:

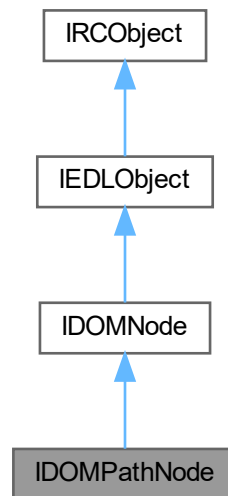
- idompathgeometry.h

7.249 IDOMPathNode Class Reference

Interface to an EDL path node. A path node specifies a geometry that can be filled or stroked 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 [getStrokeDashesCount](#) ()=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.
- virtual bool `getFillOverprints` () const =0
Retrieves whether or not the fill should overprint. Overprinting is honored only for rendering, PDF and PostScript output. It will also be honored should any output need to render as part of producing the output.
- virtual void `setFillOverprints` (bool overprint=true)=0
Sets whether or not the fill should overprint. Overprinting is honored only for rendering, PDF and PostScript output. It will also be honored should any output need to render as part of producing the output.
- virtual bool `getStrokeOverprints` () const =0
Retrieves whether or not the stroke should overprint. Overprinting is honored only for rendering, PDF and PostScript output. It will also be honored should any output need to render as part of producing the output.
- virtual void `setStrokeOverprints` (bool overprint=true)=0
Sets whether or not the stroke should overprint. Overprinting is honored only for rendering, PDF and PostScript output. It will also be honored should any output need to render as part of producing the output.
- virtual bool `getOverprintMode` () const =0
Retrieves whether or not overprint should obey PDF/PS overprint mode semantics. PDF and PostScript support a manner of overprinting for CMYK scenarios where an individual CMYK component may overprint if that channel is zero and the rendering target is also the same CMYK color space (excluding some transparency group scenarios). Setting overprint mode to true enables this feature.
- virtual void `setOverprintMode` (bool mode=true)=0
Sets whether or not overprint should obey PDF/PS overprint mode semantics. PDF and PostScript support a manner of overprinting for CMYK scenarios where an individual CMYK component may overprint if that channel is zero and the rendering target is also the same CMYK color space (excluding some transparency group scenarios). Setting overprint mode to true enables this feature.
- virtual bool `getStrokeAdjust` () const =0
Get whether or not any stroke should obey PDF/PS stroke adjust rules. PDF and PostScript support a method that intends to make any rendered strokes appear with a consistent width regardless of how the points are aligned with respect to pixels. This is a different method than that employed by `getSnapsToDevicePixels()/setSnapsToDevicePixels()` for XPS and XAML. Stroke adjustment is honored only for rendering, PDF and PostScript output. It will also be honored should any output need to render as part of producing the output.
- virtual void `setStrokeAdjust` (bool adjust=true)=0
Set whether or not any stroke should obey PDF/PS stroke adjust rules. PDF and PostScript support a method that intends to make any rendered strokes appear with a consistent width regardless of how the points are aligned with respect to pixels. This is a different method than that employed by `getSnapsToDevicePixels()/setSnapsToDevicePixels()` for XPS and XAML. Stroke adjustment is honored only for rendering, PDF and PostScript output. It will also be honored should any output need to render as part of producing the output.
- virtual `eRenderingIntent` `getRenderingIntent` () const =0

Get the rendering intent for the node if present. The rendering intent is used whenever the node requires color conversion to a different color space. Set `eRenderingIntentUnset` to unset the rendering intent, in which a default will be chosen based on the color space in use. This setting is honored only for rendering, color conversion, PDF or PostScript output.

- virtual void `setRenderingIntent (eRenderingIntent intent)=0`

Set or clear the rendering intent for the node. The rendering intent is used whenever the node requires color conversion to a different color space. Set `eRenderingIntentUnset` to unset the rendering intent, in which a default will be chosen based on the color space in use. This setting is honored only for rendering, color conversion, PDF or PostScript output.

- virtual `eBlackPointCompensation getBlackPointCompensation () const =0`

Get the black point compensation for the node. The black point compensation setting is used whenever the node requires color conversion to a different color space and affects how dark regions are handled. This setting is honored only for rendering, color conversion, PDF or PostScript output.

- virtual void `setBlackPointCompensation (eBlackPointCompensation compensation)=0`

Set the black point compensation for the node. The black point compensation setting is used whenever the node requires color conversion to a different color space and affects how dark regions are handled. This setting is honored only for rendering, color conversion, PDF or PostScript output.

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 bCheckComplete=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

Simplified 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 descendIntoForms=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 [IDOMPathGeometryPtr](#) &geometry, const [IDOMBrushPtr](#) &brush, const [FMatrix](#) renderTransform=[FMatrix](#)(), const [IDOMPathGeometryPtr](#) &clip=[IDOMPathGeometryPtr](#)(), float opacity=1.0f, [eBlendMode](#) blendMode=[eBlendModeNormal](#), [eEdgeMode](#) edgeMode=[eEMDefault](#))
Simplified creator for a filled path. Throws an IEDLError on failure.
- static [EDL_API](#) [IDOMPathNodePtr](#) [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=[eClipLongMitters](#), double dashOffset=0.0, const [CEDLVector](#)< double > dashPattern=[CEDLVector](#)< double >(), bool zeroWidthLinesAreVisible=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.

7.249.1 Detailed Description

Interface to an EDL path node. A path node specifies a geometry that can be filled or stroked 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.

7.249.2 Member Function Documentation

addStrokeDash()

```
virtual void IDOMPathNode::addStrokeDash (
    double value ) [pure virtual]
```

Append a stroke dash to the stroke dash array.

Parameters

<i>value</i>	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

<i>pFactory</i>	The factory to use.
<i>geometry</i>	The path data/geometry to fill.
<i>brush</i>	The fill brush to use.
<i>renderTransform</i>	The render transform to use.
<i>clip</i>	The geometry to use for clipping. NULL if no clip.
<i>opacity</i>	The opacity to use.
<i>blendMode</i>	The blend mode to use.
<i>edgeMode</i>	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

<i>pFactory</i>	The factory to use.
<i>image</i>	The image to use.
<i>viewPort</i>	The destination rectangle.
<i>renderTransform</i>	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

<i>pFactory</i>	The factory to use.
<i>geometry</i>	The path data/geometry to fill.
<i>brush</i>	The fill brush to use.
<i>renderTransform</i>	The render transform to use.
<i>clip</i>	The geometry to use for clipping. NULL if no clip.
<i>strokeThickness</i>	The stroke thickness. If this value is 0 and <code>zeroWidthLinesAreVisible</code> then treat the stroke as a hairline stroke.
<i>miterLimit</i>	The miter limit to be used, if applicable
<i>join</i>	The line join to use
<i>startCap</i>	The starting line cap
<i>endCap</i>	The ending line cap
<i>dashCap</i>	The line cap to be used for dashed sections
<i>miterTreatment</i>	How to treat miters that exceed the miter limit (see setStrokeMiterLimit() for details)
<i>dashOffset</i>	The dash offset to use if dashing is to be used
<i>dashPattern</i>	The dash pattern to be used. Pass an empty vector if no dashing is required. If <code>strokeThickness</code> is 0 then this is in user units, otherwise this is scaled up by the <code>strokeThickness</code> .
<i>zeroWidthLinesAreVisible</i>	Set to true if zero width lines should mark (see getShouldZeroWidthLinesBeVisible() for details)
<i>opacity</i>	The opacity to use.
<i>edgeMode</i>	The edge mode to use.
<i>blendMode</i>	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

getBlackPointCompensation()

```
virtual eBlackPointCompensation IDOMPathNode::getBlackPointCompensation ( ) const [pure virtual]
```

Get the black point compensation for the node. The black point compensation setting is used whenever the node requires color conversion to a different color space and affects how dark regions are handled. This setting is honored only for rendering, color conversion, PDF or PostScript output.

Returns

eBlackPointCompensation The black point compensation, or eBPCUnset if not present.

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.

getFillOverprints()

```
virtual bool IDOMPathNode::getFillOverprints ( ) const [pure virtual]
```

Retrieves whether or not the fill should overprint. Overprinting is honored only for rendering, PDF and PostScript output. It will also be honored should any output need to render as part of producing the output.

Returns

bool True if the fill should overprint, false otherwise.

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.

getOverprintMode()

```
virtual bool IDOMPathNode::getOverprintMode ( ) const [pure virtual]
```

Retrieves whether or not overprint should obey PDF/PS overprint mode semantics. PDF and PostScript support a manner of overprinting for CMYK scenarios where an individual CMYK component may overprint if that channel is zero and the rendering target is also the same CMYK color space (excluding some transparency group scenarios). Setting overprint mode to true enables this feature.

Returns

bool true if the overprint mode should be enabled, false otherwise.

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.

getRenderingIntent()

```
virtual eRenderingIntent IDOMPathNode::getRenderingIntent ( ) const [pure virtual]
```

Get the rendering intent for the node if present. The rendering intent is used whenever the node requires color conversion to a different color space. Set eRenderingIntentUnset to unset the rendering intent, in which a default will be chosen based on the color space in use. This setting is honored only for rendering, color conversion, PDF or PostScript output.

Returns

eRenderingIntent The rendering intent, or eRenderingIntentUnset if not 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

<i>transform</i>	The transform that should be applied before scan conversion.
<i>resolution</i>	The resolution that should be used for scan conversion.
<i>pFactory</i>	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. Used only XPS and XAML inputs and outputs.

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.

getStrokeAdjust()

```
virtual bool IDOMPathNode::getStrokeAdjust ( ) const [pure virtual]
```

Get whether or not any stroke should obey PDF/PS stroke adjust rules. PDF and PostScript support a method that intends to make any rendered strokes appear with a consistent width regardless of how the points are aligned with respect to pixels. This is a different method than that employed by [getSnapsToDevicePixels\(\)/setSnapsToDevicePixels\(\)](#) for XPS and XAML. Stroke adjustment is honored only for rendering, PDF and PostScript output. It will also be honored should any output need to render as part of producing the output.

Returns

bool true if stroke adjustment should be applied, false otherwise.

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

getStrokeDashesCount()

```
virtual uint32 IDOMPathNode::getStrokeDashesCount ( ) [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).

getStrokeOverprints()

```
virtual bool IDOMPathNode::getStrokeOverprints ( ) const [pure virtual]
```

Retrieves whether or not the stroke should overprint. Overprinting is honored only for rendering, PDF and PostScript output. It will also be honored should any output need to render as part of producing the output.

Returns

bool true if the stroke should overprint, false otherwise.

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

<i>helptext</i>	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

<i>propname</i>	The new automation properties name.
-----------------	-------------------------------------

setBlackPointCompensation()

```
virtual void IDOMPathNode::setBlackPointCompensation (
    eBlackPointCompensation compensation ) [pure virtual]
```

Set the black point compensation for the node. The black point compensation setting is used whenever the node requires color conversion to a different color space and affects how dark regions are handled. This setting is honored only for rendering, color conversion, PDF or PostScript output.

Parameters

<i>compensation</i>	The desired black point compensation setting, or eBPCUnset to clear.
---------------------	--

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

<i>blendMode</i>	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

<i>ptrClip</i>	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

<i>em</i>	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

<i>ptrFill</i>	Smart pointer to the new fill brush, or NULL to clear.
----------------	--

setFillOverprints()

```
virtual void IDOMPathNode::setFillOverprints (
    bool overprint = true ) [pure virtual]
```

Sets whether or not the fill should overprint. Overprinting is honored only for rendering, PDF and PostScript output. It will also be honored should any output need to render as part of producing the output.

Parameters

<i>overprint</i>	The desired overprint.
------------------	------------------------

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

<i>lang</i>	Default language
-------------	------------------

setNavigateLink()

```
virtual void IDOMPathNode::setNavigateLink (  
    const IDOMTargetPtr & target ) [pure virtual]
```

Sets the target of a hyperlink.

Parameters

<i>target</i>	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

<i>opc</i>	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

<i>ptrOpacityMask</i>	Smart pointer to brush to set, or NULL to clear.
-----------------------	--

setOverprintMode()

```
virtual void IDOMPathNode::setOverprintMode (
    bool mode = true ) [pure virtual]
```

Sets whether or not overprint should obey PDF/PS overprint mode semantics. PDF and PostScript support a manner of overprinting for CMYK scenarios where an individual CMYK component may overprint if that channel is zero and the rendering target is also the same CMYK color space (excluding some transparency group scenarios). Setting overprint mode to true enables this feature.

Parameters

<i>mode</i>	The desired overprint mode
-------------	----------------------------

setPathData()

```
virtual void IDOMPathNode::setPathData (
    const IDOMPathGeometryPtr & ptrPathData ) [pure virtual]
```

Sets the path geometry node.

Parameters

<i>ptrPathData</i>	Smart pointer to the new path geometry node.
--------------------	--

setRenderingIntent()

```
virtual void IDOMPathNode::setRenderingIntent (
    eRenderingIntent intent ) [pure virtual]
```

Set or clear the rendering intent for the node. The rendering intent is used whenever the node requires color conversion to a different color space. Set `eRenderingIntentUnset` to unset the rendering intent, in which a default will be chosen based on the color space in use. This setting is honored only for rendering, color conversion, PDF or PostScript output.

Parameters

<i>intent</i>	The desired rendering intent, or <code>eRenderingIntentUnset</code> to clear.
---------------	---

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

<i>matrix</i>	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

<i>visible</i>	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. Used only XPS and XAML inputs and outputs.

Parameters

<i>s2dp</i>	New value of <code>snapsToDevicePixels</code> .
-------------	---

setStroke()

```
virtual void IDOMPathNode::setStroke (
    const IDOMBrushPtr & ptrStroke ) [pure virtual]
```

Sets the stroke brush for the path.

Parameters

<i>ptrStroke</i>	Smart pointer to the new stroke brush, or NULL to clear.
------------------	--

setStrokeAdjust()

```
virtual void IDOMPathNode::setStrokeAdjust (
    bool adjust = true ) [pure virtual]
```

Set whether or not any stroke should obey PDF/PS stroke adjust rules. PDF and PostScript support a method that intends to make any rendered strokes appear with a consistent width regardless of how the points are aligned with respect to pixels. This is a different method than that employed by [getSnapsToDevicePixels\(\)/setSnapsToDevicePixels\(\)](#) for XPS and XAML. Stroke adjustment is honored only for rendering, PDF and PostScript output. It will also be honored should any output need to render as part of producing the output.

Parameters

<i>adjust</i>	The desired stroke adjust setting.
---------------	------------------------------------

setStrokeDashLineCap()

```
virtual void IDOMPathNode::setStrokeDashLineCap (
    eStrokeLineCap slc ) [pure virtual]
```

Sets the stroke dash line cap.

See also

[eStrokeLineCap](#)

Parameters

<i>slc</i>	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

<i>sdo</i>	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

<i>pattern</i>	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

<i>slc</i>	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

<i>slj</i>	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

<i>sm</i>	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

<i>treatment</i>	The desired stroke miter treatment.
------------------	-------------------------------------

setStrokeOverprints()

```
virtual void IDOMPathNode::setStrokeOverprints (
    bool overprint = true ) [pure virtual]
```

Sets whether or not the stroke should overprint. Overprinting is honored only for rendering, PDF and PostScript output. It will also be honored should any output need to render as part of producing the output.

Parameters

<i>overprint</i>	The desired overprint.
------------------	------------------------

setStrokeStartLineCap()

```
virtual void IDOMPathNode::setStrokeStartLineCap (
    eStrokeLineCap slc ) [pure virtual]
```

Sets the line cap type for the start of the stroke.

See also

[eStrokeLineCap](#)

Parameters

<i>slc</i>	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

<i>st</i>	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

<i>pFactory</i>	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:

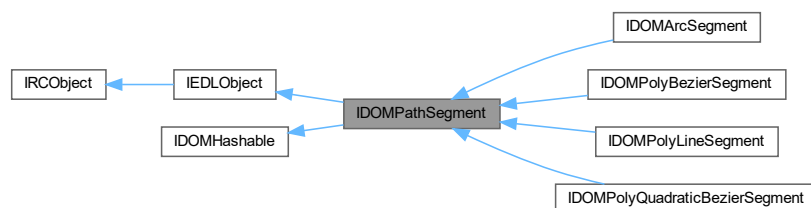
- idompath.h

7.250 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 [IEDLObject](#).

Public Member Functions inherited from [IRCOBJECT](#)

- virtual void [addRef](#) () const =0
Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool [decRef](#) () const =0
Decreases the reference count of the actual object pointed to. When the reference count falls to 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.

7.250.1 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.

7.250.2 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

<i>startPoint</i>	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 IDOMPPathSegment::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 IDOMPPathSegment::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

<i>isStroked</i>	New value of IsStroked.
------------------	-------------------------

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

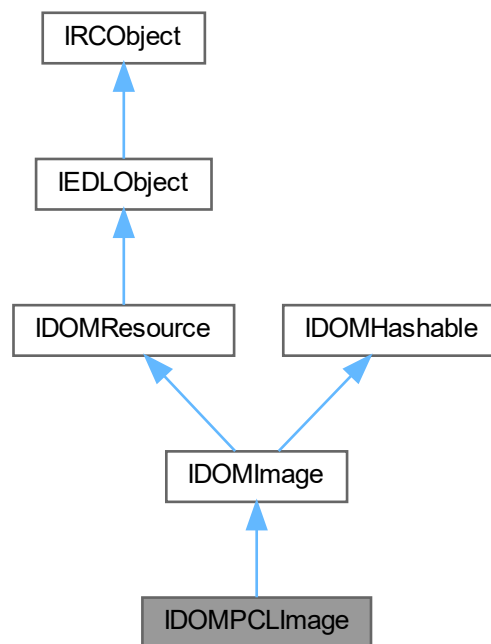
- idompathgeometry.h

7.251 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 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 [IEDLObject](#).*

Public Member Functions inherited from [IRCOBJECT](#)

- virtual `void` [addRef](#) () const =0
 - Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.*
- virtual `bool` [decRef](#) () const =0
 - Decreases the reference count of the actual object pointed to. When the reference count falls to 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.

7.251.1 Detailed Description

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

7.251.2 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:

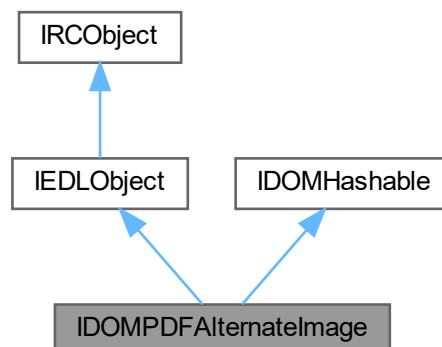
- `idomimageresource.h`

7.252 IDOMPDFAlternatImage 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 IDOMPDFAlternatImage:



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 IDOMPdFAlternatImagePtr [create](#) ([IEDLClassFactory](#) *pFactory, const IDOMImageBrushPtr &alternateBrush, bool defaultForPrinting=false, const JawsMako::IOptionalContentDetailsPtr &optionalContentDetails=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 [IDOMPdFAlternatImage](#).

Additional Inherited Members

Protected Member Functions inherited from [IRCObject](#)

- virtual `~IRCObject ()`
Virtual destructor.

7.252.1 Detailed Description

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

7.252.2 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

<i>pFactory</i>	The class factory to use
<i>alternateBrush</i>	The alternate image brush itself. It must not have any alternate images of its own
<i>defaultForPrinting</i>	Set true if this image should be the default image when printing
<i>optionalContentDetails</i>	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 details, or NULL if not optional

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

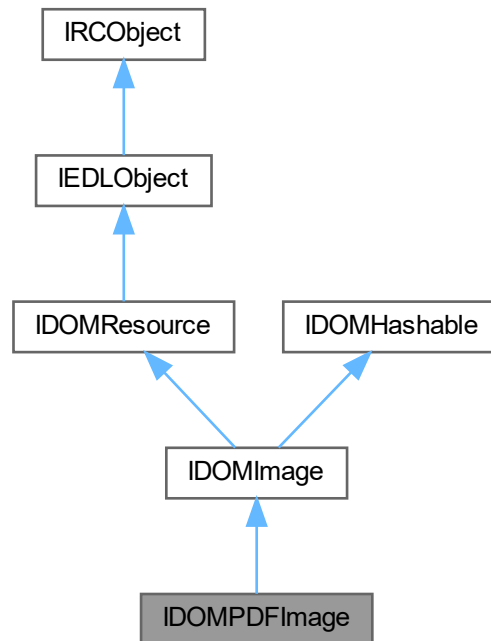
- idomimageresource.h

7.253 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 elmageAlpha [getAlphaDetails](#) () const =0
Returns if the image has alpha, and if the color 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 IDOMPDIImage).
- 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 const [CClassID](#) & [classID](#) ()
Retrieves class id of [IDOMPdfImage](#).

Additional Inherited Members

Protected Member Functions inherited from [IRCOBJECT](#)

- virtual [~IRCOBJECT](#) ()
Virtual destructor.

7.253.1 Detailed Description

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

7.253.2 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 color 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

<i>index</i>	Index of the desired decode parameters in the array to return (beginning at 0).
--------------	---

Returns

IDecodeParamsPtr The decode parameters

getImageTypes()

```
virtual CEDLVector< eDOMImageType > IDOMPNGImage::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:

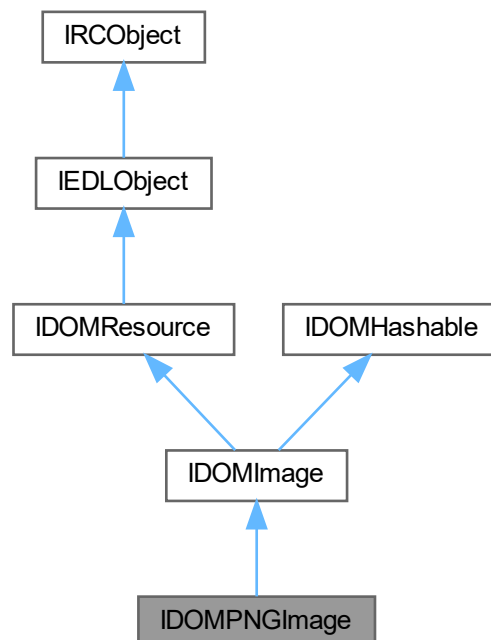
- idomimageresource.h

7.254 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=IOutputStreamPtr())
Create an IDOMPNGImage and frame that can be used to populate same.
- static EDL_API void [encode](#) (const ISessionPtr &pSession, const IDOMImagePtr &image, const IOutputStreamPtr &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 IOutputStreamPtr &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 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 `InputStreamPtr` `getStream` () const =0
Retrieves the resource stream.
- virtual void `setStream` (const `InputStreamPtr` &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.

7.254.1 Detailed Description

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

7.254.2 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

<i>pFactory</i>	The EDL Class factory to use.
<i>stream</i>	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

<i>session</i>	The session to use
<i>frame</i>	On exit, this is populated with a frame ready to receive image data via <code>frame->writeScanLine()</code> . Use <code>frame->flushData()</code> to complete the encoding process.
<i>colorSpace</i>	The color space to use. Must be compatible with the PNG format.
<i>width</i>	The width of the image, in pixels.
<i>height</i>	The height of the image, in pixels.
<i>bitsPerComponent</i>	The bits per component to use. Must be compatible with the PNG format.
<i>xResolution</i>	The x resolution, in pixels-per-inch.
<i>yResolution</i>	The y resolution, in pixels-per-inch.
<i>extraChannel</i>	The type of extra channel, if provided. Must be either <code>eIECAlpha</code> or <code>eIECNone</code> for PNG.
<i>inStream</i>	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, <i>outStream</i> must also be provided.
<i>outStream</i>	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, <i>inStream</i> 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

<i>pSession</i>	The relevant EDL session
<i>colorSpace</i>	The color space of the frame buffer
<i>dpi</i>	The image resolution in dots-per-inch
<i>bpc</i>	Bits per component (pixel)
<i>frameBuffer</i>	The frame buffer to be encoded
<i>width</i>	The width of the frame buffer
<i>height</i>	The height of the frame buffer
<i>stride</i>	The offset in bytes in the <i>frameBuffer</i> from one scanline to the next. May be negative.
<i>hasAlpha</i>	Set true if the frame buffer has an alpha channel.
<i>stream</i>	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 IOStreamPtr & 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

<i>pSession</i>	The relevant EDL session
<i>image</i>	The image to be encoded
<i>stream</i>	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 IOStreamPtr & 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

<i>pSession</i>	The relevant EDL session
<i>frame</i>	The frame providing the source image data
<i>stream</i>	The stream to use to store the image data.

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

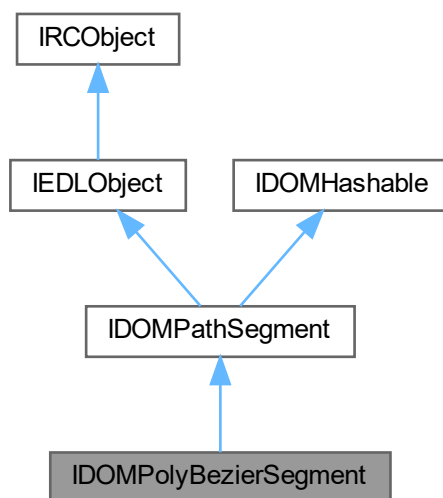
- idomimageresource.h

7.255 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 `CPointVect & 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 CFPPointVect &points=CFPointVect())`
Simplified creator for a bezier segment.

Additional Inherited Members

Protected Member Functions inherited from `IRCOBJECT`

- virtual `~IRCOBJECT ()`
Virtual destructor.

7.255.1 Detailed Description

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

Bézier curves are drawn from the previous point in the path figure or the previous Bézier segment and terminate at the third point (x_{3n}, y_{3n}) in the `Points` attribute (where n is the curve being drawn). The tangents and curvature of each Bézier curve are controlled by the first two control points (x_{3n-2}, y_{3n-2} and x_{3n-1}, y_{3n-1}) in the `Points` attribute. The `Points` attribute contains a multiple of three whitespace delimited pairs of comma delimited x, y values.

7.255.2 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

<i>point</i>	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

<i>factory</i>	The factory to use.
<i>isStroked</i>	Should the segment be stroked, if used in a stroking path.
<i>points</i>	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:

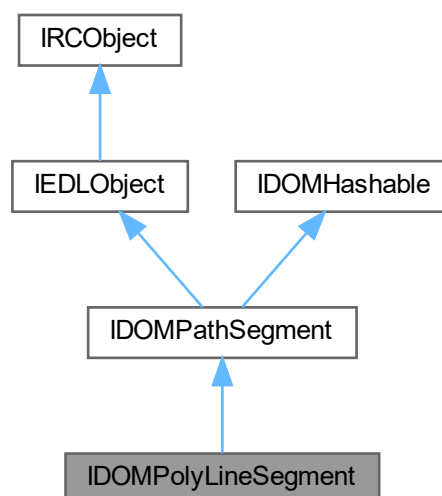
- idompathgeometry.h

7.256 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 CFPPointVect & [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.

7.256.1 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.

7.256.2 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

<i>point</i>	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 CFPPointVect & points = CFPPointVect() ) [static]
```

Simplified creator for a line segment.

Parameters

<i>factory</i>	The factory to use.
<i>isStroked</i>	Should the segment be stroked, if used in a stroking path.
<i>points</i>	The segment points.

Returns

IDOMPolyLineSegmentPtr The new segment.

getPoints()

```
virtual const CFPPointVect & 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:

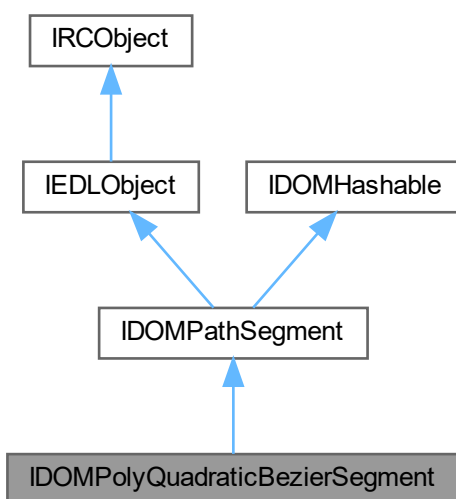
- idompathgeometry.h

7.257 IDOMPolyQuadraticBezierSegment Class Reference

Interface to a polyquadratic Bézier segment. A polyquadratic 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 (x_{2n-1} , y_{2n-1}) followed by the end point (x_{2n} , y_{2n}) 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 is↵
Stroked=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.

7.257.1 Detailed Description

Interface to a polyquadratic Bézier segment. A polyquadratic Bézier segment describes a set of quadratic Bézier curves from the starting point defined in the [IDOMPPathFigure](#), 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 (x_{2n-1}, y_{2n-1}) followed by the end point (x_{2n}, y_{2n}) for each quadratic Bézier curve (where n represents the quadratic Bézier curve).

7.257.2 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

<i>point</i>	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

<i>factory</i>	Pointer to the class factory to be used.
<i>startPoint</i>	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

<i>factory</i>	The factory to use.
<i>isStroked</i>	Should the segment be stroked, if used in a stroking path.
<i>points</i>	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:

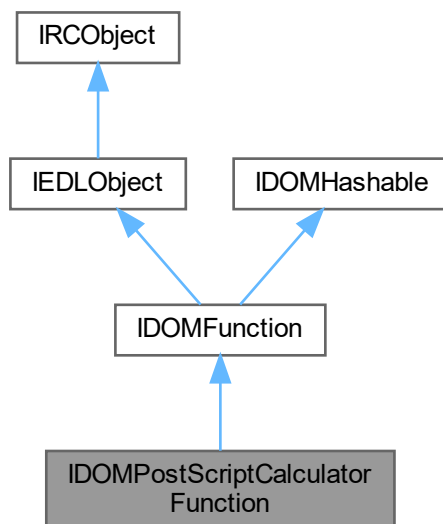
- idompathgeometry.h

7.258 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 `InputStreamPtr` [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.

7.258.1 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.

7.258.2 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:

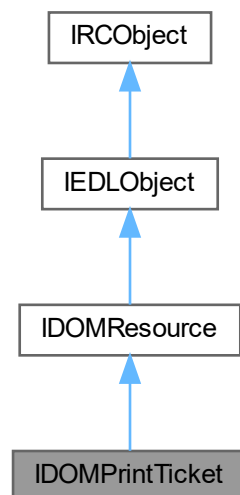
- idomfunction.h

7.259 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.

7.259.1 Detailed Description

[IDOMPrintTicket](#) interface.

7.259.2 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:

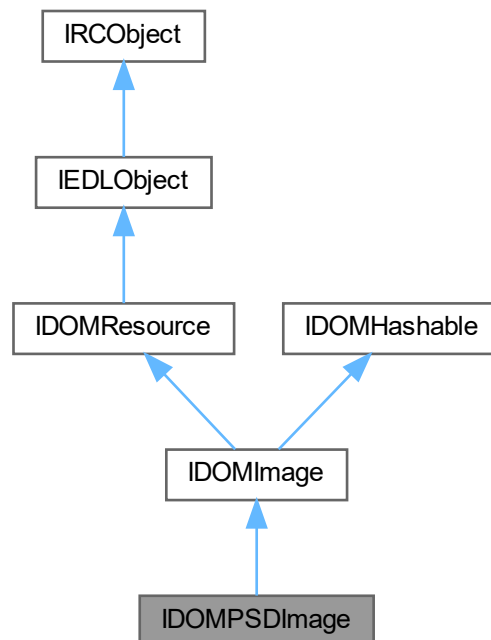
- [idomresources.h](#)

7.260 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 [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 [IDOMPSPImagePtr](#) [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 [IDOMPSPImage](#).

Additional Inherited Members

Protected Member Functions inherited from [IRCOBJECT](#)

- virtual [~IRCOBJECT](#) ()
Virtual destructor.

7.260.1 Detailed Description

[IDOMPSPImage](#) interface.

7.260.2 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

<i>pFactory</i>	The EDL Class factory to use.
<i>stream</i>	The stream containing the PSD image.
<i>layerIndex</i>	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:

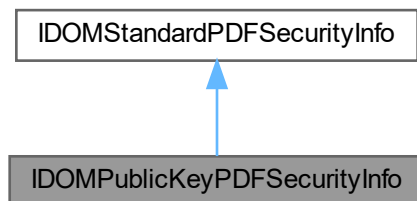
- idomimageresource.h

7.261 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.

7.261.1 Detailed Description

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

7.261.2 Member Enumeration Documentation

ePublicKeyPermissionsFlags

enum [IDOMPublicKeyPDFSecurityInfo::ePublicKeyPermissionsFlags](#)

Enumerator

eChangeEncryptionAllowed	Allow encryption parameters to be changed. Effectively grants ownership access.
--------------------------	---

7.261.3 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:

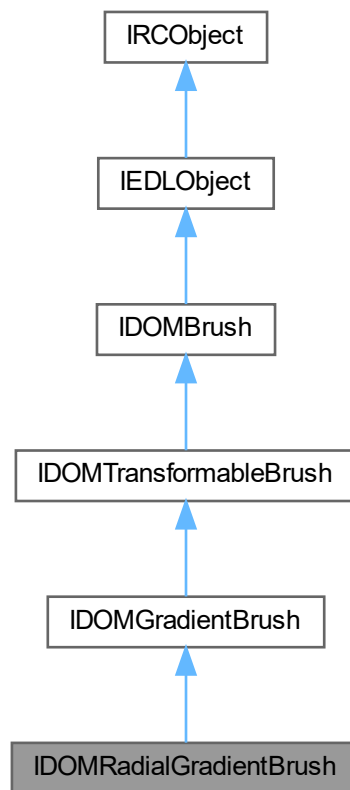
- idomsecurity.h

7.262 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 &fill←Area, 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 &fill←Area)=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 &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](#) [IDOMRadialGradientBrushPtr](#) [create](#) ([IEDLClassFactory](#) *pFactory, const [FPoint](#) ¢er, 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.

7.262.1 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.

7.262.2 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

<i>pFactory</i>	The EDL Class factory to use.
<i>center</i>	The center point of the gradient brush ellipse
<i>gradientOrigin</i>	The origin point of the gradient
<i>radiusX</i>	The x radius of the gradient brush ellipse
<i>radiusY</i>	The y radius of the gradient brush ellipse
<i>stops</i>	The gradient stops
<i>opacity</i>	The alpha to use.
<i>renderTransform</i>	The render transform to use
<i>spreadMethod</i>	The desired spread method
<i>colorInterpolationMode</i>	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

<i>pFactory</i>	The EDL class factory
<i>fillArea</i>	The rectangular area that needs to be filled by the resulting brush.
<i>useFirstStopColorSpace</i>	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

<i>pFactory</i>	The EDL class factory
<i>fillArea</i>	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

<i>pt</i>	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

<i>pt</i>	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

<i>r</i>	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

<i>r</i>	The y-radius of the gradient brush ellipse.
----------	---

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

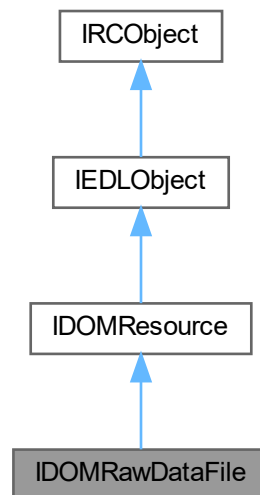
- [idombrush.h](#)

7.263 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.

7.263.1 Detailed Description

[IDOMRawDataFile](#) interface.

7.263.2 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:

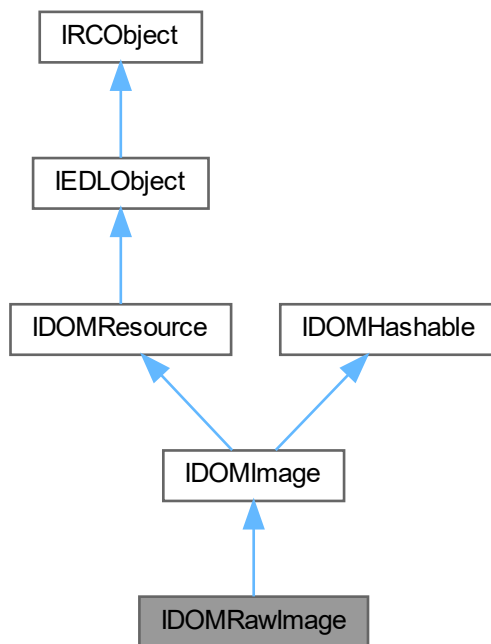
- [idomresources.h](#)

7.264 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=IOutputStreamPtr())`
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.

7.264.1 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.

7.264.2 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

<i>pFactory</i>	The EDL Class factory to use.
<i>colorSpace</i>	The color space of the data
<i>stream</i>	The stream containing the raw image.
<i>type</i>	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

<i>session</i>	The session to use
<i>frame</i>	On exit, this is populated with a frame ready to receive image data via <code>frame->writeScanLine()</code> . Use <code>frame->flushData()</code> to complete the encoding process.
<i>colorSpace</i>	The color space to use. Any valid color space may be used for IDOMRawImages.
<i>width</i>	The width of the image, in pixels.
<i>height</i>	The height of the image, in pixels.
<i>bitsPerComponent</i>	The bits per component to use. 1, 2, 4, 8, 12 and 16 bits per sample are supported.
<i>xResolution</i>	The x resolution, in pixels-per-inch.
<i>yResolution</i>	The y resolution, in pixels-per-inch.
<i>extraChannel</i>	The type of extra channel, if provided.
<i>inStream</i>	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.
<i>outStream</i>	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:

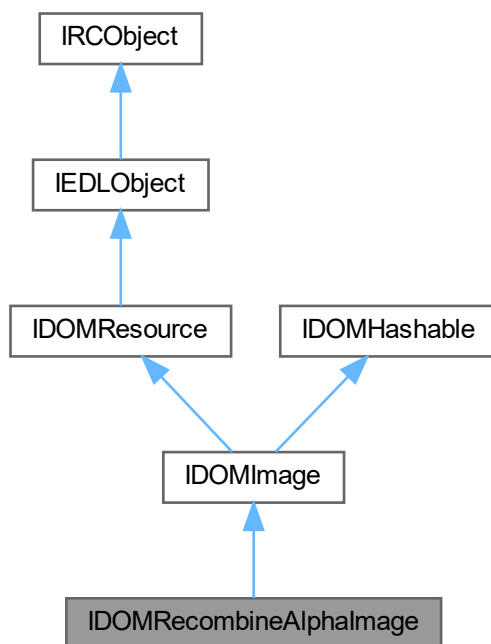
- idomimageresource.h

7.265 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 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 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 extraChannellsMask=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.

7.265.1 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.

7.265.2 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

<i>pFactory</i>	The EDL Class Factory
<i>colorImage</i>	The image with the color components
<i>extraChannel</i>	The image with the mask/alpha channel
<i>extraChannelIsMask</i>	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

<i>stream</i>	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

<i>stream</i>	A smart pointer to the stream
---------------	-------------------------------

Implements [IDOMResource](#).

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

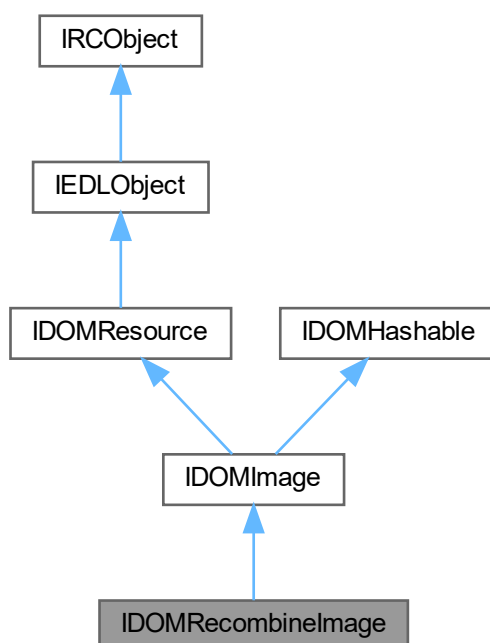
- idomimageresource.h

7.266 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 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 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 extraChannellsMask=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.

7.266.1 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.

7.266.2 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

<i>pFactory</i>	The class factory to use
<i>colorSpace</i>	The color space that describes the color channels
<i>componentImages</i>	A vector of images each providing a single color channel
<i>extraChannel</i>	The image to be used as an alpha channel or mask, if required
<i>extraChannellsMask</i>	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

<i>stream</i>	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

<i>stream</i>	A smart pointer to the stream
---------------	-------------------------------

Implements [IDOMResource](#).

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

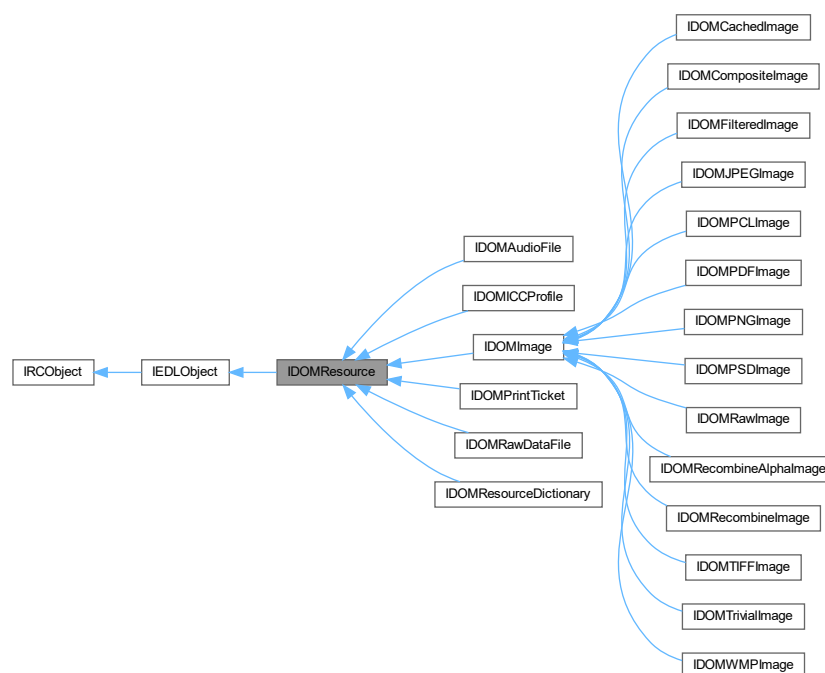
- idomimageresource.h

7.267 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 `InputStreamPtr` `getStream` () const =0
Retrieves the resource stream.
- virtual void `setStream` (const `InputStreamPtr` &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.

7.267.1 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.

7.267.2 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.

Implemented in [IDOMTrivialImage](#).

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

<i>stream</i>	Smart pointer to the new resource stream.
---------------	---

Implemented in [IDOMTrivialImage](#), [IDOMRecombineImage](#), [IDOMRecombineAlphaImage](#), [IDOMCompositelImage](#), [IDOMCachedImage](#), and [IDOMFilteredImage](#).

setUri()

```
virtual void IDOMResource::setUri (
    const EDLSysString & uri ) [pure virtual]
```

Sets the resource URI.

Parameters

<i>uri</i>	The new resource URI.
------------	-----------------------

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

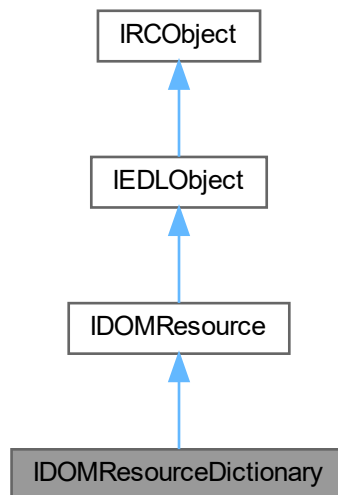
- [idomresources.h](#)

7.268 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.

7.268.1 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.

7.268.2 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

<i>name</i>	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

<i>name</i>	The name of the resource to add.
<i>element</i>	Smart pointer to an IEDLObject representing the resource.

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

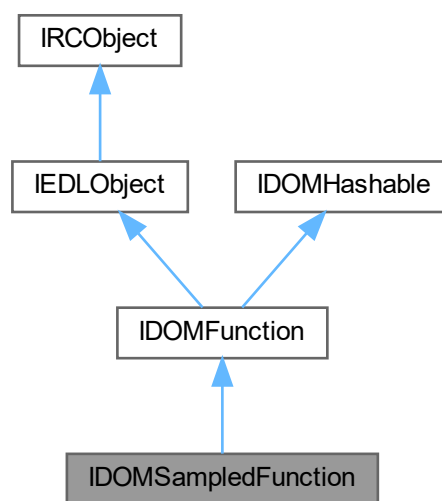
- [idomresources.h](#)

7.269 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.

7.269.1 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.

7.269.2 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

<i>inputNum</i>	The 0-indexed input number
<i>low</i>	A reference to receive the low encode bound for the given inputNum
<i>high</i>	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

<i>outputNum</i>	The 0-indexed output number
<i>low</i>	A reference to receive the low decode bound for the given outputNum
<i>high</i>	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

<code>inputNum</code>	The 0-indexed input number
-----------------------	----------------------------

Returns

uint32 The dimension

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

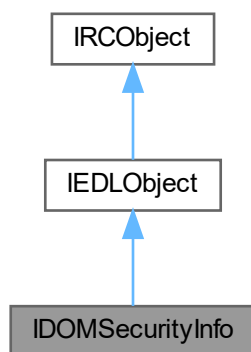
- idomfunction.h

7.270 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.

7.270.1 Detailed Description

Base DOM security class.

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

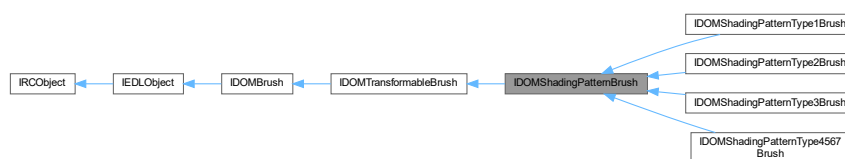
- idomsecurity.h

7.271 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.

7.271.1 Detailed Description

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

7.271.2 Member Function Documentation

[delIndex\(\)](#)

```
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

<i>pFactory</i>	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

<i>bBox</i>	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

<i>antiAlias</i>	New value of anti aliasing flag
------------------	---------------------------------

setBackground-color()

```
virtual void IDOMShadingPatternBrush::setBackground-color (
    const IDOMColorPtr & color ) [pure virtual]
```

Sets the background color to use before painting the shade.

Parameters

<i>color</i>	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

<i>bBox</i>	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

<i>colorSpace</i>	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

<i>pFactory</i>	A pointer to an EDL class factory
<i>colorSpace</i>	The smart pointer to the color space. Must not be a complex or composite color space such as Indexed or DeviceN.
<i>intent</i>	The rendering intent to use for conversion.
<i>bpc</i>	Black point compensation treatment. If in doubt, use eBPCDefault.

setFunction()

```
virtual void IDOMShadingPatternBrush::setFunction (
    const IDOMFunctionPtr & function ) [pure virtual]
```

Sets the shade function.

Parameters

<i>function</i>	Smart pointer to a function object
-----------------	------------------------------------

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

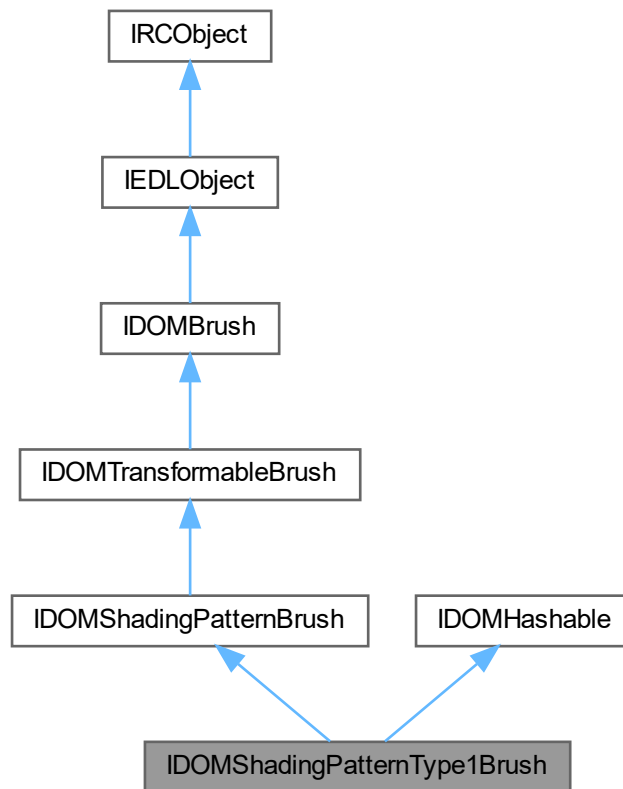
- [idombrush.h](#)

7.272 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 [delIndex](#) ([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.

7.272.1 Detailed Description

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

7.272.2 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

<i>pFactory</i>	The factory to use.
<i>colorSpace</i>	The color space for the shading pattern
<i>domain</i>	The domain of the matrix x and y axis of the shade
<i>function</i>	The function describing the color of the shade over the range of the domain
<i>hasBBox</i>	true if the bBox parameter should be used
<i>bBox</i>	The shading bounding box
<i>background</i>	The background color to use, or NULL for no background
<i>matrix</i>	The shading matrix to use
<i>renderTransform</i>	The desired render transform.
<i>antiAlias</i>	Whether anti-aliasing should be used for this shade or not
<i>opacity</i>	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

<i>domain</i>	Array of 4 floats
---------------	-------------------

setMatrix()

```
virtual void IDOMShadingPatternType1Brush::setMatrix (
    const FMatrix & matrix ) [pure virtual]
```

Sets the Type 1 shade matrix.

Parameters

<i>matrix</i>	Reference to a matrix object
---------------	------------------------------

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

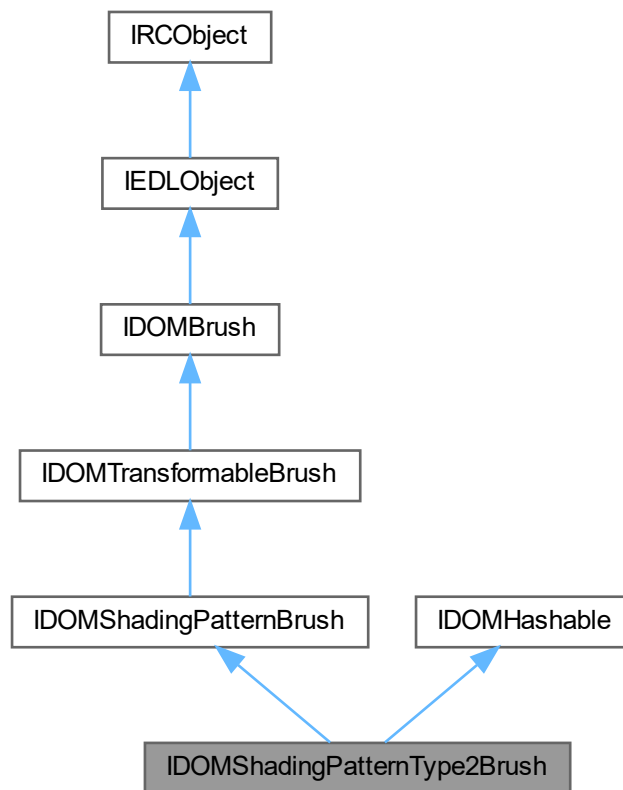
- [idombrush.h](#)

7.273 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 **IRCOject**

- virtual void **addRef** () const =0
Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool **decRef** () const =0
Decreases the reference count of the actual object pointed to. When the reference count falls to 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 &render←Transform=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.

7.273.1 Detailed Description

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

7.273.2 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

<i>pFactory</i>	The factory to use.
<i>startPoint</i>	The starting point for the shade
<i>endPoint</i>	The end point for the shade
<i>colorSpace</i>	The color space for the shading pattern
<i>domain</i>	The domain of the shading variable
<i>function</i>	The function describing the color of the shade over the range of the domain
<i>extendStart</i>	True if the shade should be extended before the start point (as per linear gradient pad mode)
<i>extendEnd</i>	True if the shade should be extended beyond the end point (as per linear gradient pad mode)
<i>hasBBox</i>	true if the bBox parameter should be used
<i>bBox</i>	The shading bounding box
<i>background</i>	The background color to use, or NULL for no background
<i>renderTransform</i>	The desired render transform.
<i>antiAlias</i>	Whether anti-aliasing should be used for this shade or not
<i>opacity</i>	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

<i>pFactory</i>	A pointer to an EDL class factory
<i>maxSamples</i>	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

<i>extendStart</i>	Reference to bool parameter representing the flags for the start point
<i>extendEnd</i>	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

<i>domain</i>	Array of 2 floats
---------------	-------------------

setEndPoint()

```
virtual void IDOMShadingPatternType2Brush::setEndPoint (
    const FPoint & point ) [pure virtual]
```

Set the end point.

Parameters

<i>point</i>	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

<i>extendStart</i>	parameter representing the flags for the start point
<i>extendEnd</i>	parameter representing the flags for the end point

setStartPoint()

```
virtual void IDOMShadingPatternType2Brush::setStartPoint (
    const FPoint & point ) [pure virtual]
```

Sets the start point.

Parameters

<i>point</i>	The start point
--------------	-----------------

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

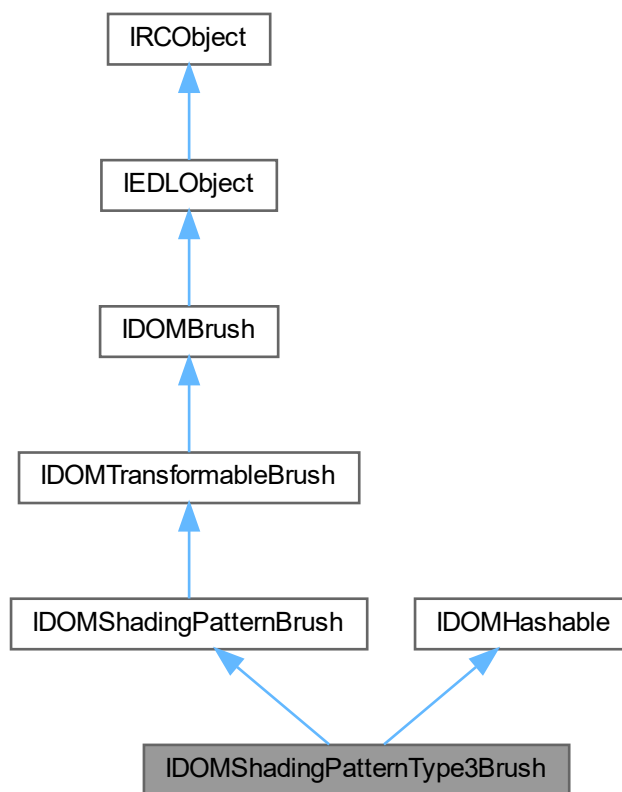
- [idombrush.h](#)

7.274 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 [IEDLObject](#).

Public Member Functions inherited from IRCObject

- virtual void [addRef](#) () const =0
Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool [decRef](#) () const =0
Decreases the reference count of the actual object pointed to. When the reference count falls to 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 [IDOMColorPtr](#) &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.

7.274.1 Detailed Description

[IDOMShadingPatternType3Brush](#) provides a way of representing a PS style type 2 shading pattern.

7.274.2 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

<i>pFactory</i>	The factory to use.
<i>startCircleCenter</i>	The center of the starting circle for the shade
<i>startCircleRadius</i>	The radius of the starting circle for the shade
<i>endCircleCenter</i>	The center of the starting circle for the shade
<i>endCircleRadius</i>	The radius of the starting circle for the shade
<i>colorSpace</i>	The color space for the shading pattern
<i>domain</i>	The domain of the shading variable
<i>function</i>	The function describing the color of the shade over the range of the domain
<i>extendStart</i>	True if the shade should be extended before the starting circle
<i>extendEnd</i>	True if the shade should be extended beyond the ending circle
<i>hasBBox</i>	true if the bBox parameter should be used
<i>bBox</i>	The shading bounding box
<i>background</i>	The background color to use, or NULL for no background
<i>renderTransform</i>	The desired render transform.
<i>antiAlias</i>	Whether anti-aliasing should be used for this shade or not
<i>opacity</i>	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

<i>pFactory</i>	A pointer to an EDL class factory
<i>maxSamples</i>	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

<i>extendStart</i>	Reference to boolean parameter representing the flags for the starting circle
<i>extendEnd</i>	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 starting circle radius

setDomain()

```
virtual void IDOMShadingPatternType3Brush::setDomain (
    float domain[2] ) [pure virtual]
```

Sets the domain range.

Parameters

<i>domain</i>	Array of 2 floats
---------------	-------------------

setEndCircleCenter()

```
virtual void IDOMShadingPatternType3Brush::setEndCircleCenter (
    const FPoint & point ) [pure virtual]
```

Sets the center of the ending circle.

Parameters

<i>point</i>	The ending circle center
--------------	--------------------------

setEndCircleRadius()

```
virtual void IDOMShadingPatternType3Brush::setEndCircleRadius (
    float radius ) [pure virtual]
```

Set the radius of the ending circle.

Parameters

<i>radius</i>	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

<i>extendStart</i>	Boolean parameter representing the flags for the starting circle
<i>extendEnd</i>	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

<i>point</i>	The starting circle center
--------------	----------------------------

setStartCircleRadius()

```
virtual void IDOMShadingPatternType3Brush::setStartCircleRadius (
    float radius ) [pure virtual]
```

Set the radius of the starting circle.

Parameters

<i>radius</i>	The starting circle radius
---------------	----------------------------

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

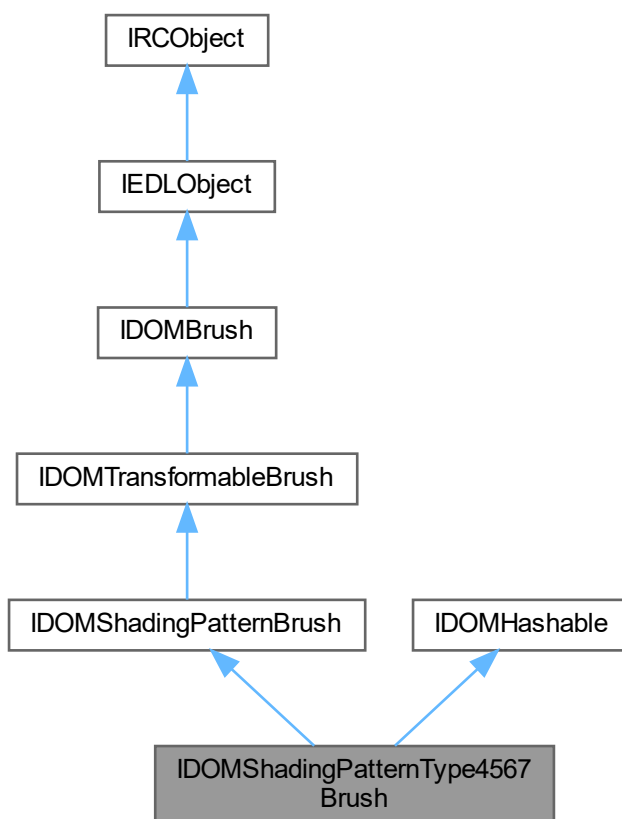
- [idombrush.h](#)

7.275 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 [delIndex](#) (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 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)`
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 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)`
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.

7.275.1 Detailed Description

[IDOMShadingPatternType4567Brush](#) provides a way of representing a PS style type 4 shading pattern.

7.275.2 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

<i>pFactory</i>	The factory to use.
<i>shadingType</i>	The shading pattern type.
<i>colorSpace</i>	The color space for the shading pattern.
<i>function</i>	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.
<i>meshEntries</i>	The mesh entries to use to construct the shade.
<i>bitsPerCoordinate</i>	The number of bits used to represent each coordinate.
<i>bitsPerComponent</i>	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.
<i>bitsPerFlag</i>	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.
<i>verticesPerRow</i>	The vertices per row parameter. Not used for all shade types.
<i>decode</i>	A vector representing the decode array of the shade. if the data source provides vertex color information.
<i>hasBBox</i>	true if the bBox parameter should be used

Parameters

<i>bBox</i>	The shading bounding box
<i>background</i>	The background color to use, or NULL for no background
<i>renderTransform</i>	The desired render transform.
<i>antiAlias</i>	Whether anti-aliasing should be used for this shade or not
<i>opacity</i>	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

<i>pFactory</i>	The factory to use.
<i>shadingType</i>	The shading pattern type.
<i>colorSpace</i>	The color space for the shading pattern.
<i>dataSource</i>	The data source used for vertex/patch data.
<i>bitsPerCoordinate</i>	The number of bits used to represent each coordinate.
<i>bitsPerComponent</i>	The number of bits used to represent each color component.
<i>bitsPerFlag</i>	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.
<i>verticesPerRow</i>	The vertices per row parameter. Not used for all shade types.
<i>decode</i>	A vector representing the decode array of the shade.
<i>function</i>	The optional shading function describing the color depending on the parametric shade variable. Must be NULL if the data source provides vertex color information.
<i>hasBBox</i>	true if the bBox parameter should be used.
<i>bBox</i>	The shading bounding box.
<i>background</i>	The background color to use, or NULL for no background.
<i>renderTransform</i>	The desired render transform.
<i>antiAlias</i>	Whether anti-aliasing should be used for this shade or not.
<i>opacity</i>	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

<i>bitsPerComponent</i>	Bits per component value to set
-------------------------	---------------------------------

setBitsPerCoordinate()

```
virtual void IDOMShadingPatternType4567Brush::setBitsPerCoordinate (   
    uint8 bitsPerCoordinate ) [pure virtual]
```

Sets the bits per coordinate.

Parameters

<i>bitsPerCoordinate</i>	Bits per coordinate value to set
--------------------------	----------------------------------

setBitsPerFlag()

```
virtual void IDOMShadingPatternType4567Brush::setBitsPerFlag (
    uint8 bitsPerFlag ) [pure virtual]
```

Sets the bits per flag.

Parameters

<i>bitsPerFlag</i>	Bits per flag value to set
--------------------	----------------------------

setDataSource()

```
virtual void IDOMShadingPatternType4567Brush::setDataSource (
    const JawsMako::IPDFStringPtr & dataSource ) [pure virtual]
```

Sets the data source property.

Parameters

<i>dataSource</i>	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

<i>decode</i>	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

<i>meshEntries</i>	The mesh entries.
--------------------	-------------------

setShadingType()

```
virtual void IDOMShadingPatternType4567Brush::setShadingType (
    uint8 shadingType ) [pure virtual]
```

Sets the shading type.

Parameters

<i>shadingType</i>	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

<i>verticesPerRow</i>	Vertices per row value to set. Must be greater than two.
-----------------------	--

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

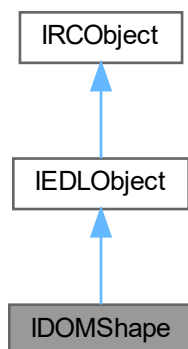
- [idombrush.h](#)

7.276 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.

7.276.1 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 occlusion 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.

7.276.2 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	<i>ptrShape</i>	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	<i>ptrShape</i>	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

<i>factory</i>	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	<i>session</i>	The current session.
in	<i>bUseTempFileForImage</i>	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	<i>ptrShape</i>	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	<i>ptrShape</i>	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	<i>ptrShape</i>	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

<i>offsetX</i>	The number of pixels to offset in x
<i>offsetY</i>	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	<i>ptrShape</i>	The shape with which to unite.
----	-----------------	--------------------------------

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

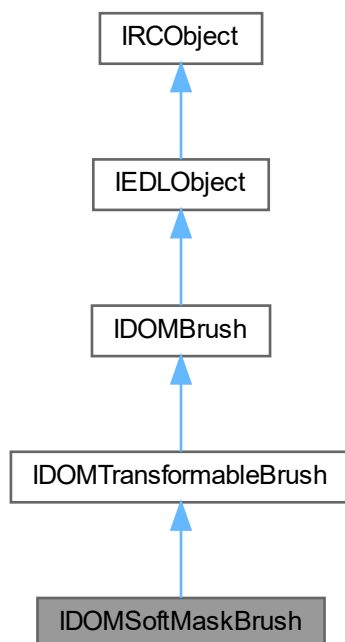
- idomshape.h

7.277 IDOMSoftMaskBrush Class Reference

[IDOMSoftMaskBrush](#) provides a way of representing a PDF style soft mask in it's entirety. 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 [IDOMTransparencyGroupPtr](#) &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.

7.277.1 Detailed Description

[IDOMSoftMaskBrush](#) provides a way of representing a PDF style soft mask in it's entirety. 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.

7.277.2 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

<i>pFactory</i>	The factory to use.
<i>group</i>	The group to form the content of the soft mask.
<i>type</i>	The type of the soft mask.
<i>renderTransform</i>	The desired render transform.
<i>backdropColor</i>	The backdrop color (must have the same color space as the group). Optional.
<i>transfer</i>	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:

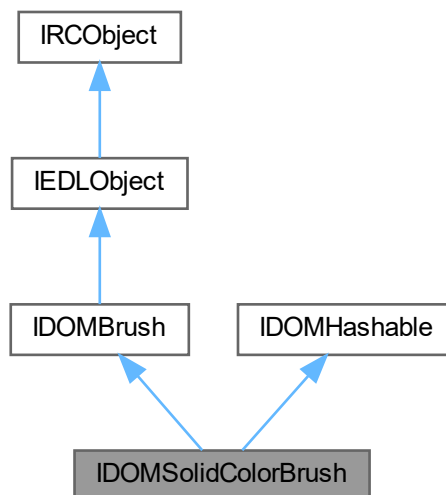
- [idombrush.h](#)

7.278 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.

7.278.1 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.

7.278.2 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

<i>pFactory</i>	The EDL Class factory to use.
<i>color</i>	The color to use.
<i>opacity</i>	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

<i>pFactory</i>	The EDL Class factory to use.
<i>c</i>	The cyan level to use.
<i>m</i>	The magenta level to use.
<i>y</i>	The yellow level to use.
<i>k</i>	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

<i>pFactory</i>	The EDL Class factory to use.
<i>gray</i>	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

<i>pFactory</i>	The EDL Class factory to use.
<i>r</i>	The red level to use.
<i>g</i>	The green level to use.
<i>b</i>	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

<i>pFactory</i>	The EDL Class factory to use.
<i>space</i>	The color space to use.
<i>opacity</i>	The opacity to use (assigned to the brush, not the color)
<i>component1</i>	The first color component to use.
<i>component2</i>	The second color component to use (if applicable).
<i>component3</i>	The third color component to use (if applicable).
<i>component4</i>	The fourth color component to use. (if applicable)
<i>component5</i>	The fifth color component to use (if applicable).

Parameters

<i>component6</i>	The sixth color component to use (if applicable).
<i>component7</i>	The seventh color component to use.
<i>component8</i>	The eighth color component to use (if applicable).
<i>component9</i>	The ninth color component to use (if applicable).
<i>component10</i>	The tenth color component to use (if applicable).
<i>component11</i>	The eleventh color component to use (if applicable).
<i>component12</i>	The twelfth color component to use (if applicable).
<i>component13</i>	The thirteenth color component to use (if applicable).
<i>component14</i>	The fourteenth color component to use (if applicable).
<i>component15</i>	The fifteenth color component to use (if applicable).
<i>component16</i>	The sixteenth color component to use (if applicable).
<i>component17</i>	The seventeenth color component to use (if applicable).
<i>component18</i>	The eighteenth color component to use (if applicable).
<i>component19</i>	The nineteenth color component to use (if applicable).
<i>component20</i>	The twentieth color component to use (if applicable).
<i>component21</i>	The twenty-first color component to use (if applicable).
<i>component22</i>	The twenty-second color component to use (if applicable).
<i>component23</i>	The twenty-third color component to use (if applicable).
<i>component24</i>	The twenty-fourth color component to use (if applicable).
<i>component25</i>	The twenty-fifth color component to use (if applicable).
<i>component26</i>	The twenty-sixth color component to use (if applicable).
<i>component27</i>	The twenty-seventh color component to use (if applicable).
<i>component28</i>	The twenty-eighth color component to use (if applicable).
<i>component29</i>	The twenty-ninth color component to use (if applicable).
<i>component30</i>	The thirtieth color component to use (if applicable).
<i>component31</i>	The thirty-first color component to use (if applicable).
<i>component32</i>	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

<i>pFactory</i>	The EDL Class factory to use.
<i>space</i>	The color space to use.
<i>opacity</i>	The opacity to use (assigned to the brush, not the color)
<i>components</i>	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::createWithSpaceAndComponentsFromVect (
    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

<i>pFactory</i>	The EDL Class factory to use.
<i>space</i>	The color space to use.
<i>opacity</i>	The opacity to use (assigned to the brush, not the color)
<i>components</i>	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

<i>color</i>	The color value
--------------	-----------------

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

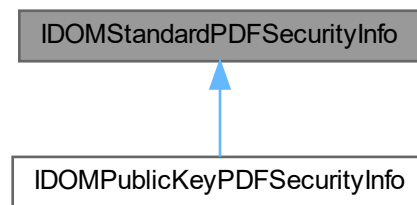
- [idombrush.h](#)

7.279 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.

7.279.1 Detailed Description

Represents security information from PDF Standard encryption handler.

7.279.2 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:

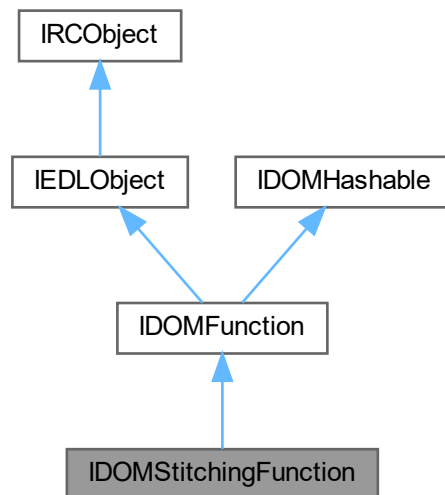
- idomsecurity.h

7.280 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.

7.280.1 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.

7.280.2 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

<i>index</i>	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:

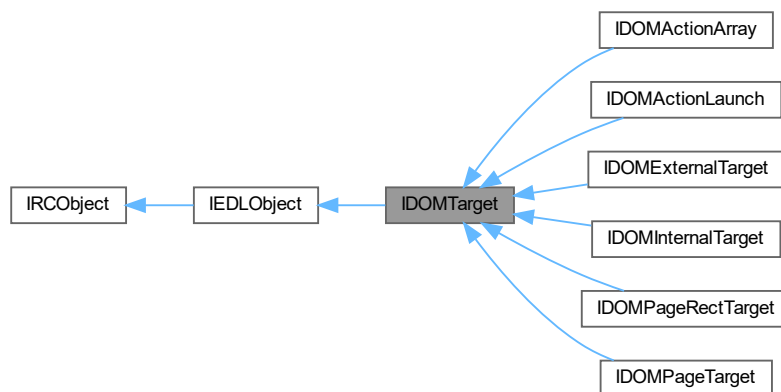
- idomfunction.h

7.281 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.

7.281.1 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.

7.281.2 Member Function Documentation

getTargetType()

```
virtual eTargetType IDOMTarget::getTargetType ( ) const [pure virtual]
```

Retrieves the target type.

Returns

[eTargetType](#) The target type

Implemented in [IDOMExternalTarget](#), [IDOMInternalTarget](#), [IDOMPPageTarget](#), [IDOMPPageRectTarget](#), [IDOMActionLaunch](#), and [IDOMActionArray](#).

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

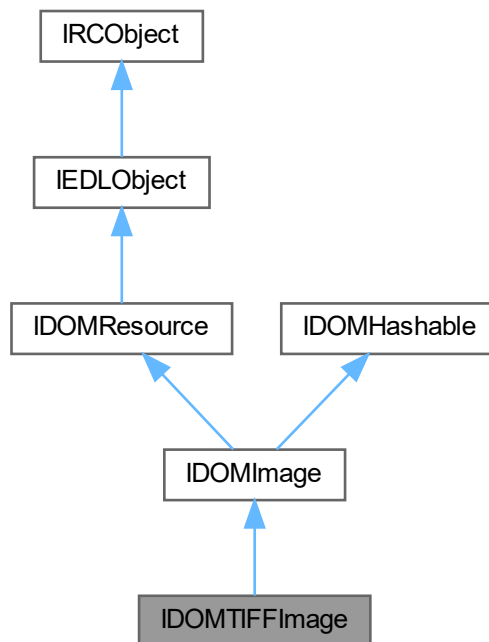
- [idomtarget.h](#)

7.282 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 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 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, IImageFrame↔WriterPtr &frame, const IDOMColorSpacePtr &colorSpace, uint32 width, uint32 height, uint8 bitsPer↔Component=8, double xResolution=96.0, double yResolution=96.0, eTIFFCompression compression↔Type=eTCAuto, eTIFFPrediction predictionType=eTPNone, eImageExtraChannelType extraChannel=e↔IECNone, 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 IOutput↔StreamPtr &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 IOutput↔StreamPtr &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.

7.282.1 Detailed Description

[IDOMTIFFImage](#) interface.

7.282.2 Member Enumeration Documentation

eTIFFCompression

```
enum IDOMTIFFImage::eTIFFCompression
```

Available TIFF compression schemes.

Enumerator

eTCNone	Choose an appropriate scheme for the incoming image.
---------	--

7.282.3 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

<i>pFactory</i>	The EDL Class factory to use.
<i>stream</i>	The stream containing the TIFF image.
<i>imageIndex</i>	For multi-image TIFF files, the index of the image to use, with 0 being the first image in the TIFF file.
<i>allowMultiChannel</i>	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

<i>session</i>	The session to use
<i>frame</i>	On exit, this is populated with a frame ready to receive image data via <code>frame->writeScanLine()</code> . Use <code>frame->flushData()</code> to complete the encoding process.
<i>colorSpace</i>	The color space to use. Must be compatible with the TIFF format.
<i>width</i>	The width of the image, in pixels.
<i>height</i>	The height of the image, in pixels.
<i>bitsPerComponent</i>	The bits per component to use. Must be compatible with the TIFF format.
<i>xResolution</i>	The x resolution, in pixels-per-inch.
<i>yResolution</i>	The y resolution, in pixels-per-inch.
<i>extraChannel</i>	The type of extra channel, if provided. Must be either <code>eIECAlpha</code> or <code>eIECNone</code> for TIFF.
<i>compresssionType</i>	The type of compression to use.
<i>predictionType</i>	The type of prediction to use. Ignored if the predictor is not compatible with the compressor.
<i>bigTiff</i>	Whether or not to use the bigTIFF extensions
<i>inStream</i>	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, <code>outStream</code> must also be provided.
<i>outStream</i>	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, <code>inStream</code> 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

<i>pSession</i>	The relevant EDL session
<i>image</i>	The image to be encoded
<i>scheme</i>	The TIFF compression scheme to be used.
<i>predictionType</i>	The type of prediction to use. Ignored if the predictor is not compatible with the compressor.
<i>bigTiff</i>	If true, use bigtiff format, allowing for very large images. Not all TIFF-consuming software will support this format.
<i>stream</i>	The stream to use to store the image data. For best performance, this should be a random-access stream (IRASStream).

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

<i>pSession</i>	The relevant EDL session
<i>frame</i>	The frame providing the source image data
<i>scheme</i>	The TIFF compression scheme to be used.
<i>predictionType</i>	The type of prediction to use. Ignored if the predictor is not compatible with the compressor.
<i>bigTiff</i>	If true, use bigtiff format, allowing for very large images. Not all TIFF-consuming software will support this format.
<i>stream</i>	The stream to use to store the image data. For best performance, this should be a random-access stream (IRASStream).

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:

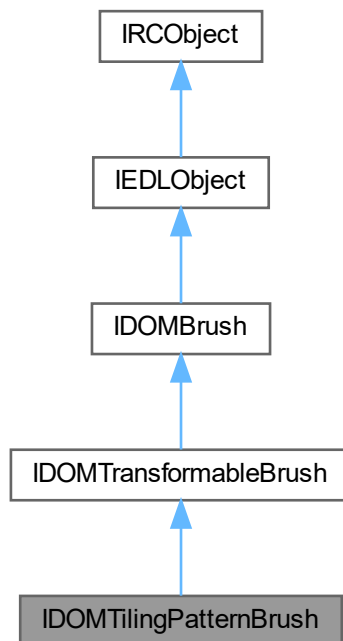
- idomimageresource.h

7.283 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.
- virtual IDOMTilingPatternBrushPtr [getColoredTilingPatternBrush](#) (IEDLClassFactory *pFactory)=0
Get a version of this tiling pattern brush that is always colored (paint type 1) If the brush is already colored, then this brush will be returned. Otherwise a new brush will be created where the pattern color is applied to the contents of the visual.

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 tilingType=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.

7.283.1 Detailed Description

[IDOMTilingPatternBrush](#) provides a way of representing a PS style tiling pattern.

7.283.2 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

<i>pFactory</i>	The factory to use.
<i>visual</i>	The node or node tree representing the pattern's content.
<i>bBox</i>	the bounding box of the pattern's content.
<i>paintType</i>	The paint type, either 1 (colored) or 2 (uncolored)
<i>color</i>	The color to use. Must be NULL for paintType 1. For paintType 2, if NULL, A DeviceGray black will be used.
<i>xStep</i>	The tiling step in X
<i>yStep</i>	The tiling step in Y
<i>tilingType</i>	The tiling type; 1 for constant spacing, 2 for no distortion, 3 for constant spacing and faster tiling.
<i>renderTransform</i>	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

getColorTilingPatternBrush()

```
virtual IDOMTilingPatternBrushPtr IDOMTilingPatternBrush::getColorTilingPatternBrush (
    IEDLClassFactory * pFactory ) [pure virtual]
```

Get a version of this tiling pattern brush that is always colored (paint type 1) If the brush is already colored, then this brush will be returned. Otherwise a new brush will be created where the pattern color is applied to the contents of the visual.

Parameters

<i>pFactory</i>	A pointer to an EDL class factory, which must be an IJawsMako instance.
-----------------	---

Returns

IDOMTilingPatternBrushPtr A colored version of the brush, or the original brush if the brush is already coloured. The returned brush may be cached, so do not edit; make a clone if the returned brush needs to be changed.

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

<i>pFactory</i>	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 may 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

<i>xstep</i>	Reference to <i>xstep</i>
<i>ystep</i>	Reference to <i>ystep</i>

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

<i>bBox</i>	The box to use
-------------	----------------

setPaintType()

```
virtual void IDOMTilingPatternBrush::setPaintType (
    uint8 paintType ) [pure virtual]
```

Sets the paint type for the pattern.

Parameters

<i>paintType</i>	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

<i>color</i>	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

<i>xstep</i>	Reference to xstep
<i>ystep</i>	Reference to ystep

setTilingType()

```
virtual void IDOMTilingPatternBrush::setTilingType (
    uint8 tilingType ) [pure virtual]
```

Sets the tiling type for the pattern.

Parameters

<i>tilingType</i>	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

<i>ptrVisual</i>	Smart pointer to the visual node
------------------	----------------------------------

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

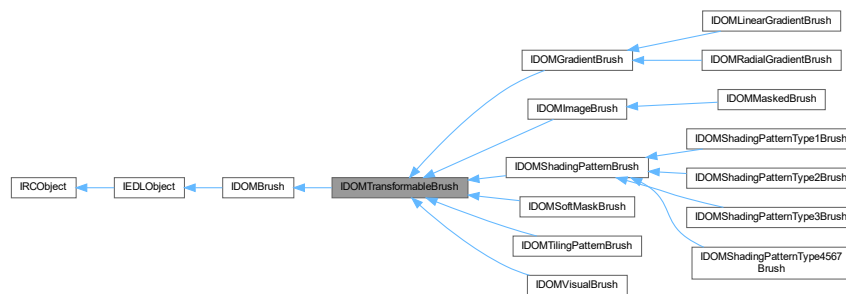
- [idombrush.h](#)

7.284 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.

7.284.1 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.

7.284.2 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

<i>matrix</i>	Render transform matrix
---------------	-------------------------

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

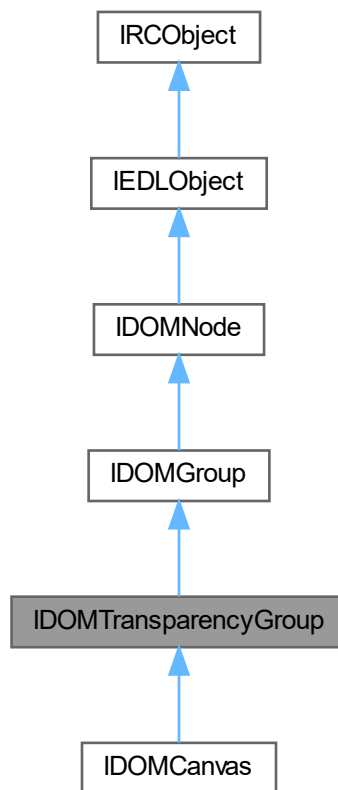
- [idombrush.h](#)

7.285 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.
- virtual [eRenderingIntent](#) [getRenderingIntent](#) () const =0
Get the rendering intent for the node if present. The rendering intent is used whenever the node requires color conversion to a different color space. Set [eRenderingIntentUnset](#) to unset the rendering intent, in which a default will be chosen based on the color space in use. This setting is honored only for rendering, color conversion, PDF or PostScript output.
- virtual void [setRenderingIntent](#) ([eRenderingIntent](#) intent)=0
Set or clear the rendering intent for the node. The rendering intent is used whenever the node requires color conversion to a different color space. Set [eRenderingIntentUnset](#) to unset the rendering intent, in which a default will be chosen based on the color space in use. This setting is honored only for rendering, color conversion, PDF or PostScript output.
- virtual [eBlackPointCompensation](#) [getBlackPointCompensation](#) () const =0
Get the black point compensation for the node. The black point compensation setting is used whenever the node requires color conversion to a different color space and affects how dark regions are handled. This setting is honored only for rendering, color conversion, PDF or PostScript output.
- virtual void [setBlackPointCompensation](#) ([eBlackPointCompensation](#) compensation)=0
Set the black point compensation for the node. The black point compensation setting is used whenever the node requires color conversion to a different color space and affects how dark regions are handled. This setting is honored only for rendering, color conversion, PDF or PostScript output.

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 bCheckComplete=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`
Simplified 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, GDOMNodeVect &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 descendIntoForms=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 `IRCOject`

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

7.285.1 Detailed Description

[IDOMTransparencyGroup](#) interface. Analogous to PDF Transparency groups.

7.285.2 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

<i>pFactory</i>	The EDL class factory
<i>transform</i>	The desired render transform
<i>clip</i>	The desired clip
<i>opacity</i>	The desired opacity
<i>opacityMask</i>	The desired opacity mask
<i>blendMode</i>	The desired blend mode
<i>colorSpace</i>	The desired group color space
<i>isolated</i>	Group isolated parameter
<i>knockout</i>	Group knockout parameter

Returns

IDOMTransparencyGroupPtr A smart pointer to the new transparency group

getBlackPointCompensation()

```
virtual eBlackPointCompensation IDOMTransparencyGroup::getBlackPointCompensation ( ) const
[pure virtual]
```

Get the black point compensation for the node. The black point compensation setting is used whenever the node requires color conversion to a different color space and affects how dark regions are handled. This setting is honored only for rendering, color conversion, PDF or PostScript output.

Returns

eBlackPointCompensation The black point compensation, or eBPCUnset if not present.

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 if no color 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 if no opacity mask is present.

getRenderingIntent()

```
virtual eRenderingIntent IDOMTransparencyGroup::getRenderingIntent ( ) const [pure virtual]
```

Get the rendering intent for the node if present. The rendering intent is used whenever the node requires color conversion to a different color space. Set eRenderingIntentUnset to unset the rendering intent, in which a default will be chosen based on the color space in use. This setting is honored only for rendering, color conversion, PDF or PostScript output.

Returns

eRenderingIntent The rendering intent, or eRenderingIntentUnset if not present.

setBlackPointCompensation()

```
virtual void IDOMTransparencyGroup::setBlackPointCompensation (
    eBlackPointCompensation compensation ) [pure virtual]
```

Set the black point compensation for the node. The black point compensation setting is used whenever the node requires color conversion to a different color space and affects how dark regions are handled. This setting is honored only for rendering, color conversion, PDF or PostScript output.

Parameters

<i>compensation</i>	The desired black point compensation setting, or eBPCUnset to clear.
---------------------	--

setBlendMode()

```
virtual void IDOMTransparencyGroup::setBlendMode (
    eBlendMode blendMode ) [pure virtual]
```

Set the blend mode to be used for compositing this group with the backdrop.

Parameters

<i>blendMode</i>	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

<i>colorSpace</i>	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

<i>isolated</i>	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

<i>knockout</i>	New value of knockout
-----------------	-----------------------

setOpacity()

```
virtual void IDOMTransparencyGroup::setOpacity (
    float opacity ) [pure virtual]
```

Set the group opacity.

Parameters

<i>opacity</i>	The desired opacity.
----------------	----------------------

setOpacityMask()

```
virtual void IDOMTransparencyGroup::setOpacityMask (
    const IDOMBrushPtr & ptrOpacityMask ) [pure virtual]
```

Sets opacity mask.

Parameters

<i>ptrOpacityMask</i>	Smart pointer to brush
-----------------------	------------------------

setRenderingIntent()

```
virtual void IDOMTransparencyGroup::setRenderingIntent (
    eRenderingIntent intent ) [pure virtual]
```

Set or clear the rendering intent for the node. The rendering intent is used whenever the node requires color conversion to a different color space. Set eRenderingIntentUnset to unset the rendering intent, in which a default will be chosen based on the color space in use. This setting is honored only for rendering, color conversion, PDF or PostScript output.

Parameters

<i>intent</i>	The desired rendering intent, or eRenderingIntentUnset to clear.
---------------	--

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

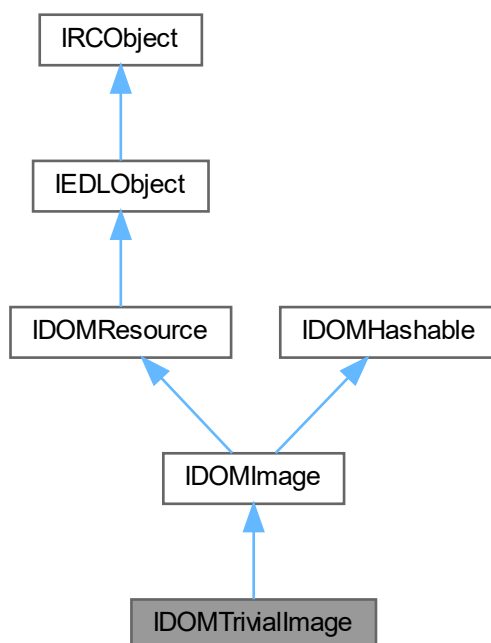
- idomgroup.h

7.286 IDOMTrivialImage Class Reference

Interface for a "trivial" image.

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

Inheritance diagram for IDOMTrivialImage:



Classes

- class [Data](#)
Initialization data.

Public Member Functions

- virtual `InputStreamPtr` [getStream](#) () const
This image type does not allow direct access to the underlying streams.
- virtual void [setStream](#) (const `InputStreamPtr` &stream)
This image type does not allow direct access to the underlying streams.

Public Member Functions inherited from IDOMImage

- virtual `ImageDecoderPtr` [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 `ImageFramePtr` [getImageFrame](#) (`IEDLClassFactory` *factory)
Fetch the image frame; convenience.
- virtual `ImageEncoderPtr` [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 `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 IDOMTrivialImagePtr [create](#) ([IEDLClassFactory](#) *pFactory, const IDOMColorSpacePtr &colorSpace, uint16 width, uint16 height, uint32 stride, uint8 bitsPerComponent, const CEDLSimpleBuffer &imageData, [eImageExtraChannelType](#) extraChannel=eIECNone, [eDOMImageType](#) type=eDITUnknown)
Create a "trivial" image resource with the given memory and pixel data.
- static const [CClassID](#) & [classID](#) ()
Retrieves class id of [IDOMRawImage](#).

Additional Inherited Members

Protected Member Functions inherited from [IRCObject](#)

- virtual [~IRCObject](#) ()
Virtual destructor.

7.286.1 Detailed Description

Interface for a "trivial" image.

A trivial image is used for cases where the image data is extremely small, to the point that it is wasteful to store in a separate stream.

The resolution is fixed to 96 dpi. Otherwise, supported colour spaces, channel formats, and available depths are as per [IDOMRawImage](#).

7.286.2 Member Function Documentation

[classID\(\)](#)

```
static const CClassID & IDOMTrivialImage::classID ( ) [inline], [static]
```

Retrieves class id of [IDOMRawImage](#).

Returns

[CClassID](#) Class id of the element

[create\(\)](#)

```
static EDL_API IDOMTrivialImagePtr IDOMTrivialImage::create (
    IEDLClassFactory * pFactory,
    const IDOMColorSpacePtr & colorSpace,
    uint16 width,
    uint16 height,
    uint32 stride,
    uint8 bitsPerComponent,
    const CEDLSimpleBuffer & imageData,
    eImageExtraChannelType extraChannel = eIECNone,
    eDOMImageType type = eDITUnknown ) [static]
```

Create a "trivial" image resource with the given memory and pixel data.

Parameters

<i>pFactory</i>	The EDL Class factory to use.
<i>colorSpace</i>	The color space to use. Any valid color space may be used for IDOMTrivialImage .
<i>width</i>	The width of the image, in pixels.
<i>height</i>	The height of the image, in pixels.
<i>stride</i>	The distance in memory, in bytes, from one scanline to the next.
<i>bitsPerComponent</i>	The bits per component to use. 1, 2, 4, 8, 12 and 16 bits per sample are supported.
<i>imageData</i>	A buffer containing the image data. Must be height * stride bytes and presented in Mako/PDF standard image byte order (big endian if 12 or 16 bit).
<i>extraChannel</i>	The type of extra channel, if provided.
<i>type</i>	The image type.

Returns

IDOMRawImagePtr The new image.

getStream()

```
virtual IInputStreamPtr IDOMTrivialImage::getStream ( ) const [inline], [virtual]
```

This image type does not allow direct access to the underlying streams.

Returns

IInputStreamPtr Always nullptr

Implements [IDOMResource](#).

setStream()

```
virtual void IDOMTrivialImage::setStream (
    const IInputStreamPtr & stream ) [inline], [virtual]
```

This image type does not allow direct access to the underlying streams.

Parameters

<i>stream</i>	A smart pointer to the stream
---------------	-------------------------------

Implements [IDOMResource](#).

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

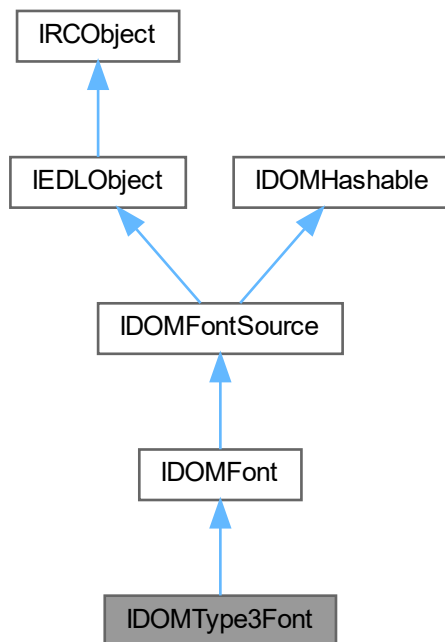
- idomimageresource.h

7.287 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.

7.287.1 Detailed Description

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

7.287.2 Member Function Documentation

addGlyph()

```
virtual void IDOMType3Font::addGlyph (
    const IDOMGlyphPtr & glyph ) [pure virtual]
```

Adds a glyph of the given name to the font.

Parameters

<i>glyph</i>	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

<i>glyphName</i>	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

<i>glyphID</i>	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

<i>codePoint</i>	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

<i>glyphName</i>	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

<i>glyphID</i>	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

<i>codePoint</i>	The unicode codepoint
------------------	-----------------------

Returns

bool true if the glyph is present, false otherwise

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

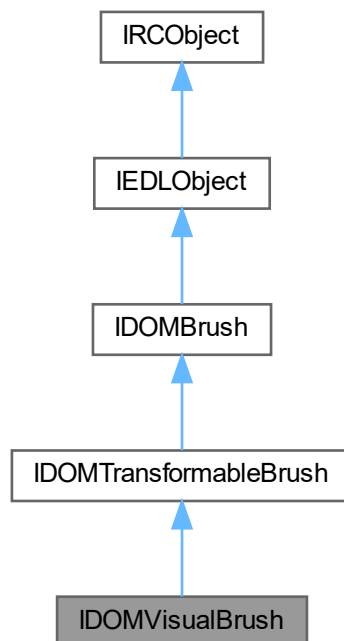
- [idomfont.h](#)

7.288 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.

7.288.1 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](#)

7.288.2 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

<i>pFactory</i>	The factory to use.
<i>visual</i>	The node to use as the visual.
<i>viewBox</i>	The desired view box.
<i>viewPort</i>	The desired view port.
<i>renderTransform</i>	The desired render transform.
<i>opacity</i>	The desired brush opacity.
<i>tileMode</i>	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

<i>pFactory</i>	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

<i>pFactory</i>	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

<i>tm</i>	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

<i>vb</i>	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**

<i>vu</i>	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

<i>vp</i>	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

<i>vu</i>	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

<i>ptrVisual</i>	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:

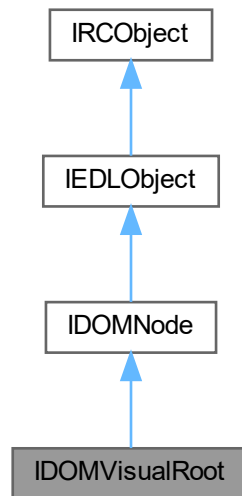
- [idombrush.h](#)

7.289 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 bCheckComplete=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

Simplified 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 descendIntoForms=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.

7.289.1 Detailed Description

[IDOMVisualRoot](#) interface.

7.289.2 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:

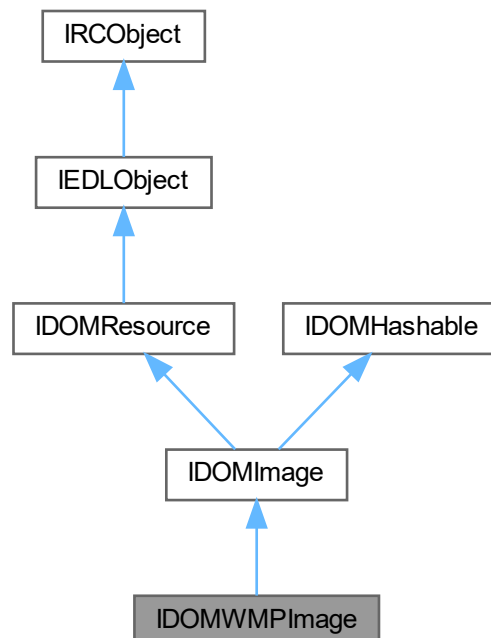
- [idombrush.h](#)

7.290 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 [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.

Protected Member Functions inherited from [IRCObject](#)

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

7.290.1 Detailed Description

[IDOMWMPImage](#) interface.

7.290.2 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:

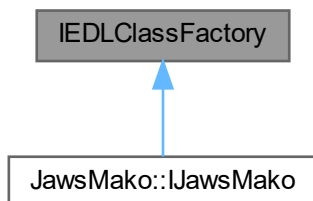
- idomimageresource.h

7.291 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)

7.291.1 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

7.291.2 Member Function Documentation

createInstance()

```
virtual IEDLObjectPtr IEDLClassFactory::createInstance (
    const CClassID & id,
    CClassParams * pParams = NULL ) [pure virtual]
```

Creates the registered class by [CClassID](#).

Parameters

<i>id</i>	Class GUID
<i>pParams</i>	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

<i>id</i>	Class GUID
<i>pParams</i>	Parameters for class initialization
<i>factory</i>	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

<i>strName</i>	Name to find
<i>id</i>	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

<i>id</i>	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

<i>id</i>	Class GUID
<i>f</i>	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

<i>strName</i>	Name for registartion
<i>id</i>	Class GUID
<i>overwrite</i>	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

7.292 IEDLError Class Reference

An abstract class for EDL exceptions.

```
#include <edlerrors.h>
```

Inherits std::exception.

7.292.1 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:

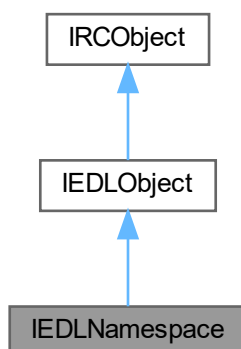
- edlerrors.h

7.293 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.

7.293.1 Detailed Description

Interface to EDL Namespace class.

7.293.2 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

<i>sNamespace</i>	Namespace
-------------------	-----------

Returns

bool True on success

getPrefix()

```
virtual bool IEDLNamespace::getPrefix (
    EDLSysString & sPrefix ) const [pure virtual]
```

Retrieves the namespace prefix.

Parameters

<i>sPrefix</i>	Namespace prefix
----------------	------------------

Returns

bool True on success

setNamespace()

```
virtual bool IEDLNamespace::setNamespace (
    const EDLSysString & sNamespace ) [pure virtual]
```

Sets the namespace.

Parameters

<i>sNamespace</i>	Namespace
-------------------	-----------

Returns

bool Returns true on success

setPrefix()

```
virtual bool IEDLNamespace::setPrefix (
    const EDLSysString & sPrefix ) [pure virtual]
```

Sets the name space prefix.

Parameters

<i>sPrefix</i>	Namespace prefix
----------------	------------------

Returns

bool True on success

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

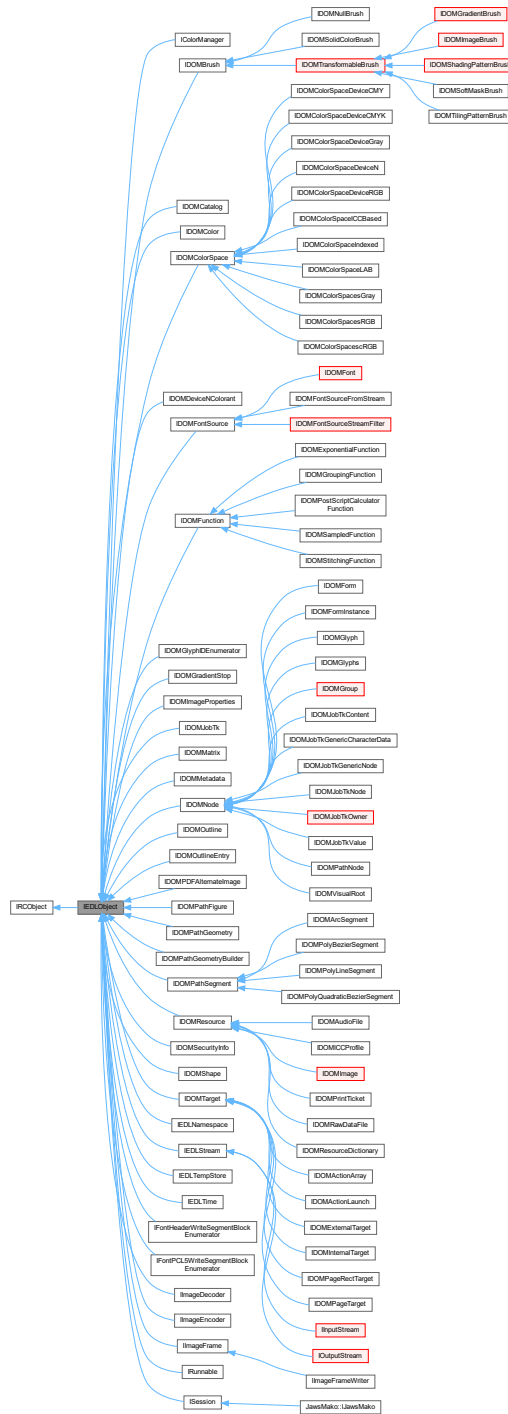
- [edlqname.h](#)

7.294 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 IEDLObject

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

- virtual `~IEDLObject` ()
Virtual destructor.

7.294.1 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

7.294.2 Member Function Documentation

`clone()`

```
virtual bool IEDLObject::clone (
    IEDLObjectPtr & ptrObject,
    IEDLClassFactory * pFactory ) [inline], [virtual]
```

Create a copy of EDLObject.

Parameters

<i>ptrObject</i>	Smart pointer to the source object
<i>pFactory</i>	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

<i>pData</i>	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:

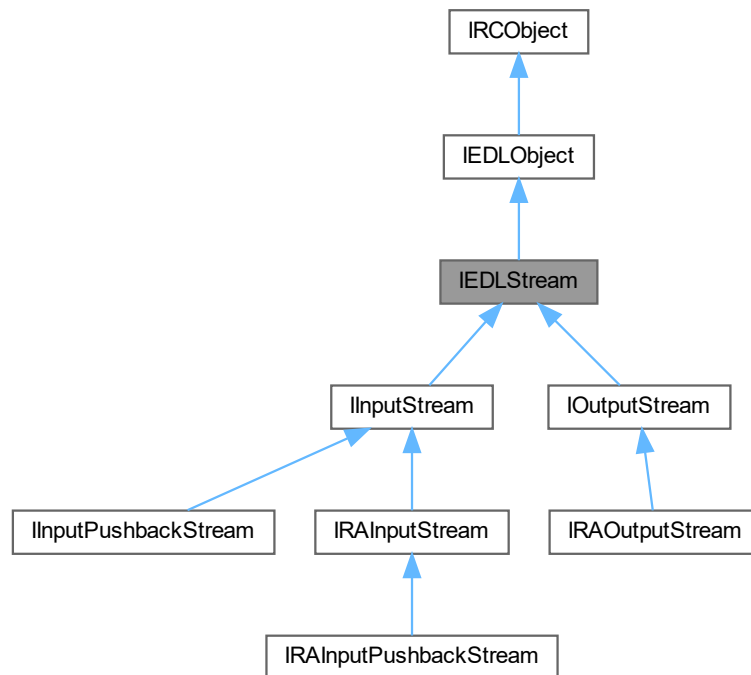
- [iedlobject.h](#)

7.295 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.

7.295.1 Detailed Description

Generic stream. Abstract base class for EDL stream subsystem.

7.295.2 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:

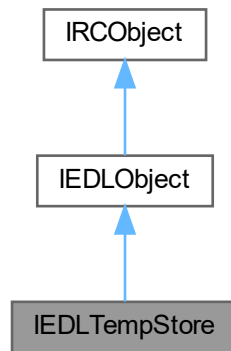
- edlstream.h

7.296 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.

7.296.1 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.

7.296.2 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

<i>stream</i>	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

<i>reader</i>	reference to receive a reader for the temporary object.
<i>writer</i>	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

<i>stream</i>	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:

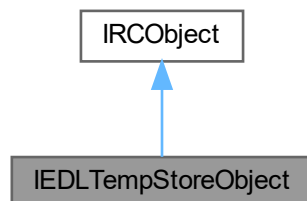
- iedltempstore.h

7.297 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 simultaneously 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.

7.297.1 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.

7.297.2 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

<i>buffered</i>	- 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:

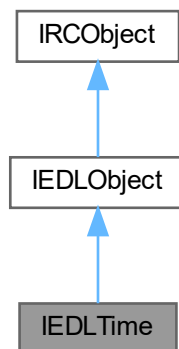
- [iedltempstore.h](#)

7.298 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.

7.298.1 Detailed Description

Interface to EDL date-time class.

7.298.2 Member Function Documentation

compare()

```
virtual int32 IEDLTime::compare (
    const IEDLTimePtr & ptrTime ) const [pure virtual]
```

Compare this time against ptrTime.

Parameters

<i>ptrTime</i>	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

<i>strTime</i>	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

<i>strTime</i>	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

<i>hour</i>	The hour component
<i>min</i>	The minute component
<i>sec</i>	The second component
<i>ms</i>	The fractional second component
<i>tzd_sign</i>	The timezone sign component
<i>tzd_hour</i>	The timezone hour component
<i>tzd_min</i>	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

<i>ptrTime</i>	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

<i>day</i>	The day value
------------	---------------

setMonth()

```
virtual bool IEDLTime::setMonth (
    uint16 month ) [pure virtual]
```

Sets the month.

Parameters

<i>month</i>	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

<i>hour</i>	The hour component
<i>min</i>	The minute component
<i>sec</i>	The second component
<i>ms</i>	The fractional second component
<i>tzd_sign</i>	The timezone sign component
<i>tzd_hour</i>	The timezone hour component
<i>tzd_min</i>	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

<i>year</i>	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:

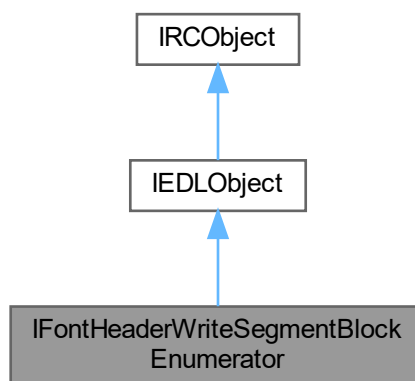
- [edltime.h](#)

7.299 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.

7.299.1 Detailed Description

[IFontHeaderWriteSegmentBlockEnumerator](#) Enumerates over the PCLXL Font Header block items for the XL ReadFontHeader operator.

7.299.2 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

<i>outStream</i>	The output stream
<i>endian</i>	The endian of the data to write out

Returns

int32 The size of data written to the output stream

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

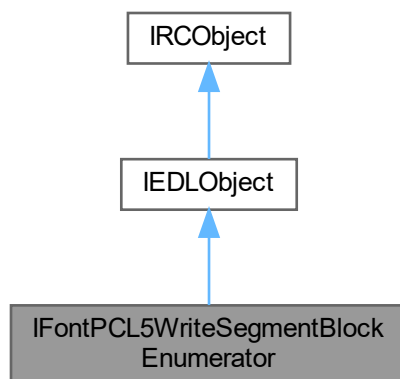
- [idomfontpcl.h](#)

7.300 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.

7.300.1 Detailed Description

[IFontPCL5WriteSegmentBlockEnumerator](#) Enumerates over the PCL5 font blocks.

7.300.2 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,
    eDLEndian endian ) [pure virtual]
```

Writes the current enumeration item block.

Parameters

<i>outStream</i>	The output stream
<i>endian</i>	The endian of the data to write out

Returns

int32 The size of data written to the output stream

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

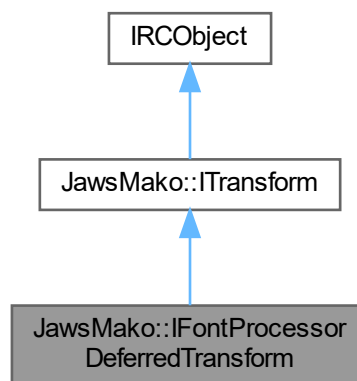
- [idomfontpcl.h](#)

7.301 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 void **setIgnoreUnembeddedFonts** (bool ignore)=0

- Set whether or not fonts that are not marked for embedding should be ignored by this transform. The default is false.*

 - virtual void **setIgnoreRestrictedFonts** (bool ignore)=0
- Set whether or not fonts that are marked as restricted should be ignored by this transform. The default is false.*

 - virtual void **setIgnoreFonts** (const CU8StringVect &ignoreFonts)=0

Specify a list of font names that should not be processed by this transform. A font will be ignored if its PostScript name matches any string in this vector. The default is an empty vector.
- 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=IAAbortPtr())

Create the transform.

Additional Inherited Members

Protected Member Functions inherited from [IRCObject](#)

- virtual [~IRCObject](#) ()

Virtual destructor.

7.301.1 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.

7.301.2 Member Function Documentation

create()

```
static JAWSMAKO_API IFontProcessorDeferredTransformPtr JawsMako::IFontProcessorDeferred←
Transform::create (
    const IJawsMakoPtr & jawsMako,
    const IAbortPtr & abort = IAbortPtr() ) [static]
```

Create the transform.

Parameters

<i>jawsMako</i>	The JawsMako instance.
<i>abort</i>	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:

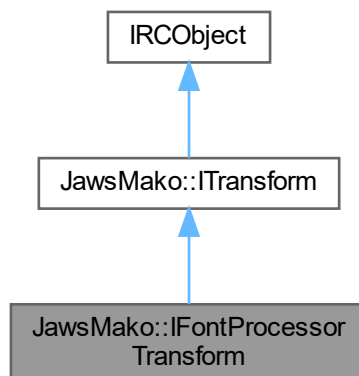
- [transforms.h](#)

7.302 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 void **setIgnoreUnembeddedFonts** (bool ignore)=0
Set whether or not fonts that are not marked for embedding should be ignored by this transform. The default is false.
- virtual void **setIgnoreRestrictedFonts** (bool ignore)=0
Set whether or not fonts that are marked as restricted should be ignored by this transform. The default is false.
- virtual void **setIgnoreFonts** (const CU8StringVect &ignoreFonts)=0
Specify a list of font names that should not be processed by this transform. A font will be ignored if its PostScript name matches any string in this vector. The default is an empty vector.
- 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 **IRCOject**

- virtual void **addRef** () const =0
Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool **decRef** () const =0
Decreases the reference count of the actual object pointed to. When the reference count falls to 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 [IRCOObject](#)

- virtual [~IRCOObject](#) ()

Virtual destructor.

7.302.1 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.

7.302.2 Member Function Documentation

create()

```
static JAWSMAKO_API IFontProcessorTransformPtr JawsMako::IFontProcessorTransform::create (
    const IJawsMakoPtr & jawsMako,
    const IAbortPtr & abort = IAbortPtr() ) [static]
```

Create the transform.

Parameters

<i>jawsMako</i>	The JawsMako instance.
<i>abort</i>	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:

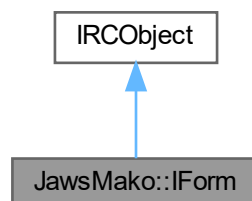
- [transforms.h](#)

7.303 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.

7.303.1 Detailed Description

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

7.303.2 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

<i>field</i>	Field to 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

<i>widgetReference</i>	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

<i>widget</i>	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

<i>jawsMako</i>	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

<i>field</i>	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

<i>field</i>	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

<i>field</i>	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

<i>field</i>	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

<i>widgetReference</i>	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

<i>widget</i>	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

<i>widget</i>	The widget to inspect
---------------	-----------------------

Returns

U8String The default appearance 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

<i>widget</i>	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

<i>widget</i>	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

<i>widget</i>	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

<i>widget</i>	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

<i>widget</i>	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

<i>widget</i>	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

<i>field</i>	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

<i>widgetReference</i>	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

<i>widget</i>	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

<i>needAppearances</i>	True or false
------------------------	---------------

setXfaPacketData()

```
virtual void JawsMako::IForm::setXfaPacketData (
    const CXFAPacketVect & xfaPacketData ) [pure virtual]
```

Set the XFA packet data for this form.

Parameters

<i>xfaPacketData</i>	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

<i>widgetReference</i>	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

<i>widget</i>	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:

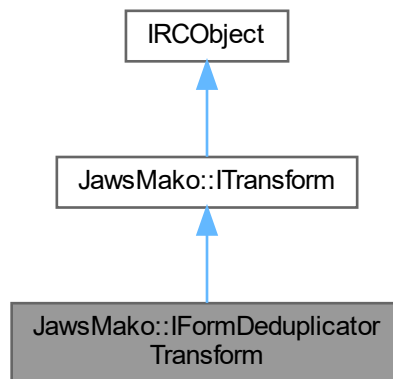
- [interactive.h](#)

7.304 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.

7.304.1 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:

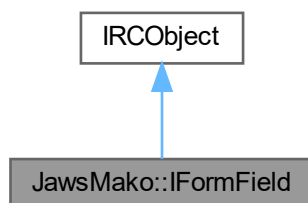
- [transforms.h](#)

7.305 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.

7.305.1 Detailed Description

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

7.305.2 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

<i>field</i>	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

<i>widgetReference</i>	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

<i>widget</i>	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

<i>jawsMako</i>	The Mako instance
<i>type</i>	The type of field to create
<i>partialName</i>	The desired partial name of the field
<i>flags</i>	The field flags to set (see eFieldFlags)
<i>childFields</i>	A vector of child fields for this field
<i>childWidgets</i>	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

<i>field</i>	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

<i>defaultAppearanceString</i>	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

<i>value</i>	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

<i>flags</i>	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

<i>options</i>	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

<i>value</i>	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

<i>field</i>	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

<i>widgetReference</i>	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

<i>widget</i>	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

<i>defaultAppearanceString</i>	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

<i>value</i>	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

<i>value</i>	The string value for the field.
--------------	---------------------------------

setFieldFlags()

```
virtual void JawsMako::IFormField::setFieldFlags (
    uint32 flags ) [pure virtual]
```

Set the field flags - #see eFieldFlags.

Parameters

<i>flags</i>	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

<i>name</i>	The partial name
-------------	------------------

setQuadding()

```
virtual void JawsMako::IFormField::setQuadding (
    eQuadding quadding ) [pure virtual]
```

Set the Quadding (Justification) for variable text.

Parameters

<i>quadding</i>	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

<i>value</i>	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

<i>value</i>	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

<i>widgetReference</i>	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

<i>widget</i>	The widget to search for
---------------	--------------------------

Returns

bool True if found, otherwise false

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

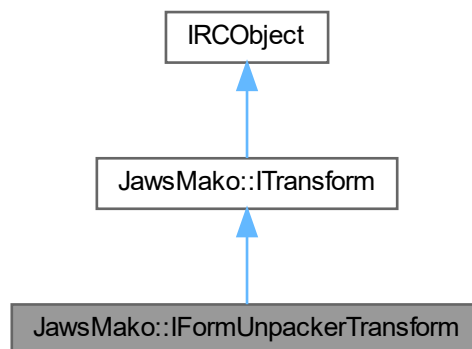
- [interactive.h](#)

7.306 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=eBGeneral, 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.

7.306.1 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.

7.306.2 Member Function Documentation

`create()`

```
static JAWSMako_API IFormUnpackerTransformPtr JawsMako::IFormUnpackerTransform::create (
    const IJawsMakoPtr & jawsMako,
    const IAbortPtr & abort = IAbortPtr() ) [static]
```

Create the transform.

Parameters

<code>jawsMako</code>	The JawsMako instance.
<code>abort</code>	An abort callback handler.

Returns

The new instance.

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

- [transforms.h](#)

7.307 IFrame Class Reference

A frame, into which text can be flowed by the layout engine.

```
#include <layout.h>
```

7.307.1 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:

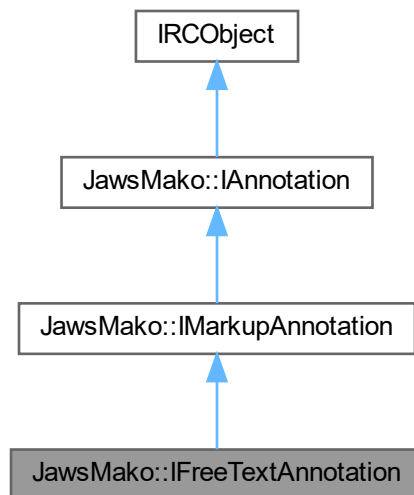
- layout.h

7.308 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.

7.308.1 Detailed Description

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

7.308.2 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:

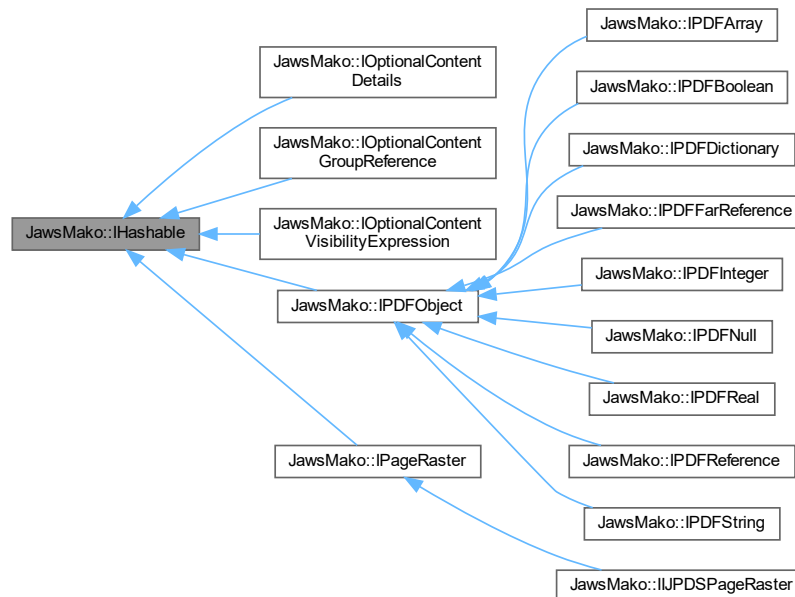
- [interactive.h](#)

7.309 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.

7.309.1 Detailed Description

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

7.309.2 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

<i>hash</i>	The hash to update.
-------------	---------------------

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

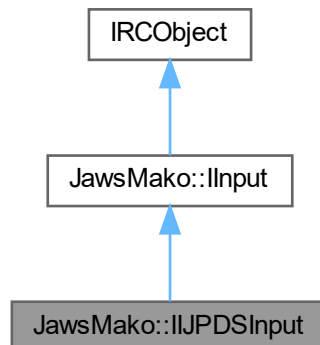
- [types.h](#)

7.310 JawsMako::IJPDSInput Class Reference

An instance of the JawsMako IJPDS input class.

```
#include <ijpdsinput.h>
```

Inheritance diagram for JawsMako::IJPDSInput:

**Static Public Member Functions**

- static JAWSMako_API IJPDSInputPtr **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](#) ¶m, 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.

7.310.1 Detailed Description

An instance of the JawsMako IJPDS input class.

7.310.2 Member Function Documentation

create()

```
static JAWSMako_API IJPDSInputPtr JawsMako::IJPDSInput::create (
    const IJawsMakoPtr & jawsMako,
    const IProgressMonitorPtr & progressMonitor = IProgressMonitorPtr() ) [static]
```

Create an input for reading source documents in IJPDS format.

Parameters

<i>jawsMako</i>	The IJawsMako object
<i>progressMonitor</i>	A smart pointer to an IProgressMonitor object which can be NULL if no such object was passed in.

Returns

IJPDSInputPtr the IJPDS input

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

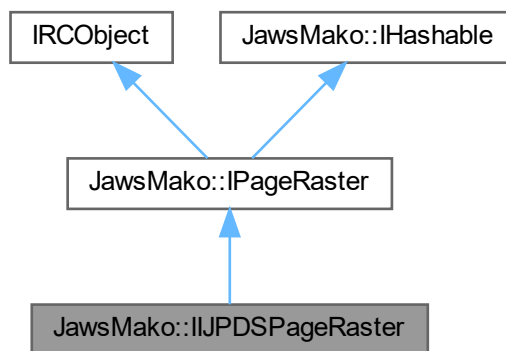
- [ijpdsinput.h](#)

7.311 JawsMako::IJPDSPageRaster 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::IJPDSPageRaster:

**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 **getRowBytesPerRow** () 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.

7.311.1 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:

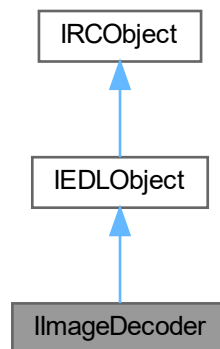
- [ijpdsinput.h](#)

7.312 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.

7.312.1 Detailed Description

[ImageDecoder](#) returns [ImageFrame](#) objects as requested by the client. This object knows about the imageformat internals and knows how to unpack the image.

7.312.2 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:

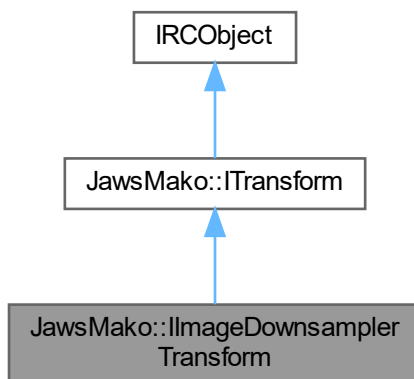
- [iimagecodec.h](#)

7.313 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 threshold 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 threshold 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 threshold 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 transformAnnotations=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.

7.313.1 Detailed Description

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

7.313.2 Member Function Documentation

`create()`

```
static JAWSMako_API IImageDownsamplerTransformPtr JawsMako::IImageDownsamplerTransform::create
(
    const IJawsMakoPtr & jawsMako,
    const IAbortPtr & abort = IAbortPtr() ) [static]
```

Create the transform.

Parameters

<code>jawsMako</code>	The JawsMako instance.
-----------------------	------------------------

Returns

The new instance.

setDownsampleMaskedImages()

```
virtual void JawsMako::ImageDownsamplerTransform::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::ImageDownsamplerTransform::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:

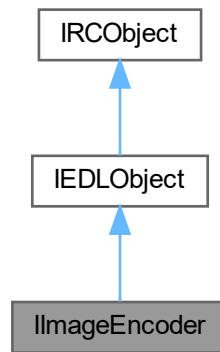
- [transforms.h](#)

7.314 ImageEncoder Class Reference

`ImageEncoder` accepts `ImageFrame` objects and streams it out to an image format.

```
#include <iimagecodec.h>
```

Inheritance diagram for `ImageEncoder`:



Public Member Functions

- `ImageEncoder ()`
Constructor.
- `virtual ~ImageEncoder ()`
Destructor.
- `virtual ImageFrameWriterPtr 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.

7.314.1 Detailed Description

[ImageEncoder](#) accepts [ImageFrame](#) objects and streams it out to an image format.

7.314.2 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:

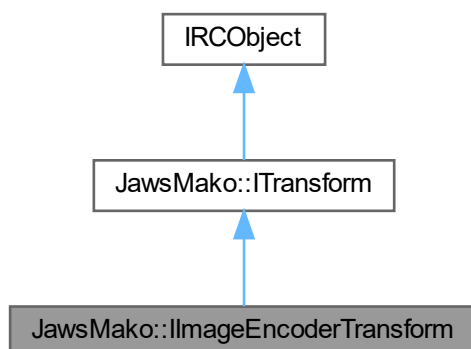
- [iimagecodec.h](#)

7.315 JawsMako::ImageEncoderTransform 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::ImageEncoderTransform`:



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 **setEnabledTIFFEncoding** (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 [setEnabledTIFFEncoding\(\)](#) 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 monochrome images. Any compression 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 ImageEncoderTransformPtr [create](#) (const IJawsMakoPtr &jawsMako, const IAbortPtr &abort=IAbortPtr())
Create the transform.

Additional Inherited Members

Protected Member Functions inherited from IRCObject

- virtual [~IRCObject](#) ()
Virtual destructor.

7.315.1 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.

7.315.2 Member Enumeration Documentation

eEncodeFormat

```
enum JawsMako::IImageEncoderTransform::eEncodeFormat
```

Available target image formats.

Enumerator

eEFJPEG	Attempt to choose an appropriate format for each image.
---------	---

7.315.3 Member Function Documentation

create()

```
static JAWSMAKO_API IImageEncoderTransformPtr JawsMako::IImageEncoderTransform::create (
    const IJawsMakoPtr & jawsMako,
    const IAbortPtr & abort = IAbortPtr() ) [static]
```

Create the transform.

Parameters

<i>jawsMako</i>	The JawsMako instance.
<i>abort</i>	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

<i>scheme</i>	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

<i>scheme</i>	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

<i>quality</i>	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 monochrome images. Any compression scheme may be used.

Parameters

<i>scheme</i>	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

<i>format</i>	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

<i>format</i>	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

<i>format</i>	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

<i>format</i>	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

<i>scheme</i>	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:

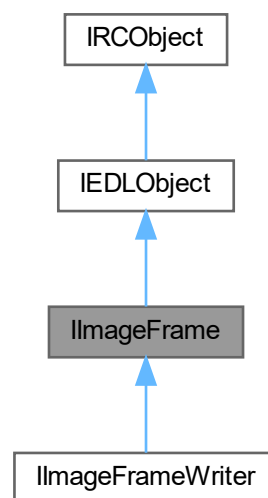
- [transforms.h](#)

7.316 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 `enumImageExtraChannelType` `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` `getRowBytesPerRow ()` 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 `getRowBytesPerRow()`)
 - 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.

7.316.1 Detailed Description

[IImageFrame](#) encapsulates an EDL image with its details.

7.316.2 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

<i>isNoisy</i>	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

<i>pRow</i>	The buffer in which the row scanline will be copied into.
<i>bufferSize</i>	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

<i>skipCount</i>	The number of scanlines to skip..
------------------	-----------------------------------

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

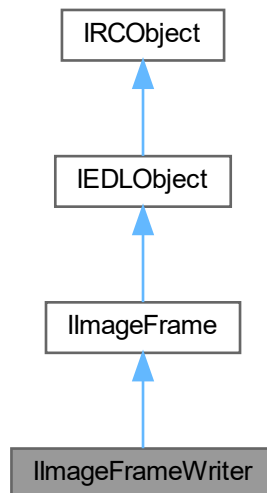
- [iimagecodec.h](#)

7.317 IImageFrameWriter Class Reference

[IImageFrameWriter](#) writes an image from an imageframe.

```
#include <iimagecodec.h>
```

Inheritance diagram for IImageFrameWriter:



Public Member Functions

- **ImageFrameWriter** ()
Constructor.
- virtual **~ImageFrameWriter** ()
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 ImageFrame

- 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

- virtual double `getYResolution ()` const =0
Gets the resolution of the image in the Y direction.
- virtual uint32 `getRowBytesPerRow ()` 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.

7.317.1 Detailed Description

`IImageFrameWriter` writes an image from an imageframe.

7.317.2 Member Function Documentation

`setBPS()`

```
virtual void IImageFrameWriter::setBPS (
    uint8 bps ) [pure virtual]
```

Set the bits per sample/components of the frame.

Parameters

<i>bps</i>	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

<i>colorSpace</i>	The desired colorspace
-------------------	------------------------

setHeight()

```
virtual void IImageFrameWriter::setHeight (
    uint32 height ) [pure virtual]
```

Set the height of the frame in pixels.

Parameters

<i>height</i>	The height
---------------	------------

setImageExtraChannelType()

```
virtual void IImageFrameWriter::setImageExtraChannelType (
    eImageExtraChannelType extraChannelType ) [pure virtual]
```

Set the type of the extra image channel, if present.

Parameters

<i>extraChannelType</i>	The extra channel type
-------------------------	------------------------

setWidth()

```
virtual void IImageFrameWriter::setWidth (
    uint32 width ) [pure virtual]
```

Set the width of the frame in pixels.

Parameters

<i>width</i>	The width
--------------	-----------

setXResolution()

```
virtual void IImageFrameWriter::setXResolution (
    double xRes ) [pure virtual]
```

Set the resolution of the frame in the X direction.

Parameters

<i>xRes</i>	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

<i>yRes</i>	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

<i>pScanLine</i>	A buffer that contains the scanline.
------------------	--------------------------------------

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

- [iimagecodec.h](#)

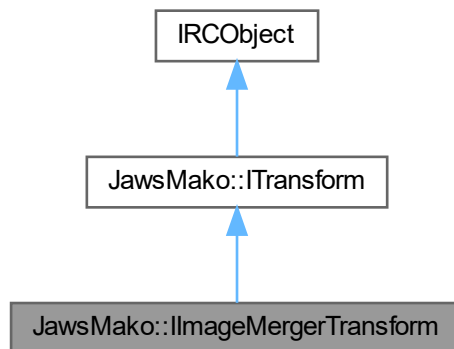
7.318 JawsMako::ImageMergerTransform Class Reference

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::ImageMergerTransform:



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 transformAnnotations=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.

7.318.1 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.

7.318.2 Member Function Documentation

create()

```
static JAWSMAKO_API IImageMergerTransformPtr JawsMako::IImageMergerTransform::create (
    const IJawsMakoPtr & jawsMako,
    const IAbortPtr & abort = IAbortPtr() ) [static]
```

Create the transform.

Parameters

<i>jawsMako</i>	The JawsMako instance.
<i>abort</i>	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](#)

7.319 JawsMako::IImageMergerTransform::Implementation 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 [IDOMFixedPagePtr](#) &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](#) (Implementation *genericImplementation, const IDOMChar↔PathGroupPtr &group, bool &changed, bool transformChildren, const [CTransformState](#) &state)
Callback to process an IDOMCharPathGroup.
- virtual IDOMNodePtr [transformTransparencyGroup](#) (Implementation *genericImplementation, const IDOMTransparencyGroupPtr &group, bool &changed, bool transformChildren, const [CTransformState](#) &state)
Callback to process an IDOMTransparencyGroup.
- virtual IDOMNodePtr [transformCanvas](#) (Implementation *genericImplementation, const IDOMCanvasPtr &canvas, bool &changed, bool transformChildren, const [CTransformState](#) &state)
Callback to process an IDOMCanvas.
- virtual IDOMNodePtr [transformGlyphs](#) (Implementation *genericImplementation, const IDOMGlyphsPtr &glyphs, bool &changed, const [CTransformState](#) &state)
Callback to process an IDOMGlyphs node.
- virtual IDOMFontPtr [transformFont](#) (Implementation *genericImplementation, const IDOMFontPtr &font, uint32 &index, const [CTransformState](#) &state)
Callback to process an IDOMFont.
- virtual IDOMNodePtr [transformPath](#) (Implementation *genericImplementation, const IDOMPathNodePtr &path, bool &changed, const [CTransformState](#) &state)
Callback to process an IDOMPathNode.
- virtual IDOMNodePtr [transformVisualRoot](#) (Implementation *genericImplementation, const IDOMVisual↔RootPtr &root, bool &changed, bool transformChildren, const [CTransformState](#) &state)
Callback to process an IDOMVisualRoot node.
- virtual IDOMNodePtr [transformForm](#) (Implementation *genericImplementation, const IDOMFormPtr &form, bool &changed, bool transformChildren, const [CTransformState](#) &state)
Callback to process an IDOMForm node.
- virtual IDOMNodePtr [transformFormInstance](#) (Implementation *genericImplementation, const IDOMForm↔InstancePtr &instance, bool &changed, bool transformChildren, const [CTransformState](#) &state)
Callback to process an IDOMFormInstance node.
- virtual IDOMBrushPtr [transformBrush](#) (Implementation *genericImplementation, const IDOMBrushPtr &brush, [eBrushUsage](#) usage, const [CTransformState](#) &state)
Callback to process any kind of brush.
- virtual IDOMBrushPtr [transformSolidColorBrush](#) (Implementation *genericImplementation, const IDOMSolidColorBrushPtr &brush, const [CTransformState](#) &state)
Callback to process an IDOMSolidColorBrush.
- virtual IDOMBrushPtr [transformGradientBrush](#) (Implementation *genericImplementation, const IDOMGradient↔BrushPtr &gradient, const [CTransformState](#) &state)
Callback to process an IDOMGradientBrush.
- virtual IDOMBrushPtr [transformLinearGradientBrush](#) (Implementation *genericImplementation, const IDOMLinearGradientBrushPtr &gradient, const [CTransformState](#) &state)
Callback to process an IDOMLinearGradientBrush.
- virtual IDOMBrushPtr [transformRadialGradientBrush](#) (Implementation *genericImplementation, const IDOMRadialGradientBrushPtr &gradient, const [CTransformState](#) &state)
Callback to process an IDOMRadialGradientBrush.
- virtual IDOMBrushPtr [transformVisualBrush](#) (Implementation *genericImplementation, const IDOMVisual↔BrushPtr &brush, const [CTransformState](#) &state)
Callback to process an IDOMVisualBrush.
- virtual IDOMBrushPtr [transformImageBrush](#) (Implementation *genericImplementation, const IDOMImage↔BrushPtr &brush, const [CTransformState](#) &state)
Callback to process an IDOMImageBrush.
- virtual IDOMBrushPtr [transformTilingPatternBrush](#) (Implementation *genericImplementation, const IDOMTilingPatternBrushPtr &brush, const [CTransformState](#) &state)
Callback to process an IDOMTilingPatternBrush.
- virtual IDOMBrushPtr [transformShadingPatternBrush](#) (Implementation *genericImplementation, const IDOMShadingPatternBrushPtr &brush, const [CTransformState](#) &state)

Callback to process an [IDOMShadingPatternBrush](#).

- virtual IDOMBrushPtr [transformSoftMaskBrush](#) ([Implementation](#) *genericImplementation, const IDOMSoftMaskBrushPtr &brush, const [CTransformState](#) &state)

Callback to process an [IDOMSoftMaskBrush](#).

- virtual IDOMBrushPtr [transformMaskedBrush](#) ([Implementation](#) *genericImplementation, const IDOMMaskedBrushPtr &brush, const [CTransformState](#) &state)

Callback to process an [IDOMMaskedBrush](#).

- virtual IDOMBrushPtr [transformNullBrush](#) ([Implementation](#) *genericImplementation, const IDOMNullBrushPtr &brush, const [CTransformState](#) &state)

Callback to process an [IDOMNullBrush](#).

7.319.1 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.

7.319.2 Member Function Documentation

transformAnnotation()

```
virtual void JawsMako::ICustomTransform::Implementation::transformAnnotation (
    Implementation * genericImplementation,
    const IAnnotationPtr & annotation ) [inline], [virtual]
```

Callback to process an annotation.

Parameters

<i>genericImplementation</i>	The generic implementation that continues processing further down the tree.
<i>annotation</i>	The annotation, which you may modify.

transformAnnotationAppearance()

```
virtual IAnnotationAppearancePtr JawsMako::ICustomTransform::Implementation::transformAnnotationAppearance (
    Implementation * genericImplementation,
    const IAnnotationAppearancePtr & appearance,
    const FRect & annotationRect ) [inline], [virtual]
```

Callback to process an annotation appearance.

Parameters

<i>genericImplementation</i>	The generic implementation that continues processing further down the tree.
<i>appearance</i>	The annotation appearance. Do not modify this; instead, if you must modify it, return a cloned version containing your modifications. Do not return NULL.
<i>annotationRect</i>	The owning annotation's rectangle.

Returns

IDOMAnnotationAppearancePtr The result appearance, or the original appearance if no changes were required.

transformBrush()

```
virtual IDOMBrushPtr JawsMako::ICustomTransform::Implementation::transformBrush (
    Implementation * genericImplementation,
    const IDOMBrushPtr & brush,
    eBrushUsage usage,
    const CTransformState & state ) [inline], [virtual]
```

Callback to process any kind of brush.

Parameters

<i>genericImplementation</i>	The generic implementation that continues processing further down the tree.
<i>brush</i>	The brush. Do not edit this brush in place, instead create a copy or a new brush instead.
<i>state</i>	The current state <i>outside</i> the brush.

Returns

IDOMBrushPtr The resulting brush, or NULL if the brush should be dropped.

transformCanvas()

```
virtual IDOMNodePtr JawsMako::ICustomTransform::Implementation::transformCanvas (
    Implementation * genericImplementation,
    const IDOMCanvasPtr & canvas,
    bool & changed,
    bool transformChildren,
    const CTransformState & state ) [inline], [virtual]
```

Callback to process an **IDOMCanvas**.

Parameters

<i>genericImplementation</i>	The generic implementation that continues processing further down the tree.
<i>canvas</i>	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

<i>changed</i>	You must set this to true if any changes have been made to the node, and set it to false if you have not.
<i>transformChildren</i>	True if the children of the node should also be transformed
<i>state</i>	The current state <i>inside</i> 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::Implementation::transformCharPathGroup (
    Implementation * genericImplementation,
    const IDOMCharPathGroupPtr & group,
    bool & changed,
    bool transformChildren,
    const CTransformState & state ) [inline], [virtual]
```

Callback to process an [IDOMCharPathGroup](#).

Parameters

<i>genericImplementation</i>	The generic implementation that continues processing further down the tree.
<i>group</i>	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.
<i>changed</i>	You must set this to true if any changes have been made to the node, and set it to false if you have not.
<i>transformChildren</i>	True if the children of the node should also be transformed
<i>state</i>	The current state <i>inside</i> 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::Implementation::transformColor (
    Implementation * 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

<i>genericImplementation</i>	The generic implementation that continues processing further down the tree.
<i>color</i>	The color. Do not modify this color; instead create a clone and apply your modifications to that.
<i>state</i>	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::Implementation::transformColorSpace (
    Implementation * 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

<i>genericImplementation</i>	The generic implementation that continues processing further down the tree.
<i>colorSpace</i>	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
<i>state</i>	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::Implementation::transformFixedPage (
    Implementation * genericImplementation,
    const IDOMFixedPagePtr & page,
    bool & changed,
    bool transformChildren,
    const CTransformState & state ) [inline], [virtual]
```

Callback to process an [IDOMFixedPage](#).

Parameters

<i>genericImplementation</i>	The generic implementation that continues processing further down the tree.
<i>page</i>	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

<i>changed</i>	You must set this to true if any changes have been made to the node, and set it to false if you have not.
<i>transformChildren</i>	True if the children of the node should also be transformed
<i>state</i>	The current state when the node was encountered.

Returns

IDOMNodePtr The resulting node.

transformFont()

```
virtual IDOMFontPtr JawsMako::ICustomTransform::Implementation::transformFont (
    IImplementation * genericImplementation,
    const IDOMFontPtr & font,
    uint32 & index,
    const CTransformState & state ) [inline], [virtual]
```

Callback to process an [IDOMFont](#).

Parameters

<i>genericImplementation</i>	The generic implementation that continues processing further down the tree.
<i>font</i>	The font. You must not modify this font. Instead, create a new font or work on a clone.
<i>index</i>	The font index of the font being edited. On return, this must be the index of the resulting font.
<i>state</i>	The current state <i>inside</i> the node, with the node's attributes taken into account (such as its <code>renderTransform</code>)

Returns

IDOMFontPtr The resulting font, or the original font if no changes were required.

transformForm()

```
virtual IDOMNodePtr JawsMako::ICustomTransform::Implementation::transformForm (
    IImplementation * genericImplementation,
    const IDOMFormPtr & form,
    bool & changed,
    bool transformChildren,
    const CTransformState & state ) [inline], [virtual]
```

Callback to process an [IDOMForm](#) node.

Parameters

<i>genericImplementation</i>	The generic implementation that continues processing further down the tree.
<i>form</i>	The form. You must not edit this form directly. Instead, make a deep clone first.

Parameters

<i>changed</i>	You must set this to true if any changes have been made to the node, and set it to false if you have not.
<i>transformChildren</i>	True if the children of the node should also be transformed
<i>state</i>	The current state <i>inside</i> the node, with the node's attributes taken into account (such as its <code>renderTransform</code>)

Returns

IDOMNodePtr The resulting node, or NULL if the node should be dropped.

transformFormInstance()

```
virtual IDOMNodePtr JawsMako::ICustomTransform::Implementation::transformFormInstance (
    Implementation * genericImplementation,
    const IDOMFormInstancePtr & instance,
    bool & changed,
    bool transformChildren,
    const CTransformState & state ) [inline], [virtual]
```

Callback to process an [IDOMFormInstance](#) node.

Parameters

<i>genericImplementation</i>	The generic implementation that continues processing further down the tree.
<i>instance</i>	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.
<i>changed</i>	You must set this to true if any changes have been made to the node, and set it to false if you have not.
<i>transformChildren</i>	True if the children of the node should also be transformed
<i>state</i>	The current state <i>inside</i> the node, with the node's attributes taken into account (such as its <code>renderTransform</code>)

Returns

IDOMNodePtr The resulting node, or NULL if the node should be dropped.

transformGlyphs()

```
virtual IDOMNodePtr JawsMako::ICustomTransform::Implementation::transformGlyphs (
    Implementation * genericImplementation,
    const IDOMGlyphsPtr & glyphs,
    bool & changed,
    const CTransformState & state ) [inline], [virtual]
```

Callback to process an [IDOMGlyphs](#) node.

Parameters

<i>genericImplementation</i>	The generic implementation that continues processing further down the tree.
<i>glyphs</i>	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.
<i>changed</i>	You must set this to true if any changes have been made to the node, and set it to false if you have not.
<i>state</i>	The current state <i>inside</i> 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::Implementation::transformGradientBrush (
    Implementation * genericImplementation,
    const IDOMGradientBrushPtr & gradient,
    const CTransformState & state ) [inline], [virtual]
```

Callback to process an [IDOMGradientBrush](#).

Parameters

<i>genericImplementation</i>	The generic implementation that continues processing further down the tree.
<i>gradient</i>	The brush. Do not edit this brush in place, instead create a copy or a new brush instead.
<i>state</i>	The current state <i>inside</i> 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::Implementation::transformGroup (
    Implementation * genericImplementation,
    const IDOMGroupPtr & group,
    bool & changed,
    bool transformChildren,
    const CTransformState & state ) [inline], [virtual]
```

Callback to process an [IDOMGroup](#).

Parameters

<i>genericImplementation</i>	The generic implementation that continues processing further down the tree.
------------------------------	---

Parameters

<i>group</i>	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.
<i>changed</i>	You must set this to true if any changes have been made to the node, and set it to false if you have not.
<i>transformChildren</i>	True if the children of the node should also be transformed
<i>state</i>	The current state <i>inside</i> the node, with the node's attributes taken into account (such as its <code>renderTransform</code>)

Returns

IDOMNodePtr The resulting node, or NULL if the node should be dropped.

transformImage()

```
virtual IDOMImagePtr JawsMako::ICustomTransform::Implementation::transformImage (
    Implementation * genericImplementation,
    const IDOMImagePtr & image,
    const CTransformState & state ) [inline], [virtual]
```

Callback to process an image.

Parameters

<i>genericImplementation</i>	The generic implementation that continues processing further down the tree.
<i>image</i>	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
<i>state</i>	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::Implementation::transformImageBrush (
    Implementation * genericImplementation,
    const IDOMImageBrushPtr & brush,
    const CTransformState & state ) [inline], [virtual]
```

Callback to process an [IDOMImageBrush](#).

Parameters

<i>genericImplementation</i>	The generic implementation that continues processing further down the tree.
<i>brush</i>	The brush. Do not edit this brush in place, instead create a copy or a new brush instead.
<i>state</i>	The current state <i>inside</i> the brush, including any <code>renderTransform</code> 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::Implementation::transformLinearGradientBrush (
    (
        Implementation * genericImplementation,
        const IDOMLinearGradientBrushPtr & gradient,
        const CTransformState & state ) [inline], [virtual]
```

Callback to process an [IDOMLinearGradientBrush](#).

Parameters

<i>genericImplementation</i>	The generic implementation that continues processing further down the tree.
<i>gradient</i>	The brush. Do not edit this brush in place, instead create a copy or a new brush instead.
<i>state</i>	The current state <i>inside</i> 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::Implementation::transformMaskedBrush (
    Implementation * genericImplementation,
    const IDOMMaskedBrushPtr & brush,
    const CTransformState & state ) [inline], [virtual]
```

Callback to process an [IDOMMaskedBrush](#).

Parameters

<i>genericImplementation</i>	The generic implementation that continues processing further down the tree.
<i>brush</i>	The brush. Do not edit this brush in place, instead create a copy or a new brush instead.
<i>state</i>	The current state <i>outside</i> 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::Implementation::transformNode (
    Implementation * genericImplementation,
    const IDOMNodePtr & node,
    bool & changed,
    bool transformChildren,
    const CTransformState & state ) [inline], [virtual]
```

Callback to process an [IDOMNode](#) (of any type).

Parameters

<i>genericImplementation</i>	The generic implementation that continues processing further down the tree.
<i>node</i>	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.
<i>changed</i>	You must set this to true if any changes have been made to the node, and set it to false if you have not.
<i>transformChildren</i>	True if the children of the node should also be transformed
<i>state</i>	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::Implementation::transformNullBrush (
    Implementation * genericImplementation,
    const IDOMNullBrushPtr & brush,
    const CTransformState & state ) [inline], [virtual]
```

Callback to process an [IDOMNullBrush](#).

Parameters

<i>genericImplementation</i>	The generic implementation that continues processing further down the tree.
<i>brush</i>	The brush. Do not edit this brush in place, instead create a copy or a new brush instead.
<i>state</i>	The current state <i>inside</i> 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::Implementation::transformPath (
    Implementation * genericImplementation,
```

```

    const IDOMPathNodePtr & path,
    bool & changed,
    const CTransformState & state ) [inline], [virtual]

```

Callback to process an [IDOMPathNode](#).

Parameters

<i>genericImplementation</i>	The generic implementation that continues processing further down the tree.
<i>path</i>	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.
<i>changed</i>	You must set this to true if any changes have been made to the node, and set it to false if you have not.
<i>state</i>	The current state <i>inside</i> 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::Implementation::transformRadialGradientBrush
(
    Implementation * genericImplementation,
    const IDOMRadialGradientBrushPtr & gradient,
    const CTransformState & state ) [inline], [virtual]

```

Callback to process an [IDOMRadialGradientBrush](#).

Parameters

<i>genericImplementation</i>	The generic implementation that continues processing further down the tree.
<i>gradient</i>	The brush. Do not edit this brush in place, instead create a copy or a new brush instead.
<i>state</i>	The current state <i>inside</i> 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::Implementation::transformShadingPatternBrush
(
    Implementation * genericImplementation,
    const IDOMShadingPatternBrushPtr & brush,
    const CTransformState & state ) [inline], [virtual]

```

Callback to process an [IDOMShadingPatternBrush](#).

Parameters

<i>genericImplementation</i>	The generic implementation that continues processing further down the tree.
<i>brush</i>	The brush. Do not edit this brush in place, instead create a copy or a new brush instead.
<i>state</i>	The current state <i>inside</i> 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::Implementation::transformSoftMaskBrush (
    Implementation * genericImplementation,
    const IDOMSoftMaskBrushPtr & brush,
    const CTransformState & state ) [inline], [virtual]
```

Callback to process an [IDOMSoftMaskBrush](#).

Parameters

<i>genericImplementation</i>	The generic implementation that continues processing further down the tree.
<i>brush</i>	The brush. Do not edit this brush in place, instead create a copy or a new brush instead.
<i>state</i>	The current state <i>inside</i> 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::Implementation::transformSolidColorBrush (
    Implementation * genericImplementation,
    const IDOMSolidColorBrushPtr & brush,
    const CTransformState & state ) [inline], [virtual]
```

Callback to process an [IDOMSolidColorBrush](#).

Parameters

<i>genericImplementation</i>	The generic implementation that continues processing further down the tree.
<i>brush</i>	The brush. Do not edit this brush in place, instead create a copy or a new brush instead.
<i>state</i>	The current state <i>inside</i> 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::Implementation::transformTilingPatternBrush
(
    Implementation * genericImplementation,
    const IDOMTilingPatternBrushPtr & brush,
    const CTransformState & state ) [inline], [virtual]
```

Callback to process an [IDOMTilingPatternBrush](#).

Parameters

<i>genericImplementation</i>	The generic implementation that continues processing further down the tree.
<i>brush</i>	The brush. Do not edit this brush in place, instead create a copy or a new brush instead.
<i>state</i>	The current state <i>inside</i> 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::Implementation::transformTransparencyGroup (
    Implementation * genericImplementation,
    const IDOMTransparencyGroupPtr & group,
    bool & changed,
    bool transformChildren,
    const CTransformState & state ) [inline], [virtual]
```

Callback to process an [IDOMTransparencyGroup](#).

Parameters

<i>genericImplementation</i>	The generic implementation that continues processing further down the tree.
<i>group</i>	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.
<i>changed</i>	You must set this to true if any changes have been made to the node, and set it to false if you have not.
<i>transformChildren</i>	True if the children of the node should also be transformed
<i>state</i>	The current state <i>inside</i> 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::Implementation::transformVisualBrush (
    Implementation * genericImplementation,
    const IDOMVisualBrushPtr & brush,
    const CTransformState & state ) [inline], [virtual]
```

Callback to process an [IDOMVisualBrush](#).

Parameters

<i>genericImplementation</i>	The generic implementation that continues processing further down the tree.
<i>brush</i>	The brush. Do not edit this brush in place, instead create a copy or a new brush instead.
<i>state</i>	The current state <i>inside</i> 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::Implementation::transformVisualRoot (
    Implementation * genericImplementation,
    const IDOMVisualRootPtr & root,
    bool & changed,
    bool transformChildren,
    const CTransformState & state ) [inline], [virtual]
```

Callback to process an [IDOMVisualRoot](#) node.

Parameters

<i>genericImplementation</i>	The generic implementation that continues processing further down the tree.
<i>root</i>	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.
<i>changed</i>	You must set this to true if any changes have been made to the node, and set it to false if you have not.
<i>transformChildren</i>	True if the children of the node should also be transformed
<i>state</i>	The current state <i>inside</i> 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:

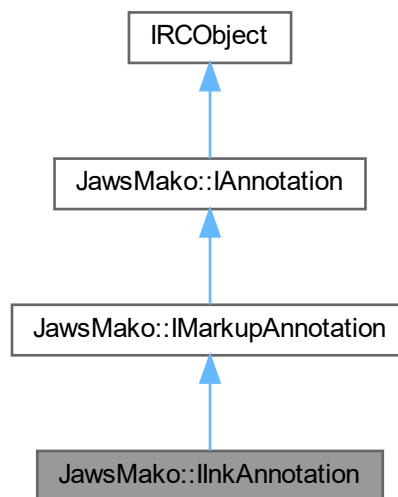
- [customtransform.h](#)

7.320 JawsMako::InkAnnotation 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::InkAnnotation:



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.

7.320.1 Detailed Description

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

7.320.2 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

<i>jawsMako</i>	The JawsMako instance
<i>rect</i>	The annotations bounds. Must not be empty.
<i>color</i>	Optional; The desired annotation color. If NULL, RGB yellow will be chosen. If provided, the color space must be DeviceRGB, DeviceCMYK or DeviceGray
<i>inkList</i>	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

<i>inkList</i>	The ink list (a collection of quadpoint data)
----------------	---

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

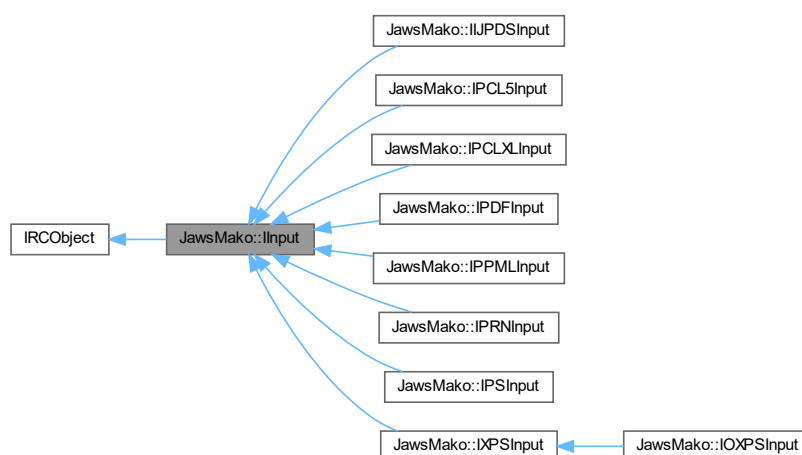
- [interactive.h](#)

7.321 JawsMako::Input 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::Input:



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](#) ¶m, 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.

7.321.1 Detailed Description

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

7.321.2 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

<i>jawsMako</i>	A smart pointer to an IJawsMako object.
<i>progressMonitor</i>	A smart pointer to an IProgressMonitor object which can be NULL if 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

<i>param</i>	Key
<i>value</i>	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 use [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

<i>sequential</i>	Whether or not to use sequential mode.
-------------------	--

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

- [jawsmako.h](#)

7.322 InputEnum< typename T > Class Reference

Iterator template class to allow iteration over a collection of instances of type <T>

```
#include <iedlenum.h>
```

7.322.1 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](#)

7.323 InputEnumRC< typename T > Class Reference

Reference-counted iterator template class to allow iteration over a collection of instances of type <T>

```
#include <iedlenum.h>
```

7.323.1 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:

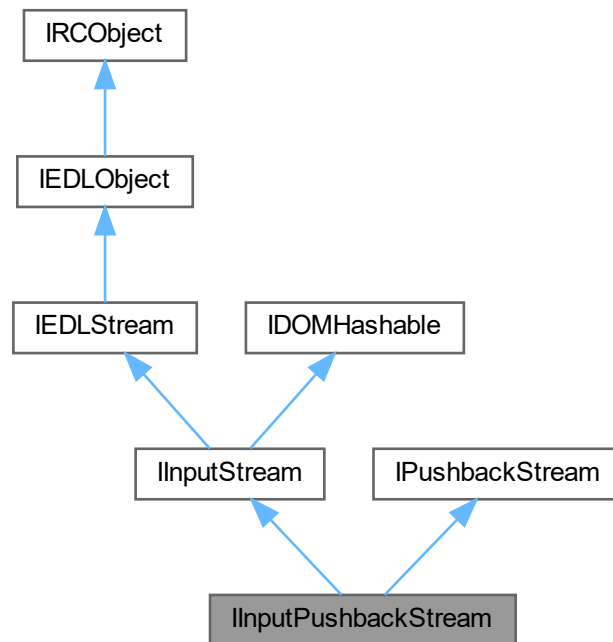
- [iedlenum.h](#)

7.324 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 EDLSysString &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, UserStreamReadFunc 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, UserRARReadFunc 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 *pFactory, 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 IInputStreamPtr &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 IInputStreamPtr &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 CInputStreamVect &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.

7.324.1 Detailed Description

Input Stream with pushback support.

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

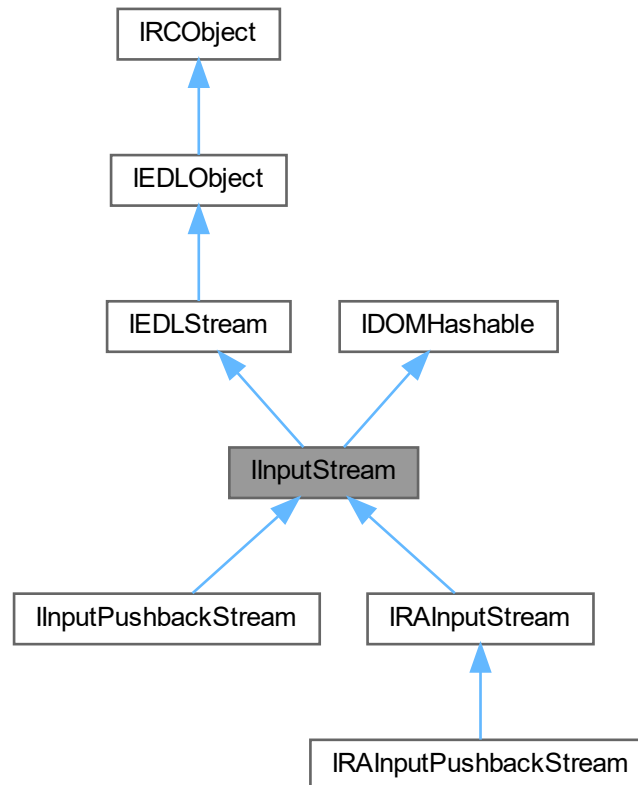
- [edlstream.h](#)

7.325 InputStream Class Reference

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

```
#include <edlstream.h>
```

Inheritance diagram for InputStream:



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 [IRCOject](#)

- virtual void [addRef](#) () const =0
Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool [decRef](#) () const =0
Decreases the reference count of the actual object pointed to. When the reference count falls to 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 EDLSysString &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, UserStreamReadFunc 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, UserRARReadFunc 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 *pFactory, 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 *InputStream* for a file on disk created with the contents of an existing stream. Throws an *IEDLError* exception on failure.

- static EDL_API *InputStreamPtr* [createSubFile](#) (*IEDLClassFactory* *pFactory, const *InputStreamPtr* &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 *InputStreamPtr* [createFromFlateCompressed](#) (*IEDLClassFactory* *pFactory, const *InputStreamPtr* &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 *InputStreamPtr* [createFromLz4Compressed](#) (*IEDLClassFactory* *pFactory, const *InputStreamPtr* &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 *InputStreamPtr* [createFromPredictorCompressed](#) (*IEDLClassFactory* *pFactory, const *InputStreamPtr* &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 *InputStreamPtr* [createCompositeStream](#) (*IEDLClassFactory* *pFactory, const *InputStreamVect* &streams)

Creation routine for creating a composite input stream representing the concatenation of a series of streams.

- static EDL_API *InputPushbackStreamPtr* [createPushbackStream](#) (*IEDLClassFactory* *pFactory, const *InputStreamPtr* &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 *InputStreamPtr* [createUelStream](#) (*IEDLClassFactory* *pFactory, const *InputStreamPtr* &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 *InputStreamPtr* [createTbcStream](#) (*IEDLClassFactory* *pFactory, const *InputStreamPtr* &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.

7.325.1 Detailed Description

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

7.325.2 Member Function Documentation

completeRead()

```
virtual bool InputStream::completeRead (
    void * buffer,
    int32 count ) [virtual]
```

Perform a complete read.

Parameters

<i>buffer</i>	Buffer to accept the data.
<i>count</i>	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

<i>buffer</i>	Address of buffer.
<i>count</i>	Number of bytes to be read.

createCompositeStream()

```
static EDL_API IInputStreamPtr IInputStream::createCompositeStream (
    IEDLClassFactory * pFactory,
    const CInputStreamVect & streams ) [static]
```

Creation routine for creating a composite input stream representing the concatenation of a series of streams.

Parameters

<i>pFactory</i>	The class factory.
<i>streams</i>	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

<i>pFactory</i>	The class factory.
<i>path</i>	Path to the file.

Returns

IRAIInputStreamPtr The new input stream.

createFromFile() [2/2]

```
static EDL_API IRAIInputStreamPtr 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

<i>pFactory</i>	The class factory.
<i>path</i>	Path to the file.

Returns

IRAIInputStreamPtr The new input stream.

createFromFileShared() [1/2]

```
static EDL_API IRAIInputStreamPtr 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

<i>pFactory</i>	The class factory.
<i>path</i>	Path to the file.

Returns

IRAIInputStreamPtr 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

<i>pFactory</i>	The class factory.
<i>path</i>	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

<i>pFactory</i>	The class factory.
<i>stream</i>	The compressed stream.
<i>raw</i>	Pass to true if the flate stream has no zlib header.
<i>ignoreChecksums</i>	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

<i>pFactory</i>	The class factory.
<i>stream</i>	The compressed stream.

Returns

InputStreamPtr 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 [InputStream](#) for data in memory. Throws an [IEDLError](#) exception on failure.

Parameters

<i>pFactory</i>	The class factory.
<i>mem</i>	A pointer to the memory to be used as the source data for the stream.
<i>length</i>	The length of the buffer pointed to by mem.
<i>copy</i>	If true, a copy of the memory pointed to by mem will be made and will be managed and released by the stream.
<i>free</i>	If copy is false and free is true, then when the stream is destroyed, mem will be deallocated using <code>stdlib free()</code> . Ignored if copy is true.

Returns

InputStreamPtr 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 [InputStream](#) for a file on disk created with the contents of an existing stream. Throws an [IEDLError](#) exception on failure.

Parameters

<i>pFactory</i>	The class factory.
<i>path</i>	Path to the file.
<i>stream</i>	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

<i>pFactory</i>	The class factory.
<i>path</i>	Path to the file.
<i>stream</i>	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 an input stream for applying a predictor algorithm to a compressed stream. Throws an [IEDLError](#) exception on failure.

Parameters

<i>pFactory</i>	The class factory.
<i>stream</i>	The compressed stream.
<i>predictor</i>	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).
<i>colors</i>	The number of interleaved color components per sample, which must be greater or equal to 1.
<i>bitsPerComponent</i>	The number of bits used to represent each color component in a sample. Valid values are 1, 2, 4, 8, and 16.
<i>columns</i>	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 [IRAIInputStream](#) from a user function that provides random access. Throws an [IEDLError](#) exception on failure.

Parameters

<i>pFactory</i>	The class factory.
<i>length</i>	The stream length.
<i>readFunc</i>	A function that when called, will provide the data for the stream.
<i>priv</i>	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

<i>pFactory</i>	The class factory.
<i>readFunc</i>	A function that when called, will provide the data for the stream.
<i>priv</i>	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

<i>pFactory</i>	The class factory.
<i>sourceStream</i>	The source stream to use.
<i>clone</i>	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

<i>pFactory</i>	The class factory. Must be a Mako instance.
<i>stream</i>	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

<i>pFactory</i>	The class factory.
<i>stream</i>	The source stream to share. Will be cloned.

Returns

IRInputStreamPtr 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

<i>pFactory</i>	The class factory.
<i>stream</i>	The stream to use for source.
<i>offset</i>	The offset within the source stream of the start of the data to use.
<i>length</i>	The length of source data to use.

Returns

InputStreamPtr The new input stream.

createTbcStream()

```
static EDL_API IInputStreamPtr IInputStream::createTbcStream (
    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

<i>pFactory</i>	The class factory.
<i>sourceStream</i>	The source stream to use.
<i>openSource</i>	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

InputStreamPtr 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

<i>pFactory</i>	The class factory.
<i>sourceStream</i>	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

<i>sourceCanonicalPath</i>	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 InputStream::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

<i>sourcePath</i>	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 InputStream::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

<i>hash</i>	A reference to receive the hash value.
-------------	--

Returns

bool True if method succeeded.

Implements [IDOMHashable](#).

read() [1/2]

```
virtual int8 InputStream::read ( ) [pure virtual]
```

Read single byte from a stream.

Returns

int8 Number of bytes actually read.

read() [2/2]

```
virtual int32 InputStream::read (
    void * buffer,
    int32 count ) [pure virtual]
```

Read specified number of bytes from a stream into buffer.

Parameters

<i>buffer</i>	Buffer to accept the data.
<i>count</i>	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

<i>count</i>	Number of bytes to skip.
--------------	--------------------------

Returns

int64 The number of bytes actually skipped.

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

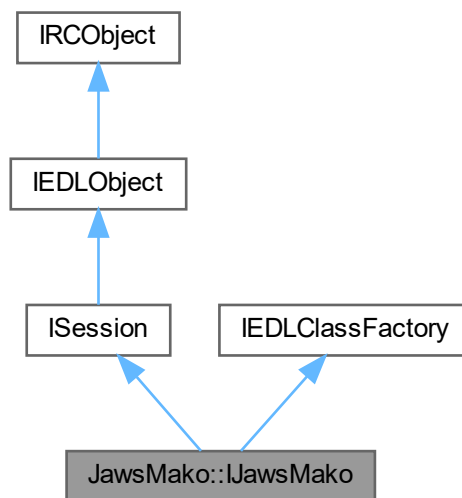
- edlstream.h

7.326 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.
- virtual bool **findFont** (const U8String &fontName)=0
Check to see if a system font with the given name is available.

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 meaningful for executable environments. Note the client must set this during initialisation 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 [IRCOject](#)

- virtual void [addRef](#) () const =0
Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool [decRef](#) () const =0
Decreases the reference count of the actual object pointed to. When the reference count falls to 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.

7.326.1 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.

7.326.2 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() [1/2]

```
virtual bool JawsMako::IJawsMako::findFont (
    const U8String & fontName ) [pure virtual]
```

Check to see if a system font with the given name is available.

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.

May throw an exception if there was an error when finding the font.

Parameters

<i>fontName</i>	The font name of the desired font.
-----------------	------------------------------------

Returns

bool true if a system font was found with a matching name, or false if no font was found.

findFont() [2/2]

```
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

<i>fontName</i>	The font name of the desired font.
<i>fontIndex</i>	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:

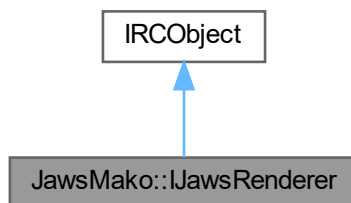
- [jawsmake.h](#)

7.327 JawsMako::IJawsRenderer Class Reference

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

```
#include <jawsmake.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 maskToInterestingNodes=false, const CSpotColorNames &retainedSpotColors=CSpotColorNames(), const IOptionalContentPtr &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 &retainedSpotColors=CSpotColorNames(), const IOptionalContentPtr &optionalContent=IOptionalContentPtr(), [eOptionalContentEvent](#) optionalContentUsage=[eOCEView](#), const CU8StringVect &processColorNames=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 [setEnabledTrueTypeNotDef](#) (bool enable)=0

Enable the use of a True Type font's /.notdef glyph in the output.

- virtual void [setEnabledRasterCompression](#) (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 [IRCOObject](#)

- virtual void **addRef** () const =0
Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool **decRef** () const =0
Decreases the reference count of the actual object pointed to. When the reference count falls to 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 IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())
Create a Jaws renderer instance.

Additional Inherited Members**Protected Member Functions inherited from [IRCOObject](#)**

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

7.327.1 Detailed Description

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

7.327.2 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

<i>jawsMako</i>	The JawsMako instance.
<i>progressMonitor</i>	A smart pointer to an IProgressMonitor object or NULL if 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

<i>node</i>	The node to render.
<i>dpi</i>	The desired resolution.
<i>depth</i>	The desired depth - either 8 or 16 bits per component.
<i>colorSpace</i>	The desired color space. If NULL, DeviceRGB will be used. If provided, this must be an instance of one of the following colorspace types: <ul style="list-style-type: none"> • DeviceRGB, • sRGB, • DeviceGray, • sGray, • DeviceCMYK, • ICCBased,
<i>bounds</i>	The desired rendering area. If the rect is empty an appropriate size is chosen based on the input node.
<i>maskToInterestingNodes</i>	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.
<i>retainedSpotColors</i>	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.
<i>optionalContent</i>	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.
<i>optionalContentUsage</i>	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

<i>node</i>	The node to render.
<i>dpi</i>	The desired resolution.
<i>quality</i>	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

<i>node</i>	The node to render.
<i>dpi</i>	The desired resolution.
<i>halftone</i>	The Halftone to use. If NULL, a 60lpi spot function will be used. If provided, this must be either a CSpotHalftone , CThresholdArrayHalftone , or CThresholdHalftone .
<i>colorSpace</i>	The desired color space. If NULL, DeviceGray will be used. If provided, only DeviceGray sGray, or a single component ICC space may be used.
<i>bounds</i>	The desired rendering area. If the rect is empty an appropriate size is chosen based on the input node.
<i>useCC</i>	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	<i>node</i>	The node to render.
in	<i>buffer</i>	The target frame buffer.
in	<i>stride</i>	The distance in bytes between the start of one scanline in the buffer to the next scanline in y.
in	<i>halftone</i>	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	<i>colorSpace</i>	The desired color space. If NULL, DeviceGray will be used. If provided, only DeviceGray or sGray may be used.
in	<i>bounds</i>	The desired rendering area. If the rect is empty an appropriate size is chosen based on the input node.
in	<i>useCC</i>	When this parameter is true then color correction is applied during rendering.
in	<i>optionalContentUsage</i>	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	<i>node</i>	The node to render.
in	<i>dpi</i>	The desired resolution.
in	<i>halftone</i>	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 .

Parameters

in	<i>colorSpace</i>	The desired color space. May be one of: <ul style="list-style-type: none"> • DeviceGray • sGray • DeviceRGB • sRGB • DeviceCMYK • One, three or four-component ICC color space
in	<i>bounds</i>	The desired rendering area. If the rect is empty an appropriate size is chosen based on the input node.
in	<i>useCC</i>	When this parameter is true then color correction is applied during rendering.
in	<i>optionalContentUsage</i>	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	<i>node</i>	The node to render.
in	<i>buffers</i>	The target frame buffers. Provide one for monochrome, or four for CMYK.
in	<i>strides</i>	The distance in bytes between the start of one scanline in each buffer to the next scanline in y, one per frame buffer.
in	<i>halftone</i>	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.

Parameters

in	<i>colorSpace</i>	The desired color space. May be one of: <ul style="list-style-type: none"> • DeviceGray • sGray • DeviceRGB • sRGB • DeviceCMYK • One, three or four-component ICC color space If NULL, DeviceGray is used.
in	<i>bounds</i>	The desired rendering area. If the rect is empty an appropriate size is chosen based on the input node.
in	<i>useCC</i>	When this parameter is true then color correction is applied during rendering.
in	<i>optionalContentUsage</i>	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

<i>node</i>	The node to render.
<i>depth</i>	The desired depth - either 8 or 16 bits per component.

Parameters

<i>colorSpace</i>	The desired color space. If NULL, DeviceRGB will be used. If provided, this must be an instance of one of the following colorspace types: <ul style="list-style-type: none"> • 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.
<i>antiAliased</i>	Anti-aliasing factor. If zero, anti-aliasing is not performed.
<i>bounds</i>	The desired rendering area. If the rect is empty an appropriate size is chosen based on the input node.
<i>destWidth</i>	The desired output width, in pixels. If zero, the input size is used.
<i>destHeight</i>	The desired output height, in pixels. If zero, the input size is used.
<i>retainedSpotColors</i>	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).
<i>optionalContent</i>	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.
<i>optionalContentUsage</i>	The desired usage for optional content. Ignored if #optionalContent is NULL. eOCEUnknown must not be used.
<i>processColorNames</i>	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".
<i>alphaGeneration</i>	Used only if the color space is Gray, RGB or CMYK based (including ICC Gray, RGB or CMYK).
<i>bandMemorySize</i>	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.
<i>ignoreSpotColors</i>	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

<i>node</i>	The node to render.
<i>depth</i>	The desired depth - either 8 or 16 bits per component.
<i>hostEndian</i>	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.
<i>destWidth</i>	The desired output width, in pixels.
<i>destHeight</i>	The desired output height, in pixels.
<i>colorSpace</i>	The desired color space. If provided, this must be an instance of one of the following colorspace types: <ul style="list-style-type: none"> • 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 <code>processColorNames</code> must be provided.
<i>frameBuffers</i>	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 <code>alphaGeneration</code> 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.
<i>antiAliased</i>	Anti-aliasing factor. If zero, anti-aliasing is not performed.
<i>bounds</i>	The desired rendering area. If the rect is empty an appropriate size is chosen based on the input node.
<i>retainedSpotColors</i>	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).
<i>optionalContent</i>	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.

Parameters

<i>optionalContentUsage</i>	The desired usage for optional content. Ignored if #optionalContent is NULL. eOCEUnknown must not be used.
<i>processColorNames</i>	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".
<i>alphaGeneration</i>	Used only if the color space is Gray, RGB or CMYK based (including ICC Gray, RGB or CMYK).
<i>bandMemorySize</i>	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.
<i>ignoreSpotColors</i>	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

<i>node</i>	The node to render.
<i>buffer</i>	The target frame buffer.
<i>destWidth</i>	The desired output width in pixels. The input content will be scaled to fit this size.
<i>destHeight</i>	The desired output height in pixels. The input content will be scaled to fit this size.
<i>stride</i>	The distance in bytes between the start of one scanline in the buffer to the next scanline in y.
<i>colorSpace</i>	The desired color space. If NULL, DeviceRGB will be used. If provided, this must be an instance of one of the following colorspace types: <ul style="list-style-type: none"> • DeviceRGB, • sRGB, • DeviceGray, • sGray, • DeviceCMYK, • ICCBased,
<i>bounds</i>	The desired rendering area. If the rect is empty an appropriate size is chosen based on the input node.

Parameters

<i>optionalContent</i>	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.
<i>optionalContentUsage</i>	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

<i>node</i>	The node to render.
<i>buffer</i>	The target frame buffer.
<i>destWidth</i>	The desired output width in pixels. The input content will be scaled to fit this size.
<i>destHeight</i>	The desired output height in pixels. The input content will be scaled to fit this size.
<i>stride</i>	The distance in bytes between the start of one scanline in the buffer to the next scanline in y.
<i>pixelStride</i>	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.
<i>reverseComponents</i>	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

<i>colorSpace</i>	The desired color space. If NULL, DeviceRGB will be used. If provided, this must be an instance of one of the following colorspace types: <ul style="list-style-type: none"> • DeviceRGB, • sRGB, • DeviceGray, • sGray, • DeviceCMYK, • ICCBased,
<i>bounds</i>	The desired rendering area. If the rect is empty an appropriate size is chosen based on the input node.
<i>optionalContent</i>	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.
<i>optionalContentUsage</i>	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 (IDMSolidColorBrush) and uncolored tiling patterns only.

The default is set by `#IColorManager::setDefaultBlackPreservation()`, which defaults to `false`. The default is applied at the time of renderer creation.

Parameters

<i>preserveForText</i>	Whether or not to apply black preservation for text.
<i>preserveForOther</i>	Whether or not to apply for non-text objects.

setEnabledRasterCompression()

```
virtual void JawsMako::IJawsRenderer::setEnabledRasterCompression (
    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.

setEnabledTrueTypeNotDef()

```
virtual void JawsMako::IJawsRenderer::setEnabledTrueTypeNotDef (
    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:

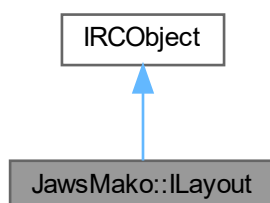
- [jawsmake.h](#)

7.328 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 ¶graphs)=0
Lay out the given paragraphs into the frames, returning a DOM representation of the result.
- virtual IDOMNodePtr [layout](#) (const ILayoutParagraphPtr ¶graph)=0
Lay out a single paragraph.
- virtual IDOMNodePtr [layout](#) (const ILayoutRunPtr &run, [ILayoutParagraph::eHorizontalAlignment](#) horizontalAlignment, 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.

7.328.1 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.

7.328.2 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

<i>frame</i>	The frame to add.
--------------	-------------------

create()

```
static JAWSMako_API ILayoutPtr JawsMako::ILayout::create (
    const IJawsMakoPtr & jawsMako ) [static]
```

Create an instance of the layout engine.

Parameters

<i>jawsMako</i>	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

<i>paragraphs</i>	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

<i>paragraph</i>	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

<i>run</i>	The run of content to lay out.
<i>horizontalAlignment</i>	The horizontal alignment to use for the ILayoutParagraph instance.
<i>spacingAfter</i>	The spacing after the paragraph, in default DOM units (96ths of an inch).
<i>spacingBefore</i>	The spacing before the paragraph, in default DOM units (96ths of an inch).
<i>leading</i>	The leading (line spacing).

Returns

IDOMNodePtr The resulting DOM.

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

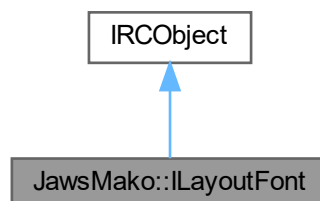
- layout.h

7.329 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 }
- enum [eFontProportion](#) { [eFPAny](#) , [eFPMonoSpaced](#) , [eFPProportional](#) }
- enum [eFontSerifStyle](#) { [eFSSAny](#) , [eFSSSerif](#) , [eFSSSansSerif](#) }

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 [ILayoutFontWeightPtr](#) &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.

7.329.1 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.

7.329.2 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.

7.329.3 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

<i>jawsMako</i>	The IJawsMako object.
<i>weight</i>	Weights used when finding fonts. When comparing weights, the <code>usWeightClass</code> field from the font OS/2 table will be used. The default value is <code>NULL</code> , which returns fonts of any weight.
<i>style</i>	The style to be used when finding fonts. The style will be compared with the bit values from the <code>fsSelection</code> field in the OS/2 table. The default value is <code>eFSAny</code> , which returns fonts of any style.
<i>proportion</i>	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 <code>isFixedPitch</code> field in the post table header, and they are expected to be consistent. The default value is <code>eFPAny</code> , which return both proportional and monospace fonts.
<i>serifStyle</i>	The Serif style used when finding fonts. This is determined using both the OS/2 table <code>sFamilyClass</code> and panose fields, and they are expected to be consistent. The default value is <code>eFSSAny</code> , which returns both Serif and Sans Serif fonts.
<i>codePage</i>	The code page character ranges used used when finding fonts. This is equivalent to the OS/2 <code>ulCodePageRange1</code> and <code>ulCodePageRange2</code> 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

<i>font</i>	Reference to the font to be returned, which will be set <code>NULL</code> if there were no matching fonts.
<i>fontIndex</i>	Reference to the index of the font within the font file.
<i>exact</i>	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:

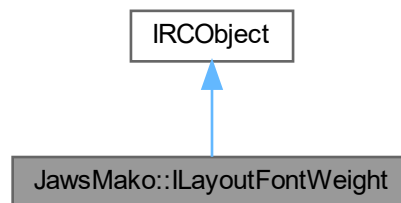
- layout.h

7.330 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.

7.330.1 Detailed Description

Font weights used by [ILayoutFont](#) for font selection.

7.330.2 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.

7.330.3 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

<i>weight</i>	The weight to use when matching. The default value of eAny will select fonts of any weight.
<i>allowedError</i>	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

<i>weightLow</i>	The lower weight range value to use when matching.
<i>weightHigh</i>	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

<i>weights</i>	The weights to use when matching.
<i>allowedError</i>	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

<i>weight</i>	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

<i>weight</i>	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

<i>weight</i>	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:

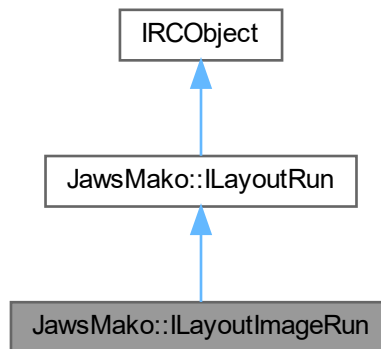
- layout.h

7.331 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.

7.331.1 Detailed Description

A run of image content to be added to an [ILayoutParagraph](#).

7.331.2 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

<i>jawsMako</i>	The IJawsMako object.
<i>image</i>	The desired image for the run.
<i>width</i>	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.
<i>height</i>	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:

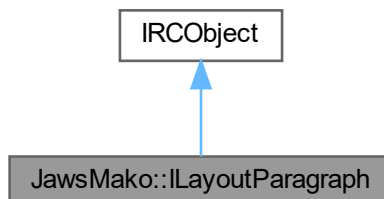
- layout.h

7.332 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 [IRCOject](#)

- virtual void [addRef](#) () const =0
Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool [decRef](#) () const =0
Decreases the reference count of the actual object pointed to. When the reference count falls to 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=[eHLeft](#), 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.

7.332.1 Detailed Description

A paragraph, consisting of a number of runs of content.

7.332.2 Member Enumeration Documentation

[eHorizontalAlignment](#)

```
enum JawsMako::ILayoutParagraph::eHorizontalAlignment
```

Horizontal alignment or justification for the paragraph.

Enumerator

eHLeft	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.

7.332.3 Member Function Documentation

[addRun\(\)](#)

```
virtual void JawsMako::ILayoutParagraph::addRun (  
    const ILayoutRunPtr & run ) [pure virtual]
```

Add a run to the end of the paragraph.

Parameters

<i>run</i>	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

<i>horizontalAlignment</i>	The horizontal alignment for this paragraph.
<i>spacingAfter</i>	The spacing after the paragraph, in default DOM units (96ths of an inch).
<i>spacingBefore</i>	The spacing before the paragraph, in default DOM units (96ths of an inch).
<i>leading</i>	The leading (line spacing).
<i>defaultColor</i>	The default color to use if an enclosed run does not specify a color. If null, a DeviceGray black will be used.
<i>defaultFont</i>	An IDOMOpenTypeFont instance used as a default if a if an enclosed run does not specify a font.
<i>defaultFontIndex</i>	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.
<i>defaultFontSize</i>	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

<i>horizontalAlignment</i>	The horizontal alignment for this paragraph.
<i>spacingAfter</i>	The spacing after the paragraph, in default DOM units (96ths of an inch).
<i>spacingBefore</i>	The spacing before the paragraph, in default DOM units (96ths of an inch).
<i>leading</i>	The leading (line spacing).
<i>defaultColor</i>	The default color to use if an enclosed run does not specify a color. If null, a DeviceGray black will be used.
<i>defaultFont</i>	An ILayoutFont instance used to select a font to use as a default if an enclosed run does not specify a font. Optional.
<i>exact</i>	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.
<i>defaultFontSize</i>	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

<i>before</i>	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

<i>defaultColor</i>	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

<i>defaultFont</i>	The font to use as a default if an enclosed run does not specify a font. Optional.
<i>defaultFontIndex</i>	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

<i>defaultFont</i>	An ILayoutFont instance used to select a font to use as a default if an enclosed run does not specify a font. Optional.
<i>exact</i>	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

<i>defaultFontSize</i>	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

<i>leading</i>	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

<i>spacing</i>	The spacing in default DOM units (96ths of an inch).
<i>before</i>	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:

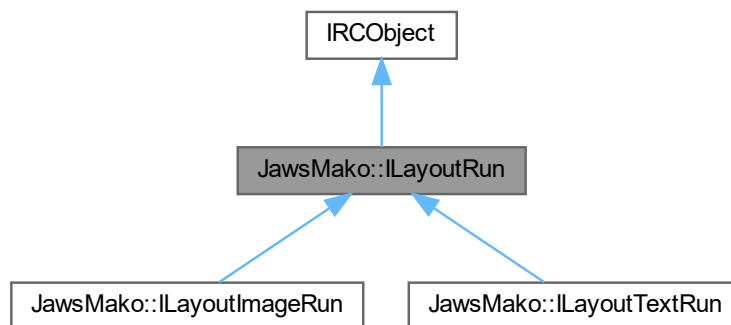
- layout.h

7.333 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.

7.333.1 Detailed Description

A run of content to be added to an `ILayoutParagraph`.

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

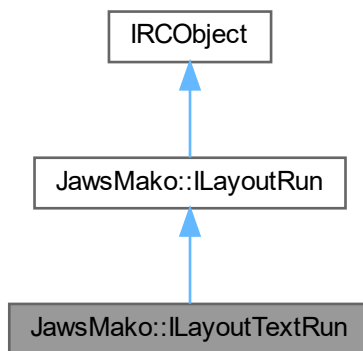
- layout.h

7.334 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.

7.334.1 Detailed Description

A run of text content to be added to an [ILayoutParagraph](#).

7.334.2 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

<i>text</i>	The text for the run as a wide character Unicode string.
<i>font</i>	The desired font for the run. If nullptr, the paragraph default font (which must be set in this circumstance) will be used.
<i>fontIndex</i>	The index of the font within the font file. Required if the font is specified and is in a TrueType Collection, ignored otherwise.
<i>size</i>	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.
<i>color</i>	The color to use for the text. If null, the paragraph default font color will be used.
<i>letterSpacing</i>	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

<i>text</i>	The text for the run as a wide character Unicode string.
<i>font</i>	An ILayoutFont instance used to select a font for the run. If nullptr, the paragraph default font (which must be set) will be used.
<i>size</i>	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.
<i>exact</i>	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.
<i>color</i>	The color to use for the text.
<i>letterSpacing</i>	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

<i>text</i>	The text for the run as UTF-8.
<i>font</i>	The desired font for the run. If nullptr, the paragraph default font (which must be set) will be used.

Parameters

<i>fontIndex</i>	The index of the font within the font file. Required if the font is specified and is in a TrueType Collection, ignored otherwise.
<i>size</i>	The desired em size of the text, in default DOM units (96ths of an inch).
<i>color</i>	The color to use for the text.
<i>letterSpacing</i>	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

<i>text</i>	The text for the run as UTF-8.
<i>font</i>	An ILayoutFont instance used to select a font for the run. If nullptr, the paragraph default font (which must be set) will be used.
<i>size</i>	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.
<i>exact</i>	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.
<i>color</i>	The color to use for the text.
<i>letterSpacing</i>	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

<i>color</i>	The text color.
--------------	-----------------

setLetterSpacing()

```
virtual void JawsMako::ILayoutTextRun::setLetterSpacing (
    double letterSpacing ) [pure virtual]
```

Set the letter spacing.

Parameters

<i>letterSpacing</i>	The letter spacing, in ems.
----------------------	-----------------------------

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

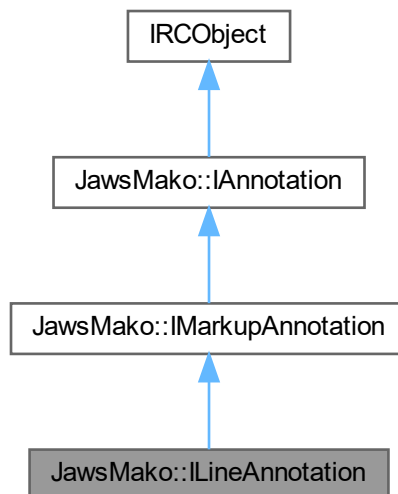
- layout.h

7.335 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.

7.335.1 Detailed Description

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

7.335.2 Member Function Documentation**getCaptionOffset()**

```
virtual void JawsMako::ILineAnnotation::getCaptionOffset (
    float & xOffset,
    float & yOffset ) const [pure virtual]
```

Get the caption offset.

Parameters

<i>xOffset</i>	The xOffset value
<i>yOffset</i>	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

<i>start</i>	Start point
<i>end</i>	End point

setInteriorColor()

```
virtual void JawsMako::ILinkAnnotation::setInteriorColor (
    const IDOMColorPtr & color ) [pure virtual]
```

Set the interior color to be used to fill the line endings.

Parameters

<i>color</i>	The color definition, or pass NULL to remove the color
--------------	--

setLineEndpoints()

```
virtual void JawsMako::ILinkAnnotation::setLineEndpoints (
    const FPoint & start,
    const FPoint & end ) [pure virtual]
```

Set the line end points. The points are relative to the annotation rect.

Parameters

<i>start</i>	Start point
<i>end</i>	End point

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

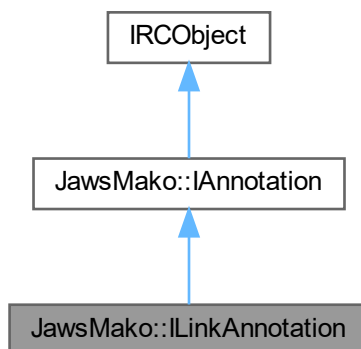
- [interactive.h](#)

7.336 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 [IRCOject](#)

- virtual void [addRef](#) () const =0
 - Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.*
- virtual bool [decRef](#) () const =0
 - Decreases the reference count of the actual object pointed to. When the reference count falls to 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::ILinkAnnotation

- 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.

7.336.1 Detailed Description

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

7.336.2 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

<i>jawsMako</i>	The JawsMako instance
<i>rect</i>	The annotations bounds. Must not be empty.
<i>color</i>	Optional; The desired annotation color. If NULL, no line will be visible.
<i>actionOrDest</i>	Optional; The Action dictionary or Dest object to be invoked for this annotation. If NULL, no action is performed.
<i>highlightMode</i>	Optional; The highlight mode for the annotation when the annotation is clicked. If NULL, it uses the default which is Invert.
<i>pa</i>	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

<i>actionOrDest</i>	The action or dest object.
---------------------	----------------------------

setHighlightMode()

```
virtual void JawsMako::ILinkAnnotation::setHighlightMode (
    const IPDFNamePtr & highlightMode ) [pure virtual]
```

Set the Highlight mode for this annotation.

Parameters

<i>highlightMode</i>	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

<i>pa</i>	The previous action dictionary object.
-----------	--

setQuadPoints()

```
virtual void JawsMako::ILinkAnnotation::setQuadPoints (
    const CQuadPointVect & quadPoints ) [pure virtual]
```

Set the link annotation's quad points.

Parameters

<i>quadPoints</i>	The link annotation's quad points
-------------------	-----------------------------------

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

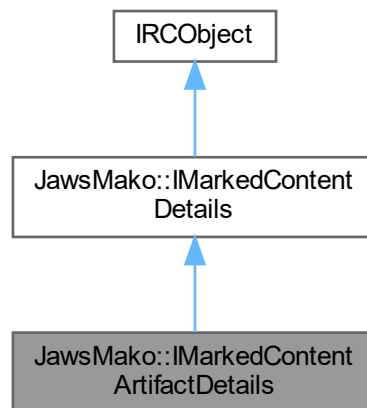
- [interactive.h](#)

7.337 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.

7.337.1 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:

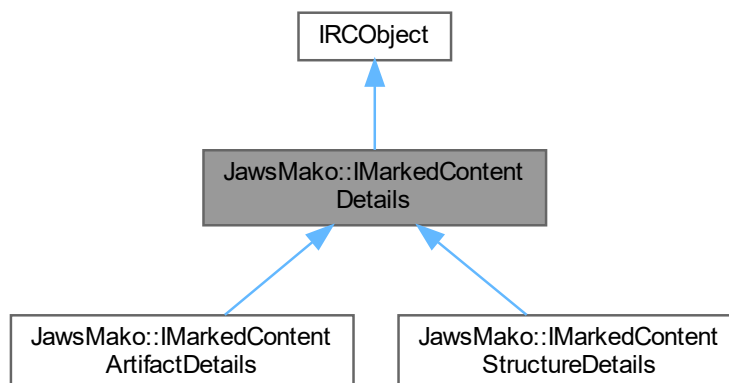
- [structure.h](#)

7.338 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.

7.338.1 Detailed Description

Details of Marked Content applied to an [IDOMGroup](#).

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

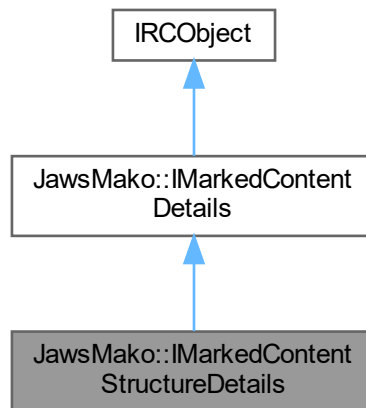
- [structure.h](#)

7.339 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 IStructureElementReferencePtr &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.

7.339.1 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:

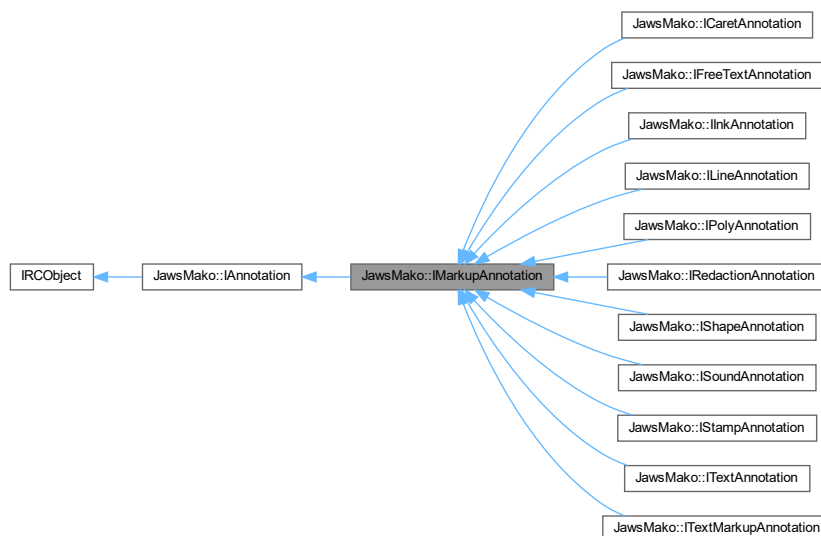
- [structure.h](#)

7.340 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.

7.340.1 Detailed Description

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

7.340.2 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

<i>author</i>	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

<i>creationTime</i>	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

<i>opacity</i>	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

<i>popupReference</i>	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

<i>popup</i>	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](#)

7.341 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.

7.341.1 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.

7.341.2 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

<code>size</code>	The requested media size, which may be an integer or a string.
<code>pageWidth</code>	The default width that will be set for the requested media size. Users may set this value to override the default page width.
<code>pageHeight</code>	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:

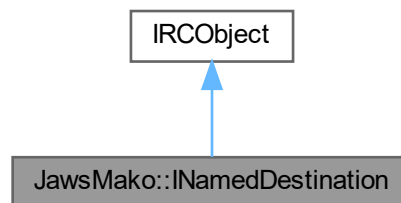
- [jawsmako.h](#)

7.342 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 IDOMPPageRectTargetPtr [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 IDOMPPageRectTargetPtr &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.

7.342.1 Detailed Description

A named destination in a PDF Document.

7.342.2 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 IDOMPPageRectTargetPtr & target ) [static]
```

Create a named destination with the given name to the given target.

Parameters

<i>jawsMako</i>	The JawsMako instance
<i>name</i>	The name of the destination
<i>target</i>	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:

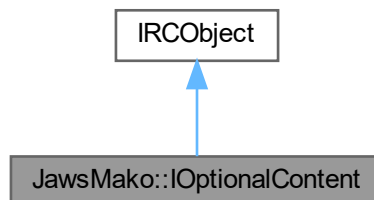
- [interactive.h](#)

7.343 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](#) &newGroup← GroupName, bool visible)

Make the given DOM node optional with a new optional content group.
Convenience.

- virtual bool **groupsVisible** (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 **groupsVisible** (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 **groupShouldBelgnored** (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 **groupShouldBelgnored** (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 **groupsPresent** (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 IOptionalContentConfigurationPtr &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.

7.343.1 Detailed Description

Root level optional content information for an entire document.

7.343.2 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

`groupsVisible()` [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

<code>group</code>	The group to check.
<code>event</code>	The event representing the intended use. Pass <code>eOCEUnknown</code> if the automatic state IOptionalContentGroupUsageApplication should not be checked.
<code>configuration</code>	The configuration to use. Pass <code>NULL</code> to use the default configuration.

groupsVisible() [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

<i>groupRef</i>	Reference to the group to check.
<i>event</i>	The event representing the intended use. Pass eOCEUnknown if the automatic state IOptionalContentGroupUsageApplication should not be checked.
<i>configuration</i>	The configuration to use. Pass NULL to use the default configuration.

groupShouldBelgnored() [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

<i>group</i>	The group to check.
<i>event</i>	The event representing the intended use. Pass eOCEUnknown if the automatic state IOptionalContentGroupUsageApplication should not be checked.
<i>configuration</i>	The configuration to use. Pass NULL to use the default configuration.

groupShouldBelgnored() [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

<i>groupRef</i>	Reference to the group to check.
-----------------	----------------------------------

Parameters

<i>event</i>	The event representing the intended use. Pass <code>eOCEUnknown</code> if the automatic state IOptionalContentGroupUsageApplication should not be checked.
<i>configuration</i>	The configuration to use. Pass <code>NULL</code> 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:

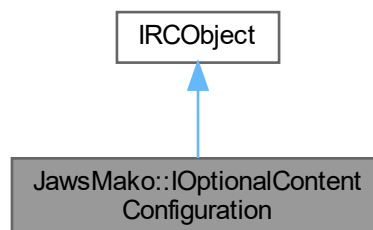
- [optionalcontent.h](#)

7.344 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 configuration.
- 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.*

7.344.1 Detailed Description

A configuration for optional content.

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

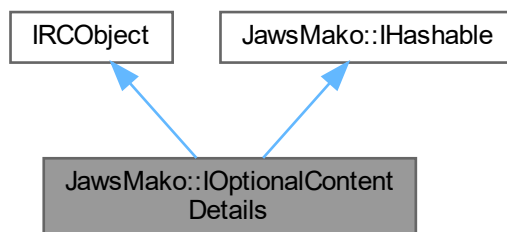
- [optionalcontent.h](#)

7.345 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](#) { [eVPAIOn](#) , [eVPAnyOn](#) , [eVPAnyOff](#) , [eVPAIOff](#) }

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](#) &optionalContent=[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 visibility 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 IOptionalContentGroupReferencePtr &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 IOptionalContentVisibilityExpressionPtr &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.

7.345.1 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.

7.345.2 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

eVPAIOn	This content is visible if all the dependent groups are visible.
eVPAAnyOn	This content is visible if any dependent group is visible.
eVPAAnyOff	This content is visible if any dependent group is invisible.
eVPAIOff	This content is visible if all the dependent groups are invisible.

7.345.3 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:

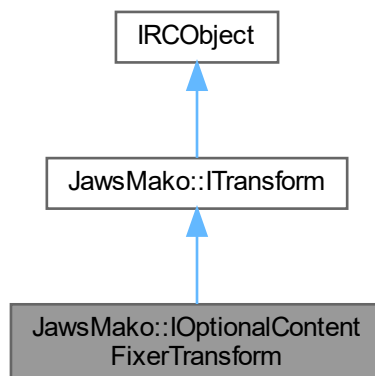
- [optionalcontent.h](#)

7.346 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 **IRCOject**

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

7.346.1 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.

7.346.2 Member Function Documentation

`create()`

```
static JAWSMAKO_API IOptionalContentFixerTransformPtr JawsMako::IOptionalContentFixerTransform↔
::create (
    const IJawsMakoPtr & jawsMako,
    const IAbortPtr & abort = IAbortPtr() ) [static]
```

Create the transform.

Parameters

<code>jawsMako</code>	The JawsMako instance.
<code>abort</code>	An abort callback handler.

Returns

The new instance.

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

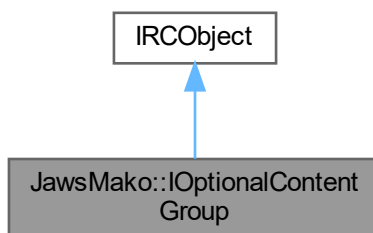
- [transforms.h](#)

7.347 JawsMako::IOptionalContentGroup Class Reference

Interface for an optional content group.

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


Inheritance diagram for JawsMako::OptionalContentGroup:



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.

7.347.1 Detailed Description

Interface for an optional content group.

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

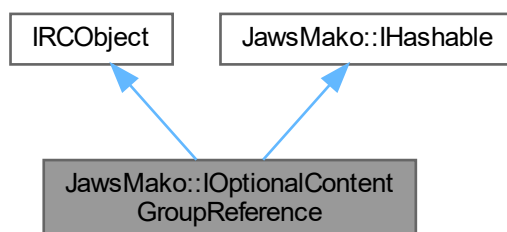
- [optionalcontent.h](#)

7.348 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.

7.348.1 Detailed Description

A reference to an optional content group.

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

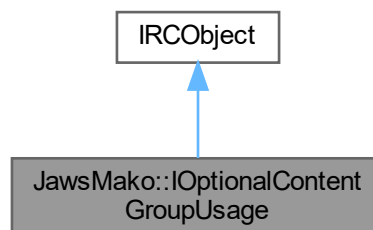
- [optionalcontent.h](#)

7.349 [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 `OptionalContentGroupUsagePtr` **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 **getLanguagesPreferred** () 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
Determine 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.

7.349.1 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 [OptionalContent](#) 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 [#OptionalContentGroupUsageApplication](#) below.

7.349.2 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

<i>type</i>	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]
```

Determine 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

<i>language</i>	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.
<i>preferred</i>	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

<i>type</i>	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

<i>type</i>	The type of user, which should be either "Ind" (Individual), "Tit" (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)
<i>users</i>	The user names.

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

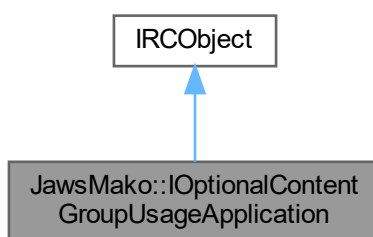
- [optionalcontent.h](#)

7.350 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 **groupsAffected** (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 &jawsMako, eOptionalContentEvent event=eOCEView)
Create usage application information.

Additional Inherited Members**Protected Member Functions inherited from IRCObject**

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

7.350.1 Detailed Description

Interface for controlling how [IOptionalContentGroupUsage](#) is applied, and for what groups.

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

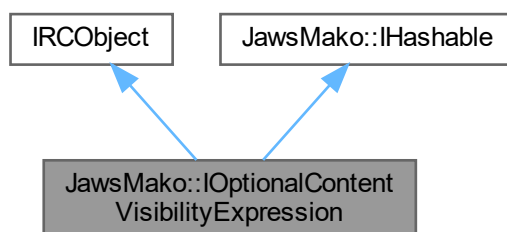
- [optionalcontent.h](#)

7.351 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 `IRCOject`

- virtual void **addRef** () const =0
Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool **decRef** () const =0
Decreases the reference count of the actual object pointed to. When the reference count falls to 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 `IOptionalContentGroupReferencePtr` &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 `IOptionalContentVisibilityExpressionPtr` &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 `COptionalContentVisibilityExpressionVect` &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 `COptionalContentVisibilityExpressionVect` &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.

7.351.1 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.

7.351.2 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.

7.351.3 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:

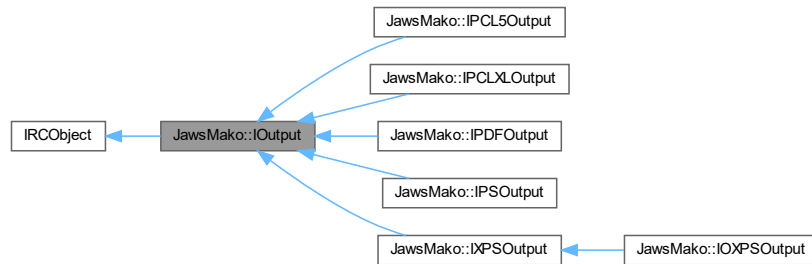
- [optionalcontent.h](#)

7.352 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](#) ¶m, 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.

7.352.1 Detailed Description

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

7.352.2 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

<i>jawsMako</i>	The JawsMako instance.
<i>format</i>	The file format of the output file.
<i>progressMonitor</i>	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

<i>assembly</i>	The document assembly for output. The assembly need not be populated with documents.
<i>stream</i>	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

<i>assembly</i>	The document assembly for output. The assembly need not be populated with documents.
<i>pathToFile</i>	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

<i>assembly</i>	The document assembly for output. The assembly need not be populated with documents.
<i>pathToFile</i>	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: `setAllowedPermissions(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

<i>assembly</i>	The document assembly to be written.
<i>stream</i>	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

<i>assembly</i>	The document assembly to be written.
<i>pathToFile</i>	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

<i>assembly</i>	The document assembly to be written.
<i>pathToFile</i>	The path to the output file on disk.

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

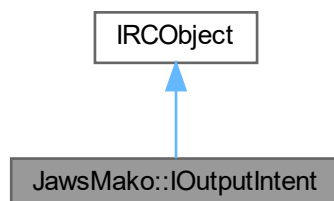
- [jawsmako.h](#)

7.353 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 IDOMICCPProfilePtr **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 ®istryName, const U8String &info, const IDOMICCPProfilePtr &profile=IDOMICCPProfilePtr(), 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.

7.353.1 Detailed Description

Interface class representing a PDF output intent.

7.353.2 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 ([IFileSpecAsEmbeddedData](#)).

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:

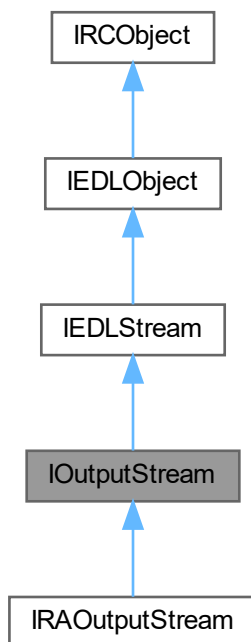
- `outputintent.h`

7.354 `IOStream` Class Reference

Generic output stream. Abstract base class for output streams.

```
#include <edlstream.h>
```

Inheritance diagram for `IOStream`:



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.

7.354.1 Detailed Description

Generic output stream. Abstract base class for output streams.

7.354.2 Member Function Documentation

[completeWrite\(\)](#) [1/2]

```
virtual bool IOutputStream::completeWrite (  
    const char * str ) [inline], [virtual]
```

Perform a complete write.

Parameters

<i>str</i>	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

<i>buffer</i>	Address of buffer.
<i>count</i>	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

<i>str</i>	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

<i>buffer</i>	Address of buffer.
<i>count</i>	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

<i>inStream</i>	The source stream.
<i>outStream</i>	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

<i>pFactory</i>	The class factory.
<i>writeFunc</i>	A function that when called, will write the data for the stream.
<i>priv</i>	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

<i>pFactory</i>	The EDL class factory.
<i>path</i>	Path to the file.
<i>append</i>	Specify what to do if the file exists. Optional <ul style="list-style-type: none"> • 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

<i>pFactory</i>	The EDL class factory.
<i>path</i>	Path to the file.
<i>append</i>	Specify what to do if the file exists. Optional <ul style="list-style-type: none"> 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

<i>pFactory</i>	The class factory.
<i>stream</i>	The compressed stream.
<i>compressionLevel</i>	Values are in the range 1 (fastest, least compression) through 9 (slowest, most compression).
<i>raw</i>	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

<i>pFactory</i>	The class factory.
<i>stream</i>	The compressed stream.
<i>openSourceStream</i>	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

<i>str</i>	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

<i>fmt</i>	C-style string that contains the text to be written to the output.
...	Additional arguments as specified by <i>fmt</i> .

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

<i>fmt</i>	C-style string that contains the text to be written to the output.
...	Additional arguments as specified by <i>fmt</i> .

`writeStream()`

```
static EDL_API int64 IOutputStream::writeStream (
    const IInputStreamPtr & inStream,
    const IOutputStreamPtr & outputStream ) [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

<i>inStream</i>	The source stream.
<i>outStream</i>	The destination stream.

Returns

int64 The number of bytes written.

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

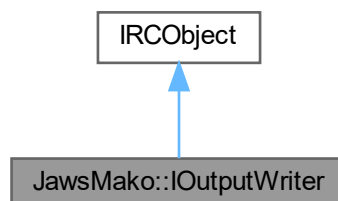
- edlstream.h

7.355 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 \sim [IRCObject](#) ()
Virtual destructor.

7.355.1 Detailed Description

A writer for writing individual pages and documents to an output in piecemeal fashion.

7.355.2 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

<i>page</i>	The page to write.
-------------	--------------------

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

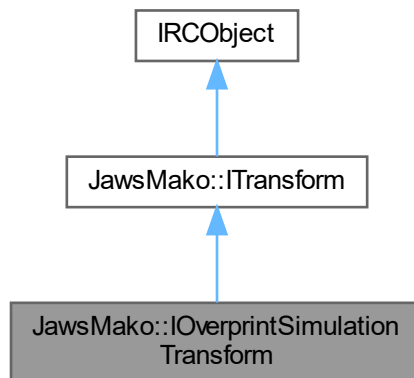
- [jawsmako.h](#)

7.356 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.

7.356.1 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 (Device↔CMYK 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.

7.356.2 Member Function Documentation

create()

```
static JAWSMAKO_API IOverprintSimulationTransformPtr JawsMako::IOverprintSimulationTransform↔
::create (
    const IJawsMakoPtr & jawsMako,
    const IAbortPtr & abort = IAbortPtr() ) [static]
```

Create the transform.

Parameters

<i>jawsMako</i>	The JawsMako instance.
<i>abort</i>	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:

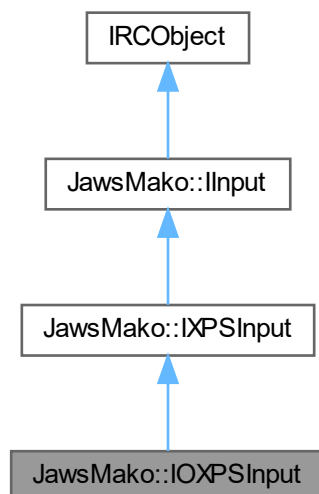
- [transforms.h](#)

7.357 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](#) ¶m, 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 [IRCOObject](#)

- virtual void [addRef](#) () const =0
Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool [decRef](#) () const =0
Decreases the reference count of the actual object pointed to. When the reference count falls to 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 [IRCOObject](#)

- virtual [~IRCOObject](#) ()
Virtual destructor.

7.357.1 Detailed Description

An instance of the JawsMako OpenXPS input class.

7.357.2 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

<i>jawsMako</i>	The IJawsMako object
<i>progressMonitor</i>	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:

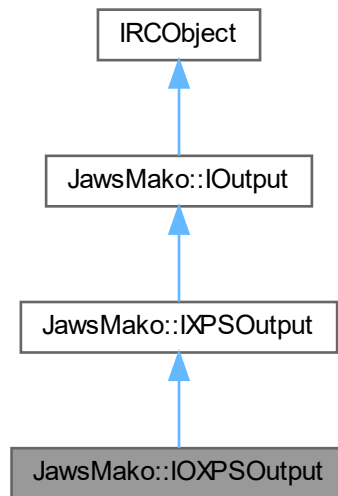
- xpsinput.h

7.358 JawsMako::IOXPSOutput Class Reference

Interface for the OpenXPS [IOutput](#) class.

```
#include <xpsoutput.h>
```


Inheritance diagram for JawsMako::IOXPSOutput:



Static Public Member Functions

- static JAW SMAKO_API IXPSOutputPtr [create](#) (const IJawsMakoPtr &jawsMako, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())
Create an OpenXPS Output instance.

Static Public Member Functions inherited from [JawsMako::IXPSOutput](#)

- static JAW SMAKO_API IXPSOutputPtr [create](#) (const IJawsMakoPtr &jawsMako, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())
Create an XPS Output instance.

Static Public Member Functions inherited from [JawsMako::IOutput](#)

- static JAW SMAKO_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::eDownsamplingMethod](#)=[IDOMImageDownsamplerFilter::eBicubic](#))=0
Set the desired maximum resolution, threshold and downsampling method for color images.
- virtual void [setGrayImageMaxResolution](#) (float resolution, float threshold=0.0f, [IDOMImageDownsamplerFilter::eDownsamplingMethod](#)=[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::eDownsamplingMethod](#)=[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](#) ¶m, 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.

7.358.1 Detailed Description

Interface for the OpenXPS [IOutput](#) class.

7.358.2 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

<code>jawsMako</code>	The IJawsMako object.
<code>progressMonitor</code>	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:

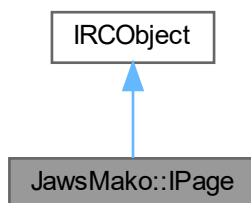
- [xpsoutput.h](#)

7.359 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 **isLoading** ()=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 **getPageld** ()=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.

7.359.1 Detailed Description

A page from an [IDocument](#), allowing high level page management, and providing on-demand access to page contents.

7.359.2 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

<i>fixed</i>	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

<i>farReference</i>	The far reference to resolve.
<i>store</i>	A reference to receive the store where the object was located.
<i>document</i>	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

<i>index</i>	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

<i>mediaBox</i>	The desired MediaBox.
-----------------	-----------------------

setPageLabel()

```
virtual void JawsMako::IPage::setPageLabel (
    const IPageLabelPtr & pageLabel ) [pure virtual]
```

Set the page label.

Parameters

<i>pageLabel</i>	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:

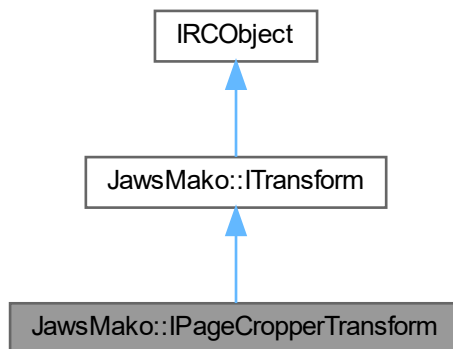
- [jawsmake.h](#)

7.360 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.

7.360.1 Detailed Description

Very simple transform for cropping pages to one of the standard boxes.

7.360.2 Member Function Documentation

create()

```
static JAWSMAKO_API IPageCropperTransformPtr JawsMako::IPageCropperTransform::create (
    const IJawsMakoPtr & jawsMako,
    const IAbortPtr & abort = IAbortPtr() ) [static]
```

Create the transform.

Parameters

<i>jawsMako</i>	The JawsMako instance.
<i>abort</i>	An abort callback handler.

Returns

The new instance.

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

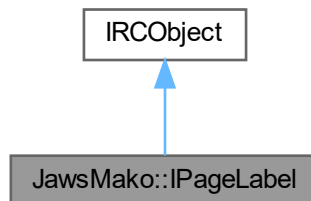
- [transforms.h](#)

7.361 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.

7.361.1 Detailed Description

Interface class representing a PDF page label.

7.361.2 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.

7.361.3 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

<i>jawsMako</i>	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

<i>number</i>	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

<i>prefix</i>	The label prefix.
---------------	-------------------

setStyle()

```
virtual void JawsMako::IPageLabel::setStyle (
    eLabelStyle style ) [pure virtual]
```

Set the page label numbering style.

Parameters

<i>style</i>	The label numbering style.
--------------	----------------------------

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

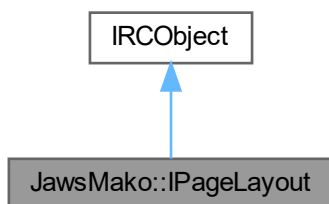
- pagelabels.h

7.362 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=ePAAI)=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 `IRCOObject`

- virtual void **addRef** () const =0
Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool **decRef** () const =0
Decreases the reference count of the actual object pointed to. When the reference count falls to 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 `IRCOObject`

- virtual `~IRCOObject` ()
Virtual destructor.

7.362.1 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.

7.362.2 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

<i>factory</i>	The factory to use
<i>runs</i>	The array 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

<i>factory</i>	The factory to use.
<i>page</i>	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

<i>threshold</i>	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

<i>multipleSpaces</i>	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

<i>virtualSpaceThreshold</i>	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:

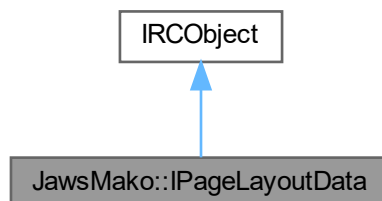
- [text.h](#)

7.363 JawsMako::IPageLayoutData Class Reference

Provides a representation of the analyzed page layout by organizing and allowing access to collections of IPageLayoutNodes.

```
#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.

7.363.1 Detailed Description

Provides a representation of the analyzed page layout by organizing and allowing access to collections of IPageLayoutNodes.

7.363.2 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

<i>data</i>	The collection of IPageLayoutNodes
-------------	------------------------------------

Returns

IPageLayoutDataPtr The new representation

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

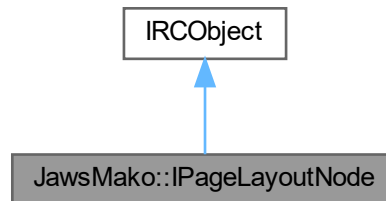
- [text.h](#)

7.364 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.

7.364.1 Detailed Description

Simple data type representing a part of an analyzed page.

7.364.2 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

<i>rect</i>	The bounds of the text area
<i>type</i>	The layout node type
<i>content</i>	The Unicode string content
<i>contentGlyphs</i>	The glyphs node from which the content was obtained
<i>columnNumber</i>	The zero-indexed column number

Returns

IPageLayoutNodePtr The new node

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

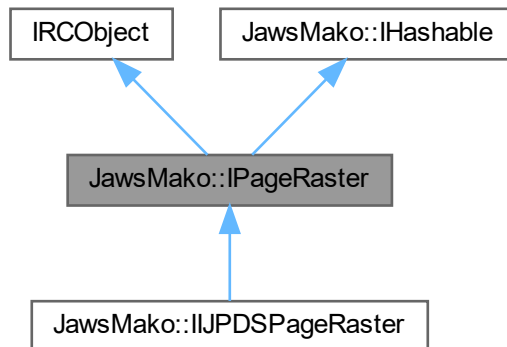
- [text.h](#)

7.365 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.

7.365.1 Detailed Description

A rasterized equivalent of a page from an [IDocument](#).

7.365.2 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:

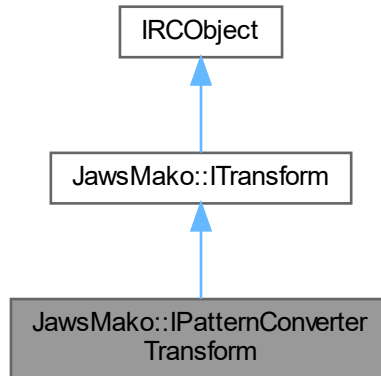
- [jawsmako.h](#)

7.366 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=eBUNGeneral, 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 transformAnnotations=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.

7.366.1 Detailed Description

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

7.366.2 Member Function Documentation

create()

```
static JAWSMako_API IPatternConverterTransformPtr JawsMako::IPatternConverterTransform::create
(
    const IJawsMakoPtr & jawsMako,
    const IAbortPtr & abort = IAbortPtr() ) [static]
```

Create the transform.

Parameters

<i>jawsMako</i>	The JawsMako instance.
<i>abort</i>	An abort callback handler.

Returns

The new instance.

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

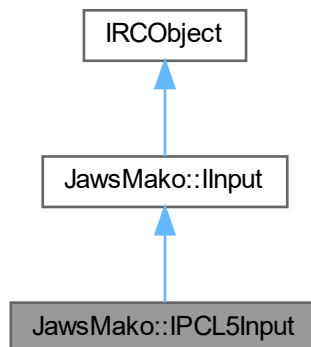
- [transforms.h](#)

7.367 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 `setDefaultFromPjl` (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` (IRCObjPtr &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 ¶m, 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 [IRCOObject](#)

- virtual [~IRCOObject](#) ()

Virtual destructor.

7.367.1 Detailed Description

An instance of the JawsMako PCL5 input class.

7.367.2 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

<i>jawsMako</i>	The IJawsMako object
<i>progressMonitor</i>	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 interrupted 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

setDefaultFromPjl()

```
virtual void JawsMako::IPCL5Input::setDefaultFromPjl (
    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

<i>flatten</i>	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

<i>permanentResourceStore</i>	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

<i>resolution</i>	The ROP rendering resolution to use.
-------------------	--------------------------------------

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

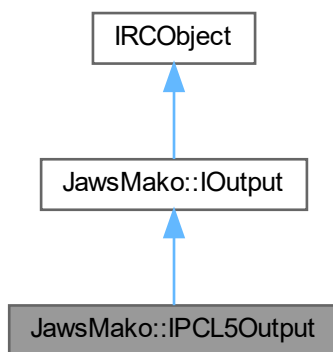
- [pcl5input.h](#)

7.368 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 "PCL5Version" 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](#) ¶m, 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.

7.368.1 Detailed Description

Interface for the PCL5 [IOutput](#) class.

7.368.2 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

<i>jawsMako</i>	The JawsMako instance.
<i>progressMonitor</i>	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".

setEnabledTrueTypeNotDef()

```
virtual void JawsMako::IPCL5Output::setEnabledTrueTypeNotDef (
    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

<i>compression</i>	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

<i>include</i>	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

<i>mediaSource</i>	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

<i>resolution</i>	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 "PCL5Version" using the string value "pcl5e" or "pcl5c".

Parameters

<i>version</i>	The version to use.
----------------	---------------------

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

- [pcl5output.h](#)

7.369 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.

7.369.1 Detailed Description

Interface allowing users to monitor/handle illegal PCL/XL operator attributes.

7.369.2 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

<code>operatorName</code>	The operator name as listed in the PCL/XL reference.
<code>attributeId</code>	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:

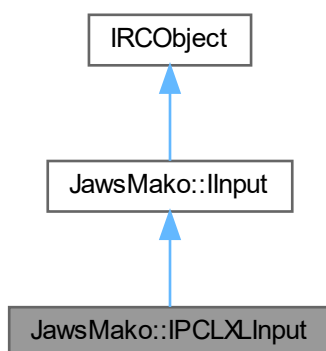
- [pclxlinput.h](#)

7.370 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 [setDefaultFromPjl](#) ([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` ¶m, 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.

7.370.1 Detailed Description

An instance of the JawsMako PCL/XL input class.

7.370.2 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

<i>jawsMako</i>	The IJawsMako object
<i>progressMonitor</i>	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 interrupted 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
- jdoubelepostcard
- jdoubelepostcard
- a5
- a6
- jb6

The matching is case-insensitive.

The default page size is a4

setDefaultFromPjl()

```
virtual void JawsMako::IPCLXLInput::setDefaultFromPjl (
    IPJLParserPtr & pjlParser ) [pure virtual]
```

Take initialisation data from the given PJI parser.

Uses the environment in the given PJI parser to set defaults in the PCL/XL input. Any previously set defaults will be ignored.

This is especially useful with unencapsulated mode, where PJI 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

<i>flatten</i>	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

<i>resolution</i>	The ROP rendering resolution to use.
-------------------	--------------------------------------

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

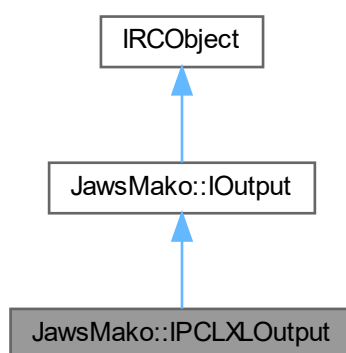
- [pclxlinput.h](#)

7.371 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](#) ¶m, 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 JAWMAKO_API IPCLXLOutputPtr `create` (const IJawsMakoPtr &jawsMako, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())
Create a PCL5 Output instance.

Static Public Member Functions inherited from `JawsMako::IOutput`

- static JAWMAKO_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.

7.371.1 Detailed Description

Interface for the PCLXL `IOutput` class.

7.371.2 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

<i>jawsMako</i>	The JawsMako instance.
<i>progressMonitor</i>	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 PJJ header or end of job should be emitted.

The default is true.

Equivalent to calling setParameter() with the parameter name "EmitPJJ", 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:

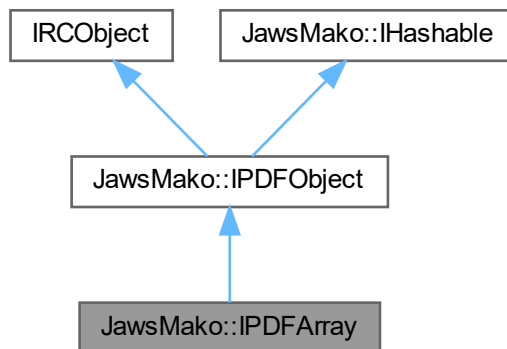
- [pclxloutput.h](#)

7.372 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 position 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 constituent 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, this 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 num←Numbers)
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.

7.372.1 Detailed Description

A simple class representing a mutable array of other PDF objects.

7.372.2 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

<i>value</i>	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

<i>object</i>	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 position in this array, replacing any currently stored object. Usually more efficient for bulk operations.

Parameters

<i>destIndex</i>	The destination index to store the object.
<i>sourceArray</i>	The array object from which to copy.
<i>sourceIndex</i>	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

<i>objects</i>	The objects to assign to the array.
<i>executable</i>	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

<i>size</i>	The initial size of the array. It will be populated with IPDFNull objects.
<i>executable</i>	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

<i>box</i>	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

<i>numbers</i>	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

<i>numbers</i>	The numbers to assign to the array.
<i>numNumbers</i>	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

<i>index</i>	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

<i>box</i>	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

<i>index</i>	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

<i>numbers</i>	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

<i>index</i>	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

<i>index</i>	The index at which to store the object.
<i>value</i>	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

<i>index</i>	The index at which to store the object.
<i>value</i>	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

<i>index</i>	The index of the object to remove.
--------------	------------------------------------

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

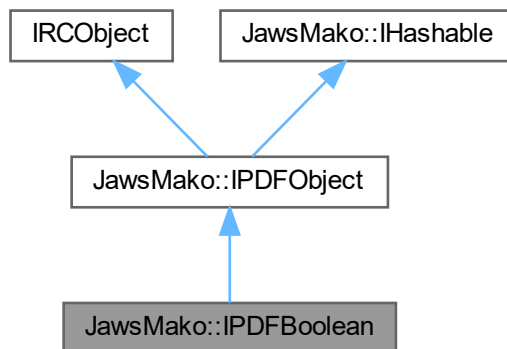
- [pdfobjects.h](#)

7.373 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 constituent 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, this 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.

7.373.1 Detailed Description

A simple immutable boolean PDF object type.

7.373.2 Member Function Documentation

`create()`

```
static JAWSMako_API IPDFBooleanPtr JawsMako::IPDFBoolean::create (
    bool boolean ) [static]
```

Create a boolean object.

Parameters

<i>boolean</i>	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:

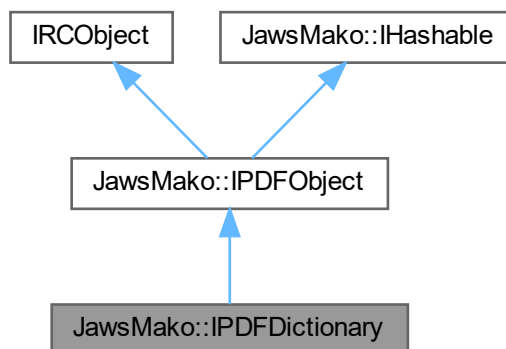
- [pdfobjects.h](#)

7.374 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 constituent 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, this 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.

7.374.1 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.

7.374.2 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

<i>object</i>	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

<i>sourceDict</i>	The dictionary from which to copy.
<i>sourceIndex</i>	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

<i>pFactory</i>	The JawsMako instance or factory to use to create the dictionary.
<i>size</i>	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 iterator 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

<i>key</i>	The null-terminated string to use as key.
<i>value</i>	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

<i>key</i>	The name object to use as key.
<i>value</i>	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

<i>key</i>	The raw string to use as key.
<i>value</i>	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

<i>key</i>	The name object to use as key.
<i>integer</i>	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

<i>key</i>	The name object to use as key.
<i>real</i>	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

<i>key</i>	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

<i>key</i>	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

<i>key</i>	The raw string to use as key.
------------	-------------------------------

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

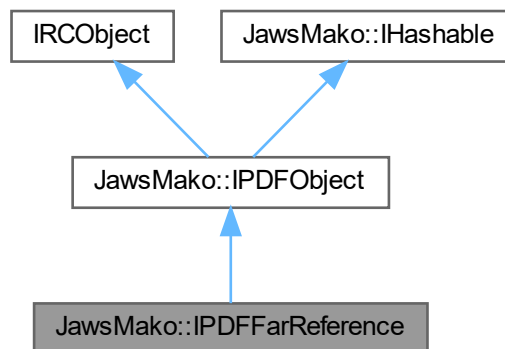
- [pdfobjects.h](#)

7.375 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 constituent 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, this 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.

7.375.1 Detailed Description

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

7.375.2 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

<i>reference</i>	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:

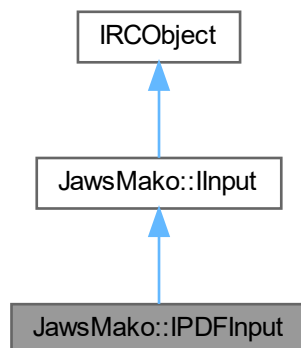
- [pdfobjects.h](#)

7.376 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 [IRAIInputStreamPtr](#) &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 [setImproveGlyphSelectionForUnembeddedIdentityFonts](#) (bool improve)=0
Set whether or not to attempt to improve glyph selection for non- embedded Identity CIDFonts.
- 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](#) &progressMonitor, 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.
- virtual void [setAlternateBaseFont](#) (const [U8String](#) &baseFontName, const [U8String](#) &alternateFontName)=0

Set an alternate base font for cases when the built-in font may not be suitable.

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](#) ¶m, 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 [IRCOject](#)

- virtual void [addRef](#) () const =0

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

Decreases the reference count of the actual object pointed to. When the reference count falls to 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.

7.376.1 Detailed Description

An instance of the JawsMako PDF input class.

7.376.2 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

<code>jawsMako</code>	The IJawsMako object
<code>progressMonitor</code>	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 CIRInputStreamVect 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

<code>pdfStream</code>	The input PDF stream.
------------------------	-----------------------

Returns

CIRInputStreamVect 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 CIRAIpStreamVect 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

<i>pathToFile</i>	The path to the input file.
-------------------	-----------------------------

Returns

CIRAIpStreamVect 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 CIRAIpStreamVect 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

<i>pathToFile</i>	The path to the input file.
-------------------	-----------------------------

Returns

CIRAIpStreamVect 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

<i>pdfStream</i>	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

<i>pathToFile</i>	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

<i>pathToFile</i>	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

<i>pdfStream</i>	The input PDF stream.
<i>incrementalUpdateIndex</i>	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

<i>pathToFile</i>	The path to the input file.
<i>incrementalUpdateIndex</i>	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

<i>pathToFile</i>	The path to the input file.
<i>incrementalUpdateIndex</i>	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

<i>pdfStream</i>	The input PDF stream to scan.
<i>progressMonitor</i>	A smart pointer to an IPProgressMonitor object which can be NULL if no such object was passed in.
<i>startPageIndex</i>	Optional index of the first page to scan. Pages are numbered starting with page 0, which is the default value.
<i>endPageIndex</i>	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 IPProgressMonitorPtr & 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

<i>pathToFile</i>	The path to the input file to scan.
<i>progressMonitor</i>	A smart pointer to an IPProgressMonitor object which can be NULL if no such object was passed in.
<i>startPageIndex</i>	Optional index of the first page to scan. Pages are numbered starting with page 0, which is the default value.
<i>endPageIndex</i>	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 IPProgressMonitorPtr & 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

<i>pathToFile</i>	The path to the input file to scan.
<i>progressMonitor</i>	A smart pointer to an IPProgressMonitor object which can be NULL if no such object was passed in.
<i>startPageIndex</i>	Optional index of the first page to scan. Pages are numbered starting with page 0, which is the default value.
<i>endPageIndex</i>	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.

setAlternateBaseFont()

```
virtual void JawsMako::IPDFInput::setAlternateBaseFont (
    const U8String & baseFontName,
    const U8String & alternateFontName ) [pure virtual]
```

Set an alternate base font for cases when the built-in font may not be suitable.

This function can be used to set the name of a system font that may be used in place of one of the base PDF fonts. Some poorly constructed PDF files attempt to define encodings for non-embedded base fonts (e.g. Helvetica) with character names that are not present in the Adobe standard Latin character set, and are not available when using Mako's base fonts, but may be present in a system font.

Setting an alternate font name indicates that Mako should substitute a non-embedded Base 35 font with a system font when the PDF encoding defines character names outside of the Adobe standard Latin character set.

Parameters

<i>baseFontName</i>	The name of a Base 35 font to substitute. Must be a non-symbolic font.
<i>alternateFontName</i>	The name of a system font to use as a substitute, which must be the name of a system font that can be found with JawsMako::findFont() .

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.

setImproveGlyphSelectionForUnembeddedIdentityFonts()

```
virtual void JawsMako::IPDFInput::setImproveGlyphSelectionForUnembeddedIdentityFonts (
    bool improve ) [pure virtual]
```

Set whether or not to attempt to improve glyph selection for non- embedded Identity CIDFonts.

Such fonts are encoded such that glyphs are directly selected by mapping character codes to glyph IDs. Jobs that reference such fonts will only work correctly if the font that is used stores the used characters using the same glyph IDs. This is true if the exact same font is present on the users system as the system where the job was generated. It may work for different versions of the same font, or cases where by convention the same glyphs slots are used.

There are no guarantees that this will be the case in general.

To attempt to improve the situation, Mako will attempt to use other clues to better select glyphs, if this feature is enabled and can oftentimes produce a legible result where disparate fonts are used.

However doing this can disrupt cases where the intent is to write the result to PDF output without the fonts embedded. Here it is necessary to use the same glyph slots regardless of what is stored in them. If IPDFOutput::setEmbedIdentityCidFonts(false) is to be used, then this feature should be disabled also.

The default is true.

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

<i>password</i>	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

<i>pkcs12</i>	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:

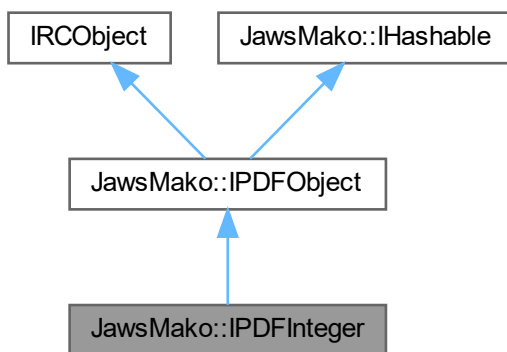
- [pdfinput.h](#)

7.377 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 constituent 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, this 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.

7.377.1 Detailed Description

A simple immutable integer PDF object type.

7.377.2 Member Function Documentation

create()

```
static JAWSMAKO_API IPDFIntegerPtr JawsMako::IPDFInteger::create (  
    int32 integer ) [static]
```

Create an integer object.

Parameters

<i>integer</i>	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:

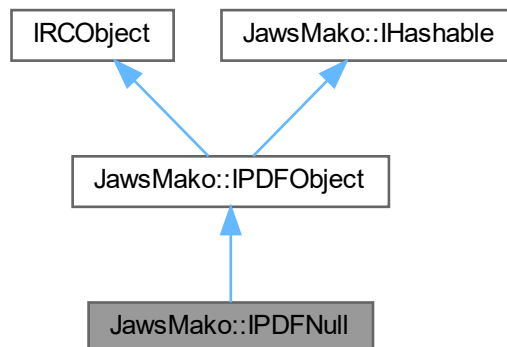
- [pdfobjects.h](#)

7.378 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 constituent 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, this 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.

7.378.1 Detailed Description

A simple immutable "null" pdf object.

7.378.2 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:

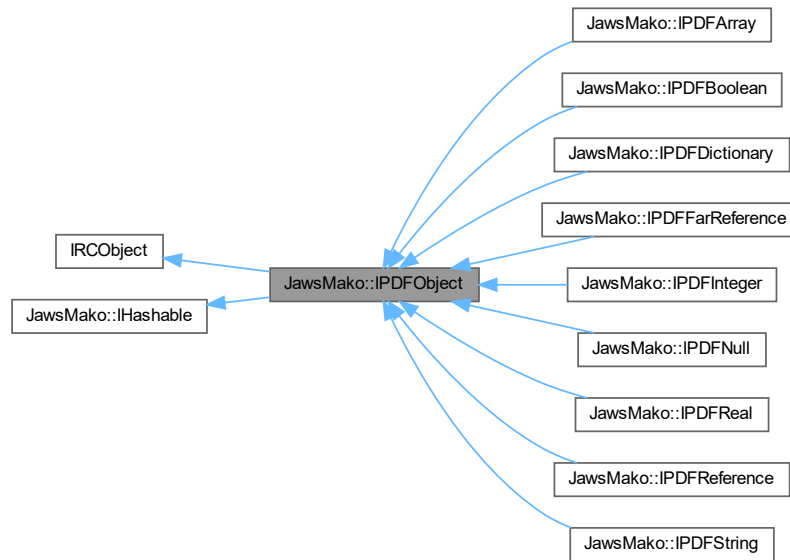
- [pdfobjects.h](#)

7.379 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 constituent 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, this 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.

7.379.1 Detailed Description

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

7.379.2 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

<i>number</i>	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 constituent 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

<i>number</i>	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, this will fail if the integer cannot be represented exactly in a single precision float.

Parameters

<i>number</i>	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

<i>number</i>	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

<i>string</i>	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:

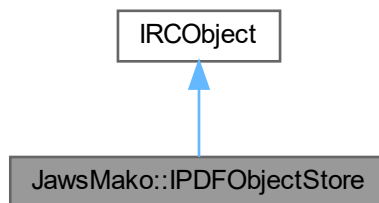
- [pdfobjects.h](#)

7.380 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.

7.380.1 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.

7.380.2 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, dictionaries, 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

<i>object</i>	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

<i>reference</i>	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

<i>reference</i>	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

<i>object</i>	The object to store
<i>reference</i>	The reference under which the object is stored. Must have previously been allocated using newReference() above.
<i>markDirty</i>	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

<i>object</i>	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:

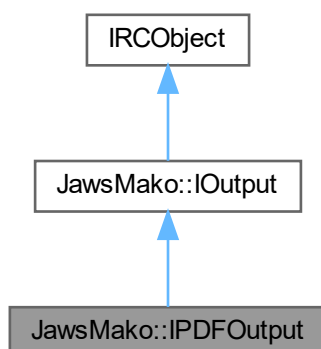
- [pdfobjects.h](#)

7.381 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 [setEnabledIncrementalOutput](#) (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 color 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 `setEmbedIdentityCidFonts` (bool embed)=0

Set whether or not Identity CIDFonts should be embedded.
- 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](#) ¶m, 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 `IRCOject`

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

- virtual `~IRCOject` ()
Virtual destructor.

7.381.1 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 color 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

7.381.2 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.
ePDNHFailOnMismatchedDeviceNProcessSpace	Throw an exception if a DeviceN Process Property does.
ePDNHFailOnNonUtf8ColorantName	Throw an exception if a DeviceN/Separation Colorant name is not valid UTF-8. Normal behavior 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.

7.381.3 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

<i>jawsMako</i>	The JawsMako instance.
<i>progressMonitor</i>	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

<i>allow</i>	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 color 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

<i>resolution</i>	The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default.
<i>threshold</i>	The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution.
<i>method</i>	The method to use when downsampling. The default is bicubic for color 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 color 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 color 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 color 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

<i>characterSet</i>	The default CJK character set for forms
---------------------	---

setDeviceNErrrorHandling()

```
virtual void JawsMako::IPDFOutput::setDeviceNErrrorHandling (
    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").

setEmbedIdentityCidFonts()

```
virtual void JawsMako::IPDFOutput::setEmbedIdentityCidFonts (
    bool embed ) [pure virtual]
```

Set whether or not Identity CIDFonts should be embedded.

If true, CIDFonts that would need to be represented in the PDF as Identity CID fonts are always embedded unless the font is restricted and [setAllowRestrictedFonts\(\)](#) has been set to false.

The reason for this is that it is generally unsafe to not embed such fonts. These fonts select a glyph in a font directly by its glyph ID, and as such, if the font that is substituted by a consuming application is not the same, then the correct glyphs may not be selected; the result will be whatever glyphs the substitute font happens to have present in the used glyph slots.

It is for this reason that the default is true.

If false, the other font embedding settings will apply to fonts of this type. For safety, if using PDF input, `IPDFInput::setImproveGlyphSelectionForUnembeddedIdentityFonts(false)` should be used to preserve the original glyph indexes from input cases.

Equivalent to calling [setParameter\(\)](#) with the parameter name "EmbedIdentityCidFonts" and the value "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".

setEnabledIncrementalOutput()

```
virtual void JawsMako::IPDFOutput::setEnabledIncrementalOutput (
    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.

setEnabledTrueTypeNotDef()

```
virtual void JawsMako::IPDFOutput::setEnabledTrueTypeNotDef (
    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, eExtractionAllowed, 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

<i>keyLength</i>	The key length to use. Must be 0, 40, 128 or 256. 0 indicates that encryption will not be performed.
<i>permissions</i>	The permissions bit mask for the output PDF. (see IDOMStandardPDFSecurityInfo::ePermissionsFlags).
<i>encryptMetadata</i>	Whether or not to encrypt metadata. The default is false. Must be set to true if a 40 bit key length is specified.
<i>ownerPassword</i>	The owner password to use.
<i>userPassword</i>	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

<i>resolution</i>	The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default.
<i>threshold</i>	The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution.
<i>method</i>	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

<i>subsample</i>	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

<i>quality</i>	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

<i>resolution</i>	The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default.
<i>threshold</i>	The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution.
<i>method</i>	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 [setDeviceNErrrorHandling](#). See the two-parameter method of [inkNameToString](#) for how this is done.

Equivalent to calling [setParameter\(\)](#) with the parameter name "NonUtf8InkNameFallbackEncoding" and the [eInkFallbackEncoding](#) enumeration name, with the "eIFE" prefix removed. For example: `setParameter("NonUtf8InkNameFallbackEncoding", "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 color 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

<i>compression</i>	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

<i>compression</i>	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

<i>compression</i>	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

<i>compression</i>	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

<i>keyLength</i>	The key length to use. Must be 0, 128 or 256. 0 indicates that encryption will not be performed.
<i>recipients</i>	A vector of CPDFRecipients objects for all users who will be allowed to access the resulting document. Pass an empty array to disable encryption.
<i>encryptMetadata</i>	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

<i>resolution</i>	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

<i>retainClippedContent</i>	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

<i>retainEmbeddedFiles</i>	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 color 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 color 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

<i>targetSpace</i>	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

<i>profile</i>	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

<i>version</i>	Version string
----------------	----------------

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

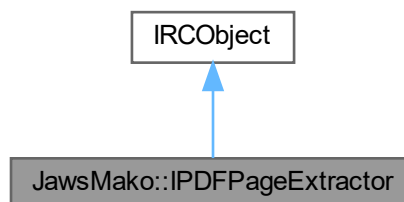
- [pdfoutput.h](#)

7.382 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 IInputStreamPtr &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.

7.382.1 Detailed Description

An instance of the JawsMako PDFPageExtractor class.

7.382.2 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

<i>jawsMako</i>	The JawsMako object.
<i>inputStream</i>	The input PDF stream.
<i>password</i>	The password to use. Assumed to be UTF-8.
<i>permissions</i>	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

<i>progressMonitor</i>	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

<i>outputStream</i>	The output stream.
<i>pageIndex</i>	The index of the first page to extract.
<i>numPages</i>	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

<i>outputStream</i>	The output stream.
<i>pageIndex</i>	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

<i>outputStream</i>	The output stream.
<i>pageIndex</i>	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:

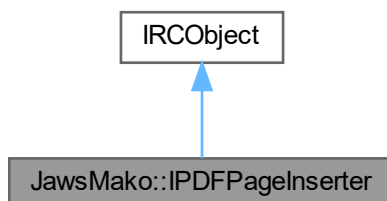
- [pdfpage.h](#)

7.383 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 IInputStreamPtr &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.

7.383.1 Detailed Description

An instance of the JawsMako PDFPageInserter class.

7.383.2 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

<i>jawsMako</i>	The IJawsMako object.
<i>inputStream</i>	The input PDF stream.
<i>password</i>	The password to use. Assumed to be UTF-8.
<i>permissions</i>	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

<i>progressMonitor</i>	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

<i>inputStream</i>	The input stream.
<i>destPageIndex</i>	The page index of the destination to insert the pages into.
<i>sourcePageIndex</i>	The page index of the source to insert the pages from.
<i>numPages</i>	The number of pages to insert.
<i>password</i>	The password to use. Assumed to be UTF-8.
<i>permissions</i>	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

<i>inputStream</i>	The input stream.
<i>destPageIndex</i>	The page index of the destination to insert the pages into.
<i>sourcePageIndex</i>	The page index of the source to insert the pages from.
<i>password</i>	The password to use. Assumed to be UTF-8.
<i>permissions</i>	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

<i>inputStream</i>	The input stream.
<i>destPageIndex</i>	The page index of the destination to insert the pages into.
<i>sourcePageIndex</i>	The final page index of the source to insert the pages from.
<i>password</i>	The password to use. Assumed to be UTF-8.
<i>permissions</i>	The allowed permissions.

save()

```
virtual void JawsMako::IPDFPageInserter::save (
    const IOutputStreamPtr & outputStream ) [pure virtual]
```

Save into a given output stream.

Parameters

<i>outputStream</i>	The output stream.
---------------------	--------------------

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

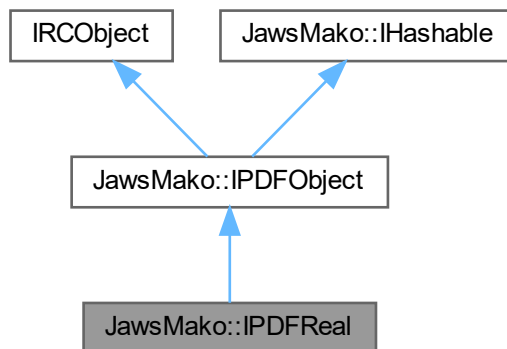
- [pdfpage.h](#)

7.384 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 constituent 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, this 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.

7.384.1 Detailed Description

A simple immutable floating-point PDF object type.

7.384.2 Member Function Documentation

[create\(\)](#)

```
static JAWSMako_API IPDFRealPtr JawsMako::IPDFReal::create (
    double real ) [static]
```

Create a real object.

Parameters

<i>real</i>	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:

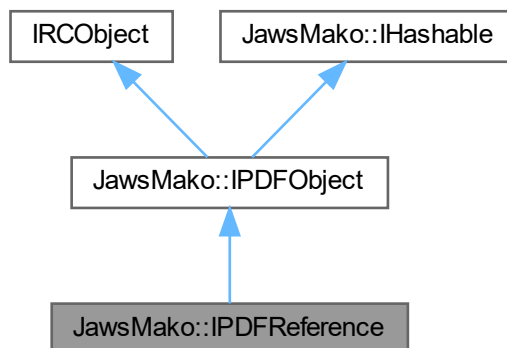
- [pdfobjects.h](#)

7.385 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 generation 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 constituent 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, this 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.

7.385.1 Detailed Description

A simple class representing an immutable PDF indirect reference.

7.385.2 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

<i>reference</i>	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

<i>objectNum</i>	The object number to use.
<i>generation</i>	The generation number to use.

Returns

IPDFReferencePtr The created object.

getGeneration()

```
int32 JawsMako::IPDFReference::getGeneration ( ) [inline]
```

Obtain the generation 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:

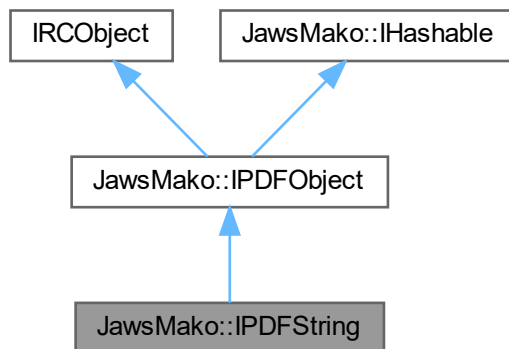
- [pdfobjects.h](#)

7.386 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 constituent 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, this 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.

7.386.1 Detailed Description

A simple immutable string pdf object containing raw unencoded data or PDF Text information.

A simple immutable name pdf object.

7.386.2 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

<i>stream</i>	The stream to use as source data.
<i>executable</i>	If this string should be marked executable.
<i>pdf2</i>	True if this string was created from a PDF 2.0 or later document.
<i>startOffset</i>	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

<i>string</i>	The raw string value.
<i>executable</i>	If this string should be marked executable.
<i>pdf2</i>	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

<i>string</i>	The string value.
<i>executable</i>	If this string should be marked executable.
<i>pdf2</i>	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:

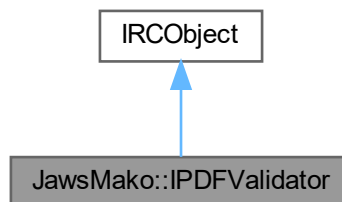
- [pdfobjects.h](#)

7.387 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.

7.387.1 Detailed Description

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

7.387.2 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

<i>pdfStream</i>	The PDF stream.
<i>globalErrors</i>	A reference to receive the list of global errors (that is, errors not attached to a particular page).
<i>pageErrors</i>	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:

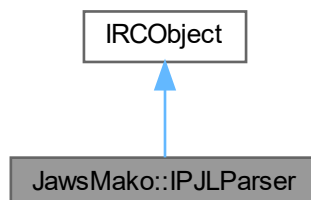
- validate.h

7.388 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](#) {
[ePREnterPclXI](#) , [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 SHORTEGE>".
- 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.

7.388.1 Detailed Description

A PjL (Printer Job Language) parser for Mako.

7.388.2 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

ePREnterPclXI	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.

7.388.3 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

<i>stream</i>	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:

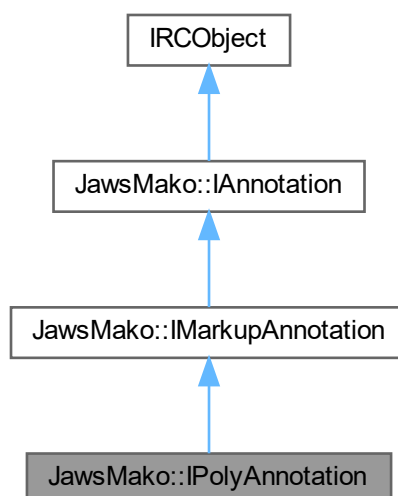
- pjl.h

7.389 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 [IRCOject](#)

- virtual void [addRef](#) () const =0
- Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.*
- virtual bool [decRef](#) () const =0
- Decreases the reference count of the actual object pointed to. When the reference count falls to 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.

7.389.1 Detailed Description

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

7.389.2 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

<i>points</i>	The vector of points
---------------	----------------------

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

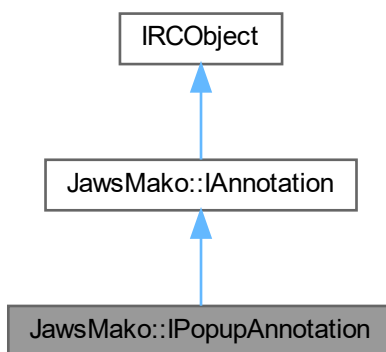
- [interactive.h](#)

7.390 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.

7.390.1 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.

7.390.2 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

<i>jawsMako</i>	The JawsMako instance
<i>rect</i>	The annotations bounds. Must not be empty.
<i>open</i>	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

<i>open</i>	Set to true for open, false for closed
-------------	--

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

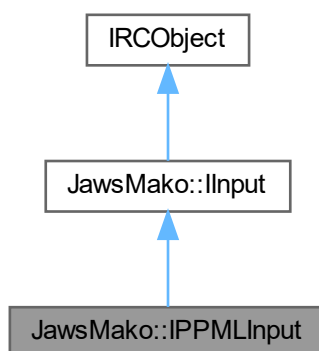
- [interactive.h](#)

7.391 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](#) ¶m, 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.

7.391.1 Detailed Description

An instance of the JawsMako PPML input class.

7.391.2 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

<i>jawsMako</i>	The IJawsMako object
<i>progressMonitor</i>	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:

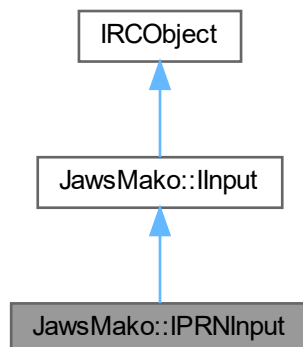
- [ppmlinput.h](#)

7.392 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](#) ¶m, 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.

7.392.1 Detailed Description

An instance of the JawsMako PRN input class.

7.392.2 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

<i>jawsMako</i>	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:

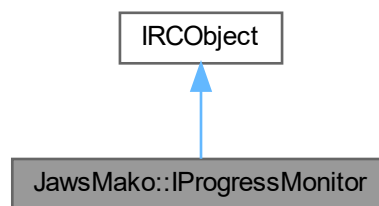
- [prninput.h](#)

7.393 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.

7.393.1 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.

7.393.2 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	<i>abort</i>	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	<i>progressTick</i>	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	<i>progressTick</i>	A smart pointer to an IProgressTick object which can be NULL if no object was passed in.
in	<i>abort</i>	A smart pointer to an IAbort object which can be NULL if 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:

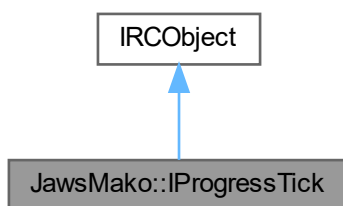
- [jawsmake.h](#)

7.394 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.

7.394.1 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\(\)](#).

7.394.2 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	<i>cbFunc</i>	A floating point callback function that will be called on each tick.
in	<i>cbPriv</i>	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	<i>cbFunc</i>	An integer callback function that will be called on each tick.
in	<i>cbPriv</i>	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	<i>cbFunc</i>	The integer type callback function to be invoked
in	<i>cbPriv</i>	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	<i>cbFunc</i>	The integer type callback function to be invoked
in	<i>cbPriv</i>	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	<i>tickMax</i>	The maximum tick counter value.
----	----------------	---------------------------------

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

- [jawsmako.h](#)

7.395 JawsMako::IPSCCommentMonitor 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.

7.395.1 Detailed Description

Interface allowing users to monitor comments in PostScript input. Use `setCommentMonitor()` to install subclasses of this type.

7.395.2 Member Function Documentation**comment()**

```
virtual void JawsMako::IPSCCommentMonitor::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

<i>comment</i>	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](#)

7.396 JawsMako::IPSIjector Class Reference

Interface allowing users of [IPSOOutput](#) to inject raw PostScript directly into the output stream at strategic points in the output process. Use [IPSOOutput::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.

7.396.1 Detailed Description

Interface allowing users of [IPSOOutput](#) to inject raw PostScript directly into the output stream at strategic points in the output process. Use [IPSOOutput::setInjector\(\)](#) to install subclasses of this type.

7.396.2 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

<i>pageNumber</i>	The page number being prepared, with 0 indicating the first page.
<i>psStream</i>	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

<i>psStream</i>	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

<i>psStream</i>	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

<i>pageNumber</i>	The page number being prepared, with 0 indicating the first page.
<i>psStream</i>	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

<i>psStream</i>	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

<i>psStream</i>	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

<i>psStream</i>	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

<i>pageNumber</i>	The page number being output, with 0 indicating the first page.
<i>psStream</i>	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:

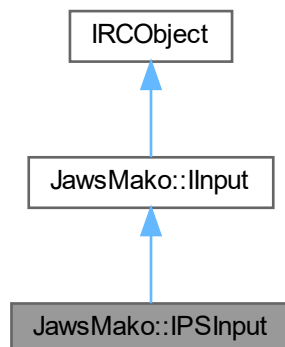
- psoutput.h

7.397 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](#) ¶m, 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, 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.

7.397.1 Detailed Description

An instance of the JawsMako PS input class.

7.397.2 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

<i>jawsMako</i>	The IJawsMako object
<i>eps</i>	Whether or not the source is Encapsulated PostScript. The default is false.
<i>progressMonitor</i>	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

<i>prolog</i>	The prolog stream to use.
---------------	---------------------------

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

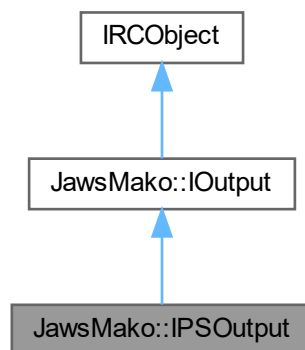
- [psinput.h](#)

7.398 JawsMako::IPSOOutput Class Reference

Interface for the PS [IOutput](#) class.

```
#include <psoutput.h>
```

Inheritance diagram for JawsMako::IPSOOutput:



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 color 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 be represented in PostScript will be forcibly converted to the target color 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** (IPSIInjector *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 **setPreferredMonolImageCompression** (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 color 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 **setMonolImageMaxResolution** (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 **setAllowReusableImageStreams** (bool allowReusableImageStreams)=0
Sets whether to allow reusable streams when writing images.

Public Member Functions inherited from JawsMako::Output

- 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 ¶m, 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.

7.398.1 Detailed Description

Interface for the PS [IOuput](#) 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

7.398.2 Member Function Documentation

create()

```
static JAWSMako_API IPSOutputPtr JawsMako::IPSOuput::create (
    const IJawsMakoPtr & jawsMako,
    const IProgressMonitorPtr & progressMonitor = IProgressMonitorPtr() ) [static]
```

Create a PS Output instance.

Parameters

<i>jawsMako</i>	The JawsMako instance.
<i>progressMonitor</i>	The progress monitor which allows aborting an operation or registering a progress callback.

setAllowReusableImageStreams()

```
virtual void JawsMako::IPSOuput::setAllowReusableImageStreams (
    bool allowReusableImageStreams ) [pure virtual]
```

Sets whether to allow reusable streams when writing images.

When set to true, a single reusable stream (PostScript ReusableStreamDecode filter) will be used as the data source for any images that are considered to be reusable.

This can produce significantly smaller output for content with lots of image reuse, but will increase the printer memory that is used.

Currently, only color images will be considered for reuse.

The default is false.

setColorImageMaxResolution()

```
virtual void JawsMako::IPSOuput::setColorImageMaxResolution (
    float resolution,
```



```

float threshold = 0.0f,
IDOMImageDownsamplerFilter::eDownsamplingMethod method = IDOMImageDownsamplerFilter↔
::eBicubic ) [pure virtual]

```

Set the desired maximum resolution, threshold and downsampling method for color 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

<i>resolution</i>	The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default.
<i>threshold</i>	The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution.
<i>method</i>	The method to use when downsampling. The default is bicubic for color images.

setConvertAllObjectsToTargetColorSpace()

```

virtual void JawsMako::IPSOOutput::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 color space. Exquivalent to calling [setParameter\(\)](#) with the param name "ConvertAllObjectsToTargetColorSpace".

The default is true.

setGrayImageMaxResolution()

```

virtual void JawsMako::IPSOOutput::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

<i>resolution</i>	The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default.
<i>threshold</i>	The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution.
<i>method</i>	The method to use when downsampling. The default is bicubic for gray images.

setJPEGQuality()

```
virtual void JawsMako::IPSOuput::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

<i>quality</i>	The desired quality level, with 1 being lowest quality and 5 being highest quality.
----------------	---

setMonoImageMaxResolution()

```
virtual void JawsMako::IPSOuput::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

<i>resolution</i>	The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default.
<i>threshold</i>	The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution.
<i>method</i>	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::IPSOuput::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 color 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

<i>compression</i>	The desired compression.
--------------------	--------------------------

setPreferredGrayImageCompression()

```
virtual void JawsMako::IPSOOutput::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

<i>compression</i>	The desired compression.
--------------------	--------------------------

setPreferredMonoImageCompression()

```
virtual void JawsMako::IPSOOutput::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

<i>compression</i>	The desired compression.
--------------------	--------------------------

setStreamingOutput()

```
virtual void JawsMako::IPSOOutput::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::IPSOOutput::setTargetColorSpace (
    const IDOMColorSpacePtr & targetSpace ) [pure virtual]
```

Set the target color 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).

Parameters

<i>targetSpace</i>	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::IPSOOutput::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

<i>profile</i>	The desired profile.
----------------	----------------------

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

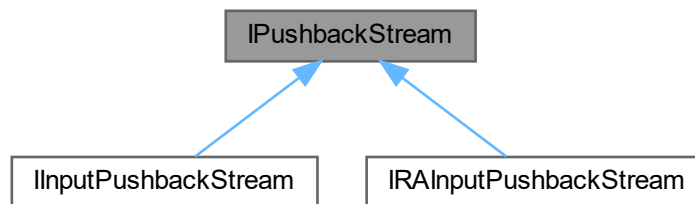
- psoutput.h

7.399 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.

7.399.1 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()`.

7.399.2 Member Function Documentation

`pushBack()` [1/2]

```
virtual bool IPushbackStream::pushBack (
    const void * buffer,
    int32 count ) [pure virtual]
```

Push back from a buffer.

Parameters

<i>buffer</i>	Buffer to push back from.
<i>count</i>	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

<i>byte</i>	Byte to be pushed back.
-------------	-------------------------

Returns

bool True if method succeeded.

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

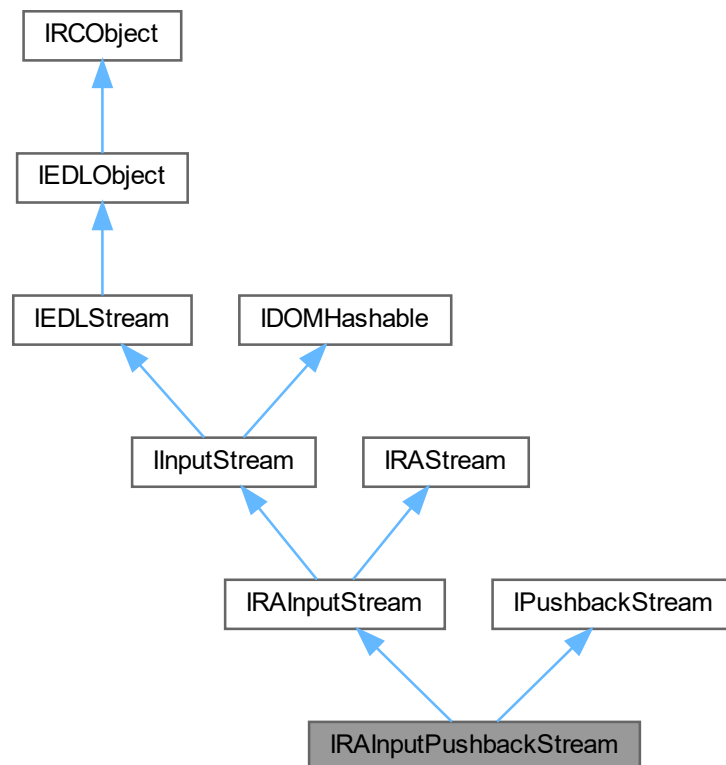
- edlstream.h

7.400 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 EDLSysString &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, UserRARReadFunc 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 IInput↔StreamPtr &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 CInput↔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.

7.400.1 Detailed Description

Random-access Input Stream with pushback support.

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

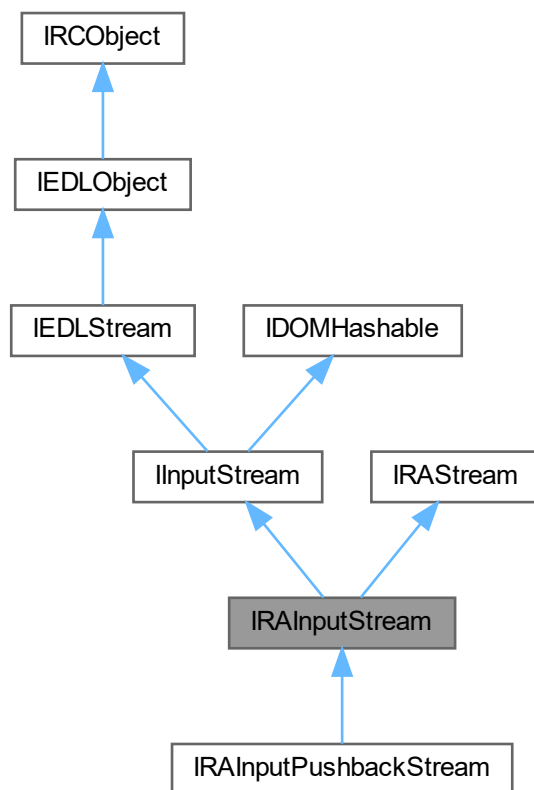
- `edlstream.h`

7.401 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 [IRCOject](#)

- virtual void [addRef](#) () const =0
Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool [decRef](#) () const =0
Decreases the reference count of the actual object pointed to. When the reference count falls to 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 [IRASStream](#)

- virtual `~IRASStream ()`
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 EDLSysString &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, UserStreamReadFunc 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, UserRARReadFunc 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 *pFactory, 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 CInput←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.

7.401.1 Detailed Description

Random Access Input Stream.

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

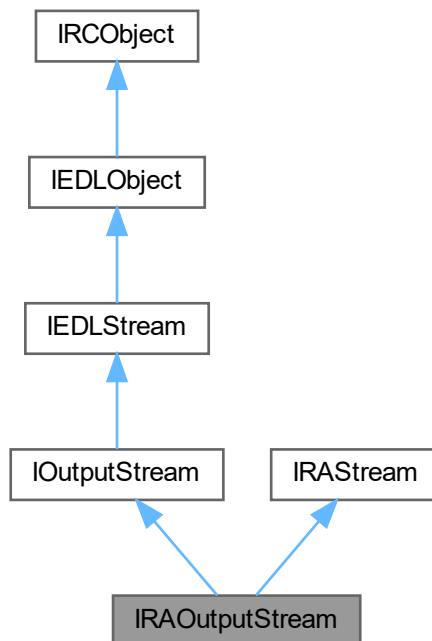
- edlstream.h

7.402 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 IRASStream

- virtual ~**IRASStream** ()
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 IOOutputStreamPtr [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 IOOutputStreamPtr [createToFlateCompressed](#) (IEDLClassFactory *pFactory, const IOOutput↔ 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 IOOutputStreamPtr [createToLz4Compressed](#) (IEDLClassFactory *pFactory, const IOOutput↔ 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 IOOutputStreamPtr &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 IOOutputStreamPtr &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.

7.402.1 Detailed Description

Random Access Output Stream.

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

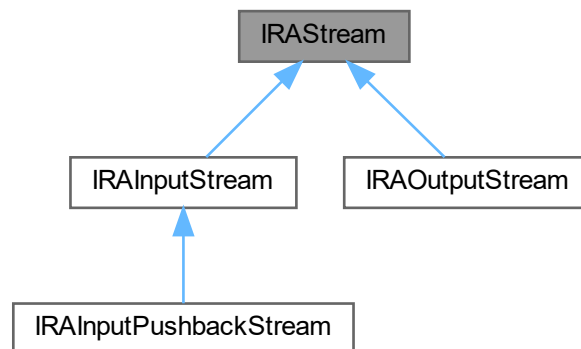
- edlstream.h

7.403 IRASStream 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.

7.403.1 Detailed Description

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

7.403.2 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

<i>newPos</i>	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

<i>newPos</i>	The desired new position.
---------------	---------------------------

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

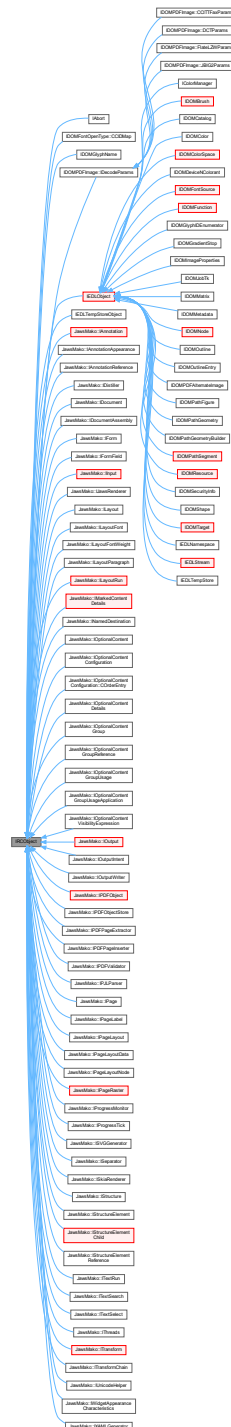
- edlstream.h

7.404 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.

7.404.1 Detailed Description

Base class Interface for all Reference Counted objects.

7.404.2 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:

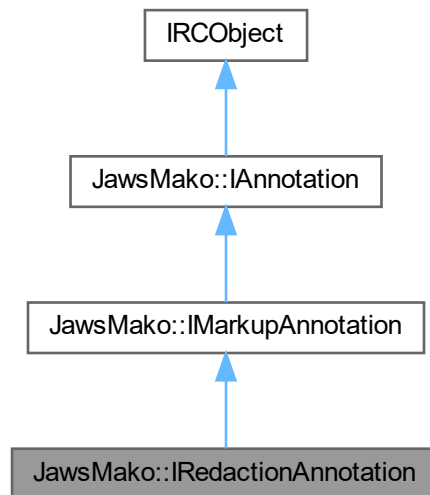
- [ircobject.h](#)

7.405 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.

7.405.1 Detailed Description

A generic interface class for a redaction annotation.

7.405.2 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

<i>jawsMako</i>	The JawsMako instance
<i>rect</i>	The annotations bounds. Must not be empty.
<i>quadPoints</i>	The quadrilaterals defining the redaction area. May be empty, in which case the annotation rect is used.
<i>color</i>	The outline color for the annotation when viewed in the user interface. Pass NULL for the default black
<i>interiorColor</i>	The color used to fill the annotation when it is applied. An empty color 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

<i>color</i>	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

<i>quadPoints</i>	The annotation's desired quad points
-------------------	--------------------------------------

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

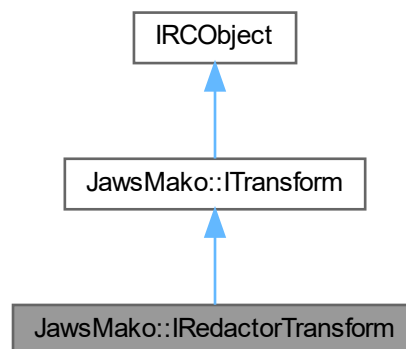
- [interactive.h](#)

7.406 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 [IRCOObject](#)

- virtual void `addRef` () const =0

Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.

- virtual bool `decRef` () const =0

Decreases the reference count of the actual object pointed to. When the reference count falls to 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 [IRCOObject](#)

- virtual `~IRCOObject` ()

Virtual destructor.

7.406.1 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.

7.406.2 Member Function Documentation

create()

```
static JAWSMako_API IRedactorTransformPtr JawsMako::IRedactorTransform::create (
    const IJawsMakoPtr & jawsMako,
    const IAbortPtr & abort = IAbortPtr() ) [static]
```

Create the transform.

Parameters

<i>jawsMako</i>	The JawsMako instance.
<i>abort</i>	An abort callback handler.

Returns

The new instance.

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

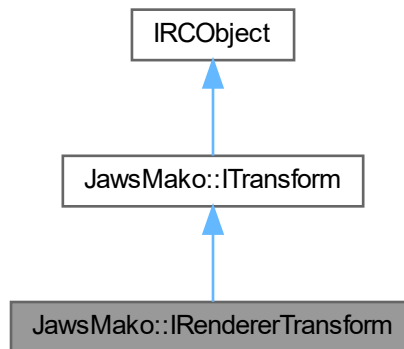
- [transforms.h](#)

7.407 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 flattening 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 and 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 halftone 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 rendered. 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 rendered. 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.

7.407.1 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.

7.407.2 Member Function Documentation

create()

```
static JAWSMAKO_API IRendererTransformPtr JawsMako::IRendererTransform::create (
    const IJawsMakoPtr & jawsMako,
    const IAbortPtr & abort = IAbortPtr() ) [static]
```

Create the transform.

Parameters

<i>jawsMako</i>	The JawsMako instance.
<i>abort</i>	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

<i>jawsMako</i>	The Mako instance to use.
<i>inkInfo</i>	The ink info to convert.
<i>processSpace</i>	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

<i>preserveForText</i>	Whether or not to apply black preservation for text.
<i>preserveForOther</i>	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 and 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 generate masked images as the rendered 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 targeted 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 attempt 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 renderable 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 halftone 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

<i>width</i>	The width of the halftone cell, in pixels
<i>height</i>	The height of the halftone cell, in pixels
<i>thresholds</i>	The threshold array. Must be <code>width * height</code> 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 flattening 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:

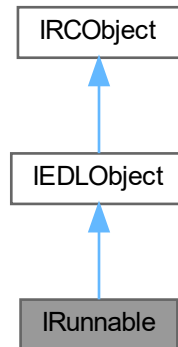
- [transforms.h](#)

7.408 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.

7.408.1 Detailed Description

Interface to filter's runnable classes.

7.408.2 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:

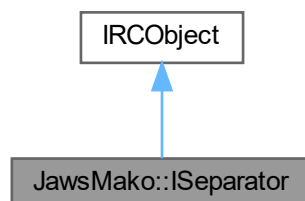
- `isession.h`

7.409 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 colors 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.

7.409.1 Detailed Description

An instance of the JawsMako Separator class.

7.409.2 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

<i>jawsMako</i>	The JawsMako instance.
<i>transformSpots</i>	Whether spot colors should be transformed to the target color space. The default is false.
<i>progressMonitor</i>	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

<i>index</i>	The index of the separated fixed page to retrieve. Will throw an exception if there is no separated page at the given index.
<i>content</i>	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

<i>index</i>	The index of the separated fixed page to retrieve. Will throw an exception if there is no separated page at the given index.
<i>content</i>	The separated fixed page.
<i>name</i>	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

<i>content</i>	The input fixed page to separate. Use getSeparation() to get the separated pages.
----------------	---

setEnabledVectorMode()

```
virtual void JawsMako::ISeparator::setEnabledVectorMode (
    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 renderable 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 [setEnabledVectorMode\(\)](#) is false.

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

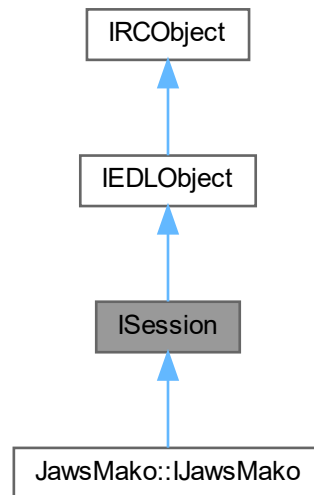
- [separator.h](#)

7.410 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 meaningful for executable environments. Note the client must set this during initialisation 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.

7.410.1 Detailed Description

EDL session class.

7.410.2 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

<i>sStartupDirectory</i>	The path of the directory that the executable 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

<i>sTempDirectory</i>	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

<i>pFactory</i>	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 meaningful for executable environments. Note the client must set this during initialisation for it to be valid.

Parameters

<i>sStartupDirectory</i>	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

<i>sTempDirectory</i>	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:

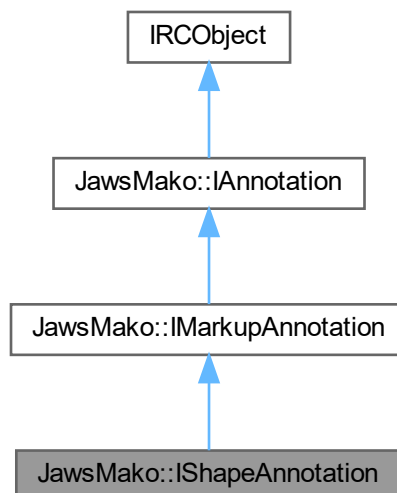
- isession.h

7.411 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 **IRCOject**

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

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

Decreases the reference count of the actual object pointed to. When the reference count falls to 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 [IRCOobject](#)

- virtual [~IRCOobject](#) ()

Virtual destructor.

7.411.1 Detailed Description

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

7.411.2 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

<i>jawsMako</i>	The JawsMako instance
<i>rect</i>	The annotations bounds. Must not be empty.
<i>type</i>	The type of annotation to create. Must be IAnnotation::eATSquare or IAnnotation::eATCircle
<i>color</i>	Optional; The desired annotation color. If NULL, no line will be visible.
<i>border</i>	Optional; The border to use. If not provided, a default will be used.
<i>interior</i>	Optional; the fill color to use. If NULL, no fill will be visible.
<i>inset</i>	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

<i>color</i>	The color definition, or pass NULL to remove the color
--------------	--

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

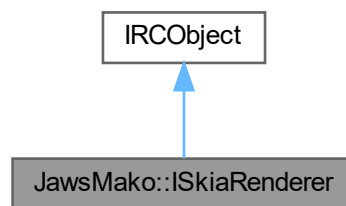
- [interactive.h](#)

7.412 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.

7.412.1 Detailed Description

A renderer that can paint XPS compatible DOM into a Skia canvas using the Skia API.

7.412.2 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:

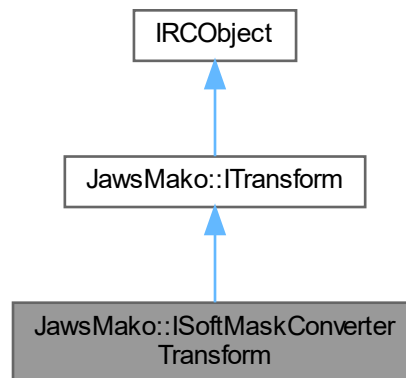
- `skiarenderer.h`

7.413 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 transformAnnotations=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 [IRCOject](#)

- virtual void **addRef** () const =0
Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool **decRef** () const =0
Decreases the reference count of the actual object pointed to. When the reference count falls to 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 [IRCOject](#)

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

7.413.1 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:

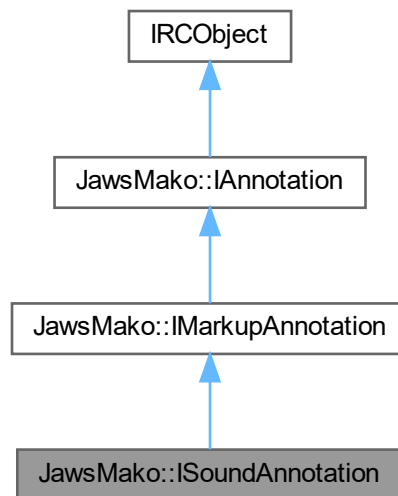
- [transforms.h](#)

7.414 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 `InputStreamPtr` [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.

7.414.1 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.

7.414.2 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:

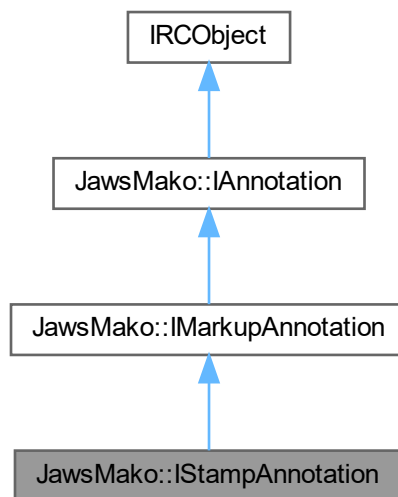
- [interactive.h](#)

7.415 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 **IRCOject**

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

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

Decreases the reference count of the actual object pointed to. When the reference count falls to 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.

7.415.1 Detailed Description

A generic interface class for a stamp annotation.

7.415.2 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

<code>jawsMako</code>	The JawsMako instance
<code>rect</code>	The annotations bounds. Must not be empty
<code>name</code>	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

<i>intent</i>	The intent
---------------	------------

setName()

```
virtual void JawsMako::IStampAnnotation::setName (
    const U8String & name ) [pure virtual]
```

Set the stamp annotation's name.

Parameters

<i>name</i>	The desired name
-------------	------------------

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

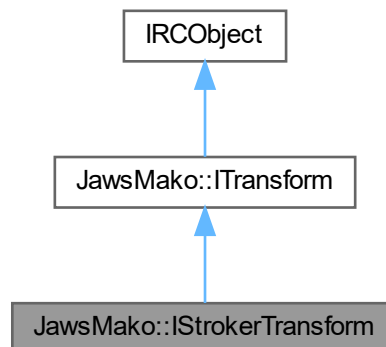
- [interactive.h](#)

7.416 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 consumer.
- 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** (IDOMPPathNode::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** (IDOMPPathNode::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** (IDOMPPathNode::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 [setConvertAllStrokes\(\)](#) 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.

7.416.1 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.

7.416.2 Member Function Documentation

create()

```
static JAWSMako_API IStrokerTransformPtr JawsMako::IStrokerTransform::create (
    const IJawsMakoPtr & jawsMako,
    const IAbortPtr & abort = IAbortPtr() ) [static]
```

Create the transform.

Parameters

<i>jawsMako</i>	The JawsMako instance.
<i>abort</i>	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 consumer.

```
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:

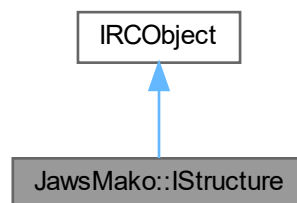
- [transforms.h](#)

7.417 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.

7.417.1 Detailed Description

Top level tracking structure describing the logical structure of the document.

7.417.2 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 its 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:

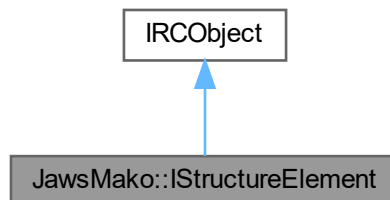
- [structure.h](#)

7.418 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 targeting the given form to the end of the children. 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 targeting 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.

7.418.1 Detailed Description

A structure element in the structure tree.

7.418.2 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

<i>attributes</i>	Either a Dictionary, Array or NULL to clear the attributes.
-------------------	---

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

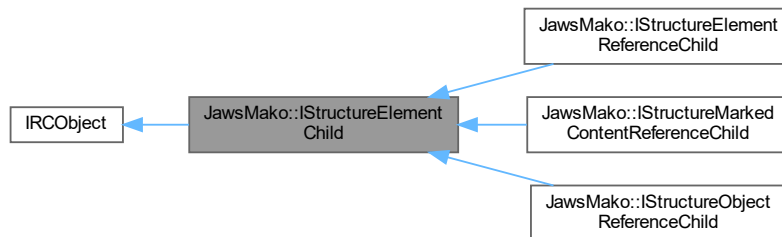
- [structure.h](#)

7.419 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.

7.419.1 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:

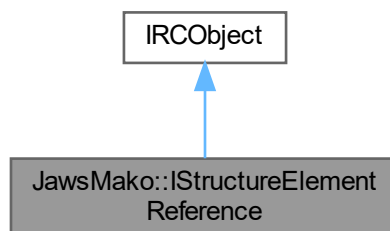
- [structure.h](#)

7.420 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.

7.420.1 Detailed Description

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

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

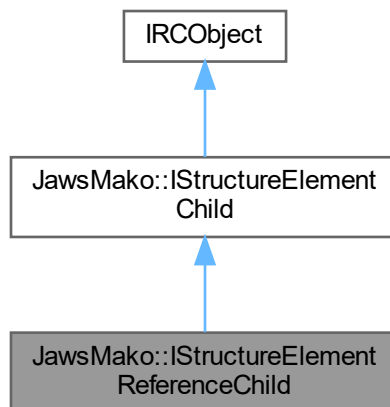
- [structure.h](#)

7.421 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.

7.421.1 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:

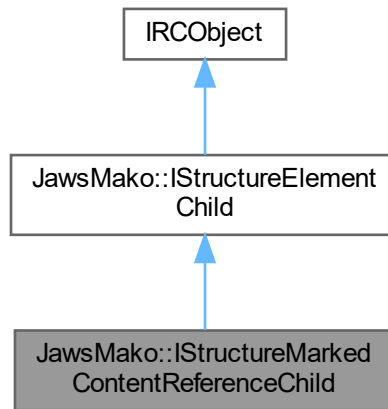
- [structure.h](#)

7.422 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.

7.422.1 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:

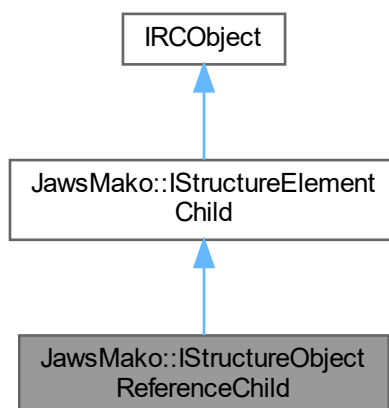
- [structure.h](#)

7.423 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.

7.423.1 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:

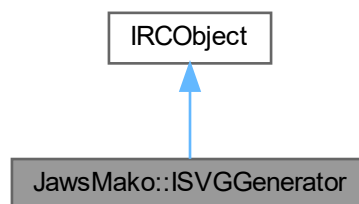
- [structure.h](#)

7.424 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 `IRAIInputStreamPtr` [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.
- virtual void [setVisibleZeroWidthLines](#) (bool visible)=0
Sets whether or not to generate visible zero width lines.

Public Member Functions inherited from [IRObject](#)

- virtual void [addRef](#) () const =0
Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool [decRef](#) () const =0
Decreases the reference count of the actual object pointed to. When the reference count falls to 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.

7.424.1 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.

7.424.2 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

<i>jawsMako</i>	The JawsMako instance.
<i>resourcePrefix</i>	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.
<i>progressMonitor</i>	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

<i>node</i>	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

<i>node</i>	The DOM node to be represented as SVG.
<i>outputStream</i>	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

<i>name</i>	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

<i>resources</i>	A reference to a vector to receive the entries.
------------------	---

setEnabledImageDownsampling()

```
virtual void JawsMako::ISVGGenerator::setEnabledImageDownsampling (
    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.

setVisibleZeroWidthLines()

```
virtual void JawsMako::ISVGGenerator::setVisibleZeroWidthLines (
    bool visible ) [pure virtual]
```

Sets whether or not to generate visible zero width lines.

```
Setting this value to true will convert zero width lines to
visible lines by setting the stroke width to 1 and setting
the vector-effect property to non-scaling-stroke.
```

```
The default value is false.
```

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

- `svggenerator.h`

7.425 Iterator Class Reference

A convenience iterator-like class for iterating through the contents of a dictionary.

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

7.425.1 Detailed Description

A convenience iterator-like class for iterating through the contents of a dictionary.

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

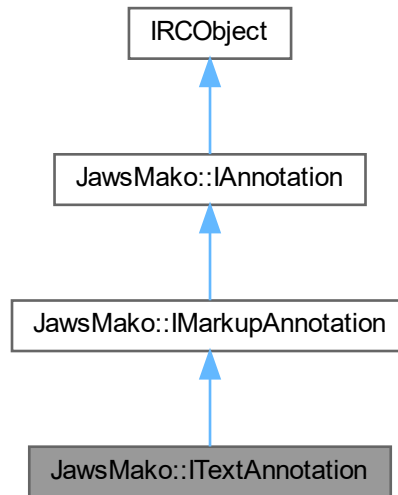
- [pdfobjects.h](#)

7.426 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.

7.426.1 Detailed Description

A generic interface class for a text (sticky note) annotation.

7.426.2 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

<i>jawsMako</i>	The JawsMako instance
<i>rect</i>	The annotations bounds. Must not be empty.
<i>contents</i>	The textual contents of the text annotation.
<i>open</i>	Optional; whether or not the note is open. Default is false.
<i>iconName</i>	Optional; the name of the icon to use for displaying the annotation. The default is "Note". Must not be empty.
<i>popup</i>	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

<i>iconName</i>	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

<i>open</i>	Set to true for open, false for closed
-------------	--

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

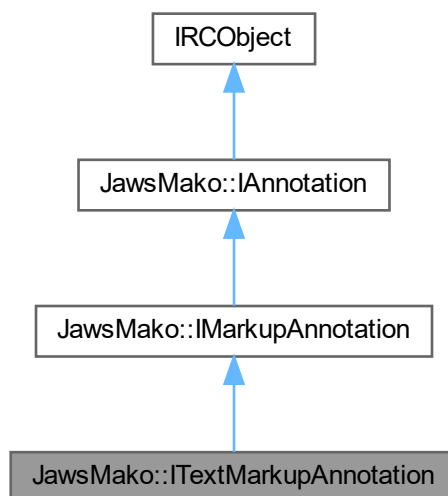
- [interactive.h](#)

7.427 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.

7.427.1 Detailed Description

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

7.427.2 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

<code>quadPoints</code>	The markup annotation's quad points
-------------------------	-------------------------------------

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

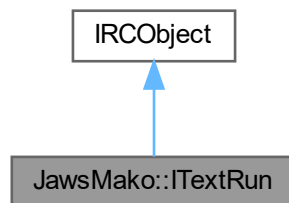
- [interactive.h](#)

7.428 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.

7.428.1 Detailed Description

A run of text, containing unicode information, the position, transformation and bounds of the text.

7.428.2 Member Function Documentation

getBoundsOnPage()

```
FRect JawsMako::ITextRun::getBoundsOnPage (
    bool tight = true ) const [inline]
```

Get the bounds of the run, in page coordinates. Convenience.

Parameters

<i>tight</i>	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

<i>tight</i>	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:

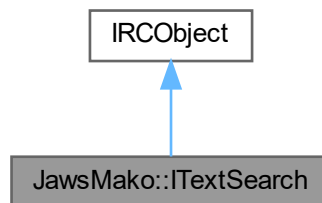
- [text.h](#)

7.429 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 &pageLayout)
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.

7.429.1 Detailed Description

Perform text searching using the page information obtained from an [IPageLayout](#).

7.429.2 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

<i>factory</i>	The factory to use
<i>pageLayout</i>	The page layout to use

Returns

ITextSearchPtr The new text searcher

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

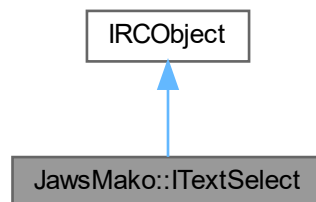
- [text.h](#)

7.430 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](#) &pageLayout)

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.

7.430.1 Detailed Description

Perform text selection using the page information obtained from an [IPageLayout](#).

7.430.2 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

<i>factory</i>	The factory to use
<i>pageLayout</i>	The page layout to use

Returns

ITextSelectPtr The new text selector

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

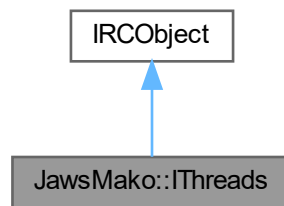
- [text.h](#)

7.431 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.

7.431.1 Detailed Description

An interface class for document threads, Currently a stub interface.

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

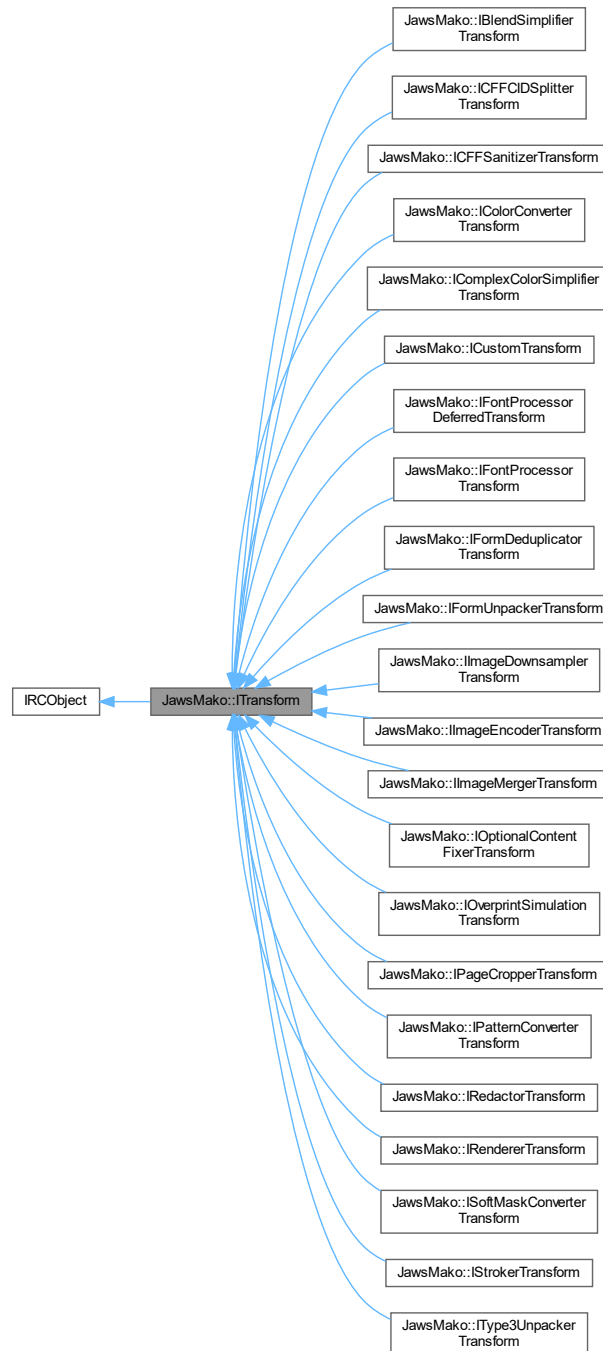
- [interactive.h](#)

7.432 JawsMako::ITransform Class Reference

ITransforms provide a method of applying common operations on DOM objects such as brushes, nodes, colors, colorspace 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.

7.432.1 Detailed Description

ITransforms provide a method of applying common operations on DOM objects such as brushes, nodes, colors, colorspace 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.

7.432.2 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

<i>progressMonitor</i>	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

<i>brush</i>	The brush to be processed.
<i>usage</i>	What the brush is to be used for.
<i>state</i>	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

<i>color</i>	The color to be processed.
<i>state</i>	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

<i>colorSpace</i>	The colorSpace to be processed.
<i>state</i>	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

<i>image</i>	The image to be processed.
<i>state</i>	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

<i>node</i>	The node to be processed.
<i>changed</i>	Will be set to true on return if any changes have been made.
<i>transformChildren</i>	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.
<i>state</i>	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

<i>page</i>	The page to be processed.
<i>transformContent</i>	If true, apply the transform to the page content
<i>transformAnnotations</i>	If true, apply the transform to the annotations

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

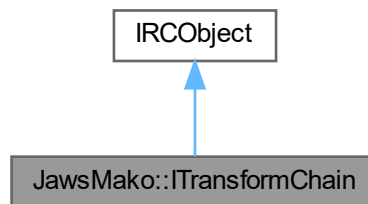
- [transforms.h](#)

7.433 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 IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr(), const [CTransformVect](#) &transforms=[CTransformVect](#)())
Create a new transform chain.

Additional Inherited Members

Protected Member Functions inherited from [IRCOObject](#)

- virtual [~IRCOObject](#) ()
Virtual destructor.

7.433.1 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.

7.433.2 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

<i>jawsMako</i>	The JawsMako instance.
<i>progressMonitor</i>	The progress monitor which allows aborting an operation or registering a progress callback.
<i>transforms</i>	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

<i>transform</i>	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

<i>transform</i>	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

<i>index</i>	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

<i>node</i>	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

<i>node</i>	The node to transform.
<i>changed</i>	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

<i>page</i>	The page to be processed.
<i>transformContent</i>	If true, apply the transform to the page content
<i>transformAnnotations</i>	If true, apply the transform to the annotations

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

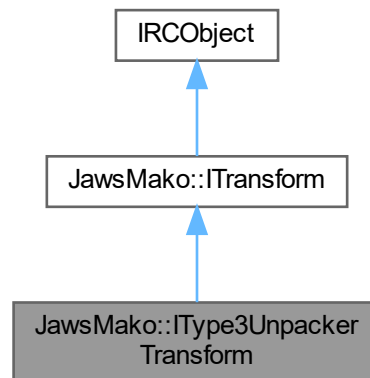
- [transforms.h](#)

7.434 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 JAWMAKO_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 transformAnnotations=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.

7.434.1 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.

7.434.2 Member Function Documentation

create()

```
static JAWSMAKO_API IType3UnpackerTransformPtr JawsMako::IType3UnpackerTransform::create (  
    const IJawsMakoPtr & jawsMako,  
    const IAbortPtr & abort = IAbortPtr() ) [static]
```

Create the transform.

Parameters

<i>jawsMako</i>	The JawsMako instance.
<i>abort</i>	An abort callback handler.

Returns

The new instance.

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

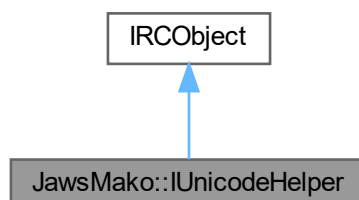
- [transforms.h](#)

7.435 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
Applies 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.

7.435.1 Detailed Description

An interface into language specific unicode helpers.

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

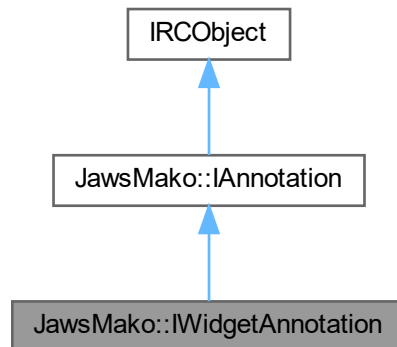
- [text.h](#)

7.436 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 highlight 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 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 &appearanceCharacteristics)=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::!Annotation`

- 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.

7.436.1 Detailed Description

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

7.436.2 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

<i>form</i>	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.
<i>onStateName</i>	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.
<i>font</i>	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.
<i>fontIndex</i>	The index of the font to use.
<i>captionSize</i>	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

<i>jawsMako</i>	The Mako instance
<i>form</i>	The form to receive the button field
<i>page</i>	The page to attach the annotation to
<i>rect</i>	The annotation area
<i>partialName</i>	The desired partial name of the field. If an empty string is passed, no partial name will be set.
<i>caption</i>	The string to use as the caption
<i>textSize</i>	The text size for the caption
<i>font</i>	The OpenType font to use for the field text
<i>fontIndex</i>	If font is a TrueType collection, the index of the font within the collection
<i>encodingHint</i>	A hint of the encoding to use for the font, if an encoding cannot be unambiguously detected from the font itself. Not usually critical for buttons. Must be consistent with other form users of this font.
<i>color</i>	The color to use for the caption. Must use a DeviceGray, DeviceRGB or DeviceCMYK colorspace. Default is black.
<i>backgroundColor</i>	The background color for the button
<i>actions</i>	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

<i>jawsMako</i>	The Mako instance
<i>form</i>	The form to receive the button field
<i>page</i>	The page to attach the annotation to
<i>rect</i>	The annotation area
<i>partialName</i>	The desired partial name of the field. If an empty string is passed, no partial name will be set.
<i>onValue</i>	The name used by the form to specify an on value. Must be supplied.
<i>on</i>	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

<i>jawsMako</i>	The Mako instance
<i>form</i>	The form to receive the button field
<i>page</i>	The page to attach the annotation to
<i>rect</i>	The annotation area
<i>partialName</i>	The desired partial name of the field. If an empty string is passed, no partial name will be set.
<i>combo</i>	If true, a combo box/popup will be created. If false, a selectable field will be created.
<i>editable</i>	If true, and combo is set, the combo box text will be editable
<i>multiSelect</i>	If true, and combo is not set, the choice field will allow multiple selections
<i>options</i>	The options present in the choice field
<i>selectedOptions</i>	The options that are initially selected
<i>textSize</i>	The text size
<i>font</i>	The OpenType font to use for the field text
<i>fontIndex</i>	If font is a TrueType collection, the index of the font within the collection
<i>encodingHint</i>	A hint of the encoding to use for the font, if an encoding cannot be unambiguously detected from the font itself. Must be consistent with other form users of this font.
<i>color</i>	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

<i>jawsMako</i>	The Mako instance
<i>form</i>	The form to receive the fields
<i>page</i>	The page to attach the annotation to
<i>partialName</i>	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.
<i>value</i>	The value of the field; that is, the name of the radio button that is checked. Pass an empty string if none are selected.
<i>buttonDetails</i>	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

<i>jawsMako</i>	The Mako instance
<i>form</i>	The form to receive the text field
<i>page</i>	The page to attach the annotation to

Parameters

<i>rect</i>	The annotation area
<i>partialName</i>	The desired partial name of the field. If an empty string is passed, no partial name will be set.
<i>text</i>	The text present in the field (the field's value)
<i>textSize</i>	The text size
<i>font</i>	The OpenType font to use for the field text
<i>fontIndex</i>	If font is a TrueType collection, the index of the font within the collection
<i>encodingHint</i>	A hint of the encoding to use for the font, if an encoding cannot be unambiguously detected from the font itself. Must be consistent with other form users of this font.
<i>color</i>	The color to use for the text. Must use a DeviceGray, DeviceRGB or DeviceCMYK colorspace
<i>multiLine</i>	True if the text should wrap within the field, false otherwise
<i>quadding</i>	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

<i>defaultAppearanceString</i>	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

<i>value</i>	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 the flags changes depending on type, and that the value may be inheritable.

Parameters

<i>flags</i>	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

<i>options</i>	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

<i>value</i>	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

<i>actions</i>	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

<i>actions</i>	Either an actions dictionary or NULL to clear
----------------	---

setAppearanceCharacteristics()

```
virtual void JawsMako::IWidgetAnnotation::setAppearanceCharacteristics (
    const IWidgetAppearanceCharacteristicsPtr & appearanceCharacteristics ) [pure
virtual]
```

Set the appearance characteristics.

Parameters

<i>appearanceCharacteristics</i>	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

<i>defaultAppearanceString</i>	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

<i>value</i>	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

<i>value</i>	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

<i>flags</i>	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

<i>option</i>	The desired index of the first option visible in a list
---------------	---

setHighlightMode()

```
virtual void JawsMako::IWidgetAnnotation::setHighlightMode (
    eHighlightMode highlightMode ) [pure virtual]
```

Set the highlight mode.

Parameters

<i>highlightMode</i>	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

<i>name</i>	The partial name to be set
-------------	----------------------------

setQuadding()

```
virtual void JawsMako::IWidgetAnnotation::setQuadding (
    eQuadding quadding ) [pure virtual]
```

Set the quadding (Justification) for variable text.

Parameters

<i>quadding</i>	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

<i>selectedOptions</i>	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

<i>value</i>	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

<i>value</i>	The string value for the field.
--------------	---------------------------------

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

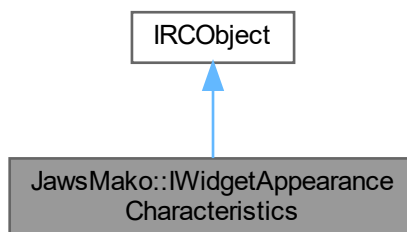
- [interactive.h](#)

7.437 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.

7.437.1 Detailed Description

Appearance information for a widget annotation.

7.437.2 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

<i>jawsMako</i>	The Mako instance
<i>rotation</i>	The rotation of the widget in degrees. Must be a multiple of 90.
<i>borderColor</i>	The color to use for the border. Must use a DeviceGray, DeviceRGB or DeviceCMYK colorspace.
<i>backgroundColor</i>	The color to use for the background. Must use a DeviceGray, DeviceRGB or DeviceCMYK colorspace.
<i>caption</i>	The widget's normal caption.
<i>rolloverCaption</i>	The widget's rollover caption. Used only for push button widgets.
<i>alternateCaption</i>	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

<i>caption</i>	The caption, or an empty string to clear
----------------	--

setBackground-color()

```
virtual void JawsMako::IWidgetAppearanceCharacteristics::setBackground-color (   
    const IDOMColorPtr & color ) [pure virtual]
```

Set the background color of the widget.

Parameters

<i>color</i>	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

<i>color</i>	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

<i>caption</i>	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

<i>caption</i>	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

<i>rotation</i>	The rotation in degrees. Must be a multiple of 90 degrees.
-----------------	--

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

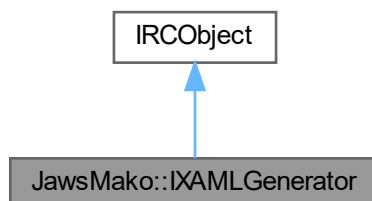
- [interactive.h](#)

7.438 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=IDOMCatalogPtr())=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 `generateXAMLForPageAndAnnotations()`.
- 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 [generateXAMLForPageAndAnnotations\(\)](#) 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 color 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 [IRCOject](#)

- virtual void `addRef` () const =0
Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
- virtual bool `decRef` () const =0
Decreases the reference count of the actual object pointed to. When the reference count falls to 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 [IRCOject](#)

- virtual `~IRCOject` ()
Virtual destructor.

7.438.1 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.

7.438.2 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

<i>jawsMako</i>	The JawsMako instance.
<i>resourcePrefix</i>	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.
<i>progressMonitor</i>	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

<i>node</i>	The DOM node to be represented as XAML.
<i>catalog</i>	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

<i>node</i>	The DOM node to be represented as XAML.
<i>outputStream</i>	The destination stream for the XAML.
<i>catalog</i>	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 [generateXAMLForPageAndAnnotations\(\)](#).

Parameters

<i>appearance</i>	The appearance to be presented as XAML.
<i>annotation</i>	The annotation that contains the appearance. Required in order to position the annotation against the page background correctly.
<i>page</i>	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

<i>appearance</i>	The appearance to be presented as XAML.
<i>annotation</i>	The annotation that contains the appearance. Required in order to position the annotation against the page background correctly.
<i>page</i>	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

<i>page</i>	The DOM node to be produced as XAML
<i>appearanceXAMLs</i>	A vector to receive the XAML for the annotation appearances.

Returns

IStreamPtr 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 [generateXAMLForPageAndAnnotations\(\)](#) to generate XAML for the page and the annotations in one bulk operation.

Parameters

<i>page</i>	The page whose annotations should be represented as XAML.
<i>appearanceXAMLs</i>	A vector to receive the XAML for the annotation appearances.

getResource()

```
virtual IStreamPtr JawsMako::IXAMLGenerator::getResource (
    const UString & name ) [pure virtual]
```

Get the stream for a named resource. An exception will be thrown if the resource cannot be found.

Parameters

<i>name</i>	The name of the resource.
-------------	---------------------------

Returns

IStreamPtr The resource stream.

getResources()

```
virtual void JawsMako::IXAMLGenerator::getResources (
    CEDLVector< CResourceEntry > & resources ) [pure virtual]
```

Get all the resources in a vector.

Parameters

<i>resources</i>	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 color images for XAML output.

The default behaviour is leave the image resolution unchanged.

Parameters

<i>resolution</i>	The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default.
<i>threshold</i>	The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution.
<i>method</i>	The method to use when downsampling. The default is bicubic for color 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

<i>resolution</i>	The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default.
<i>threshold</i>	The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution.
<i>method</i>	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

<i>quality</i>	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 advantageous 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

<i>resolution</i>	The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default.
<i>threshold</i>	The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution.
<i>method</i>	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

<i>format</i>	The desired format.
---------------	---------------------

setPreferredGrayImageFormat()

```
virtual void JawsMako::IXAMLGenerator::setPreferredGrayImageFormat (
    ImageEncoderTransform::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

<i>format</i>	The desired format.
---------------	---------------------

setPreferredMonoImageFormat()

```
virtual void JawsMako::IXAMLGenerator::setPreferredMonoImageFormat (
    ImageEncoderTransform::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

<i>format</i>	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

<i>resolution</i>	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

<i>targetSpace</i>	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

<i>profile</i>	The desired profile.
----------------	----------------------

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

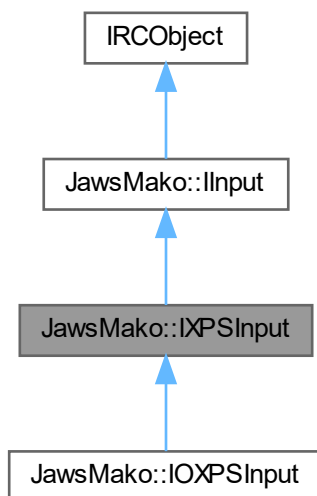
- xamlgenerator.h

7.439 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` ¶m, 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.

7.439.1 Detailed Description

An instance of the JawsMako XPS input class.

7.439.2 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

<i>jawsMako</i>	The IJawsMako object
<i>progressMonitor</i>	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 recommended 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:

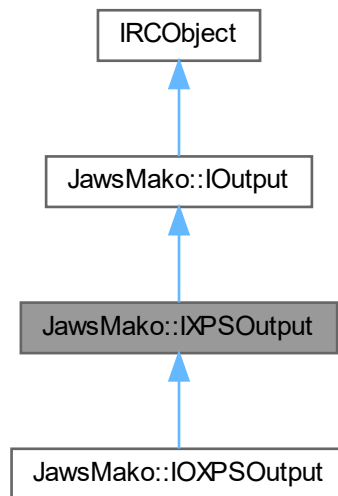
- `xpsinput.h`

7.440 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 "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 color 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 **setPreferredMonochromeImageFormat** (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 "MonochromeImageFormat" 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 ¶m, 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.

7.440.1 Detailed Description

Interface for the XPS [IOutput](#) class.

7.440.2 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

<i>jawsMako</i>	The IJawsMako object.
<i>progressMonitor</i>	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 color 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

<i>resolution</i>	The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default.
<i>threshold</i>	The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution.
<i>method</i>	The method to use when downsampling. The default is bicubic for color 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

<i>resolution</i>	The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default.
<i>threshold</i>	The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution.
<i>method</i>	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

<i>quality</i>	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 advantageous 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 "MonoImageDownsamplingMethod" with the respective values.

Parameters

<i>resolution</i>	The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default.
<i>threshold</i>	The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution.
<i>method</i>	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 (
    ImageEncoderTransform::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

<i>format</i>	The desired format.
---------------	---------------------

setPreferredGrayImageFormat()

```
virtual void JawsMako::IXPSOutput::setPreferredGrayImageFormat (
    ImageEncoderTransform::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

<i>format</i>	The desired format.
---------------	---------------------

setPreferredMonoImageFormat()

```
virtual void JawsMako::IXPSOutput::setPreferredMonoImageFormat (
    ImageEncoderTransform::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

<i>format</i>	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

<i>resolution</i>	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

<i>targetSpace</i>	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

<i>profile</i>	The desired profile.
----------------	----------------------

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

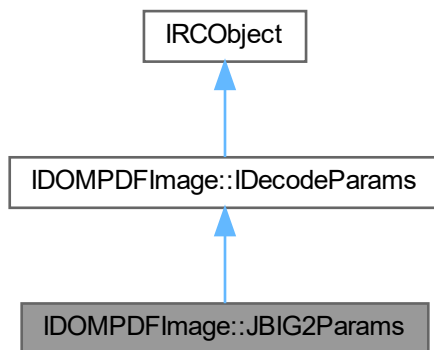
- [xpsoutput.h](#)

7.441 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 IRObject

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

7.441.1 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`

7.442 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.

7.442.1 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`

7.443 PValue Class Reference

Stores a "property" value that is tagged with an enumeration value that indicates the underlying type.

```
#include <edlproperty.h>
```

7.443.1 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](#)

7.444 SignatureID Class Reference

Opentype table signatures.

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

7.444.1 Detailed Description

Opentype table signatures.

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

- [idomfont.h](#)

8 File Documentation

- 8.1 [edlblend.h File Reference](#)
- 8.2 [edlmath.h File Reference](#)
- 8.3 [edlnamespaces.h File Reference](#)
- 8.4 [edlproperty.h File Reference](#)
- 8.5 [edlqname.h File Reference](#)
- 8.6 [edlquartz.h File Reference](#)
- 8.7 [edlstring.h File Reference](#)
- 8.8 [edltime.h File Reference](#)
- 8.9 [edltypes.h File Reference](#)
- 8.10 [edlvector.h File Reference](#)
- 8.11 [edlversion.h File Reference](#)
- 8.12 [idombrush.h File Reference](#)
- 8.13 [idomfont.h File Reference](#)
- 8.14 [idomfontpcl.h File Reference](#)
- 8.15 [idomform.h File Reference](#)
- 8.16 [idomnode.h File Reference](#)
- 8.17 [idomresources.h File Reference](#)
- 8.18 [iedlcollection.h File Reference](#)
- 8.19 [iedlenum.h File Reference](#)
- 8.20 [iedltree.h File Reference](#)
- 8.21 [ifilespec.h File Reference](#)
- 8.22 [iimagecodec.h File Reference](#)
- 8.23 [ircobject.h File Reference](#)
- 8.24 [memutils.h File Reference](#)
- 8.25 [objclassid.h File Reference](#)
- 8.26 [platform.h File Reference](#)
- 8.27 [smartptr.h File Reference](#)
- 8.28 [customtransform.h File Reference](#)
- 8.29 [distiller.h File Reference](#)
- 8.30 [iipdsinput.h File Reference](#)

Index

- add
 - IDOMGlyphIDEnumerator, [522](#)
- addAction
 - IDOMActionArray, [320](#)
- addAppearance
 - JawsMako::IAnnotation, [226](#)
- addAttribute
 - IDOMJobTkGenericNode, [642](#)
- addChildField
 - JawsMako::IFormField, [1016](#)
- addChildWidget
 - JawsMako::IFormField, [1016](#)
- addEmbeddedStream
 - JawsMako::IDocument, [307](#)
- addFeature
 - IDOMJobTkContent, [626](#)
- addField
 - JawsMako::IForm, [1003](#)
- addFigure
 - IDOMPathGeometry, [730](#)
- addFont
 - JawsMako::IDistiller, [286](#)
- addFonts
 - JawsMako::IDistiller, [286](#)
- addFrame
 - JawsMako::ILayout, [1124](#)
- addGlyph
 - IDOMType3Font, [946](#)
- addGradientStop
 - IDOMGradientBrush, [552](#)
- addNamedDestination
 - JawsMako::IDocument, [308](#)
- addNamespace
 - IDOMJobTkContent, [626](#), [627](#)
- addParameterInit
 - IDOMJobTkContent, [627](#)
- addPoint
 - IDOMPolyBezierSegment, [793](#)
 - IDOMPolyLineSegment, [796](#)
 - IDOMPolyQuadraticBezierSegment, [800](#)
- addRun
 - JawsMako::ILayoutParagraph, [1138](#)
- addSegment
 - IDOMPathFigure, [724](#)
- addStrokeDash
 - IDOMPathNode, [750](#)
- addToInitString
 - IDOMJobTkContent, [627](#)
- addWidget
 - JawsMako::IForm, [1003](#)
- afterBeginPageSetup
 - JawsMako::IPSInjector, [1385](#)
- afterBeginSetup
 - JawsMako::IPSInjector, [1385](#)
- afterLastByte
 - JawsMako::IPSInjector, [1385](#)
- Annotations, [84](#)
 - eAppearanceUsage, [85](#)
 - eAUDown, [85](#)
 - eAUNormal, [85](#)
 - eAURollover, [85](#)
- append
 - JawsMako::IPDFArray, [1270](#)
- appendCharacterData
 - IDOMJobTkGenericCharacterData, [637](#)
- appendChild
 - IDOMNode, [687](#)
- appendDocument
 - JawsMako::IDocumentAssembly, [315](#)
- appendPage
 - JawsMako::IDocument, [308](#)
- arcTo
 - IDOMPathGeometryBuilder, [739](#)
- asArray
 - CTransformMatrix< TItem >, [138](#)
- beforeEndPageSetup
 - JawsMako::IPSInjector, [1385](#)
- beforeEndSetup
 - JawsMako::IPSInjector, [1386](#)
- beforeFirstByte
 - JawsMako::IPSInjector, [1386](#)
- beforePsHeader
 - JawsMako::IPSInjector, [1386](#)
- beforeShowpage
 - JawsMako::IPSInjector, [1387](#)
- begin
 - JawsMako::IPDFDictionary, [1281](#)
- beginDocument
 - JawsMako::IOutputWriter, [1211](#)
- beginEnumeration
 - IDOMGlyphIDEnumerator, [523](#)
- BoxTmpl< PointType >, [106](#)
- Brushes, [54](#)
 - eAbsolute, [57](#)
 - eBrushType, [56](#)
 - eColorInterpolationMode, [56](#)
 - eFlipX, [57](#)
 - eFlipXY, [57](#)
 - eFlipY, [57](#)
 - eImage, [56](#)
 - eLinearGradient, [56](#)
 - eMasked, [56](#)
 - eNoSpread, [57](#)
 - eNoTile, [57](#)
 - eNull, [56](#)
 - ePad, [57](#)
 - eRadialGradient, [56](#)
 - eReflect, [57](#)
 - eRepeat, [57](#)

- eSRGBLinearInterpolation, 56
- eSoftMask, 56
- eSolidColor, 56
- eSpreadMethod, 57
- eSRGBLinearInterpolation, 56
- eTile, 57
- eTilingMode, 57
- eTilingPattern, 56
- eType1ShadingPattern, 56
- eType2ShadingPattern, 56
- eType3ShadingPattern, 56
- eType4567ShadingPattern, 56
- eViewUnits, 57
- eVisual, 56
- CAnnotationBorder
 - JawsMako::CAnnotationBorder, 107
- CClassID, 111
 - CClassID, 111, 112
 - equal, 112
 - operator=, 113
- CClassParams, 113
- CGlyphsCluster, 119
 - similar, 119
- CGlyphsClusters, 120
- CIndicesGlyph, 120
 - similar, 121
- classID
 - IDOMArcSegment, 329
 - IDOMAudioFile, 335
 - IDOMCachedImage, 341
 - IDOMCanvas, 349
 - IDOMCatalog, 354
 - IDOMCharPathGroup, 363
 - IDOMColor, 368
 - IDOMColorSpaceDeviceCMY, 388
 - IDOMColorSpaceDeviceCMYK, 391
 - IDOMColorSpaceDeviceGray, 394
 - IDOMColorSpaceDeviceN, 398
 - IDOMColorSpaceDeviceRGB, 404
 - IDOMColorSpaceICCBased, 407
 - IDOMColorSpaceIndexed, 411
 - IDOMColorSpaceLAB, 415
 - IDOMColorSpacescRGB, 419
 - IDOMColorSpacesGray, 422
 - IDOMColorSpacesRGB, 425
 - IDOMCompositelImage, 428
 - IDOMDeviceNColorant, 433
 - IDOMExponentialFunction, 436
 - IDOMFilteredImage, 443
 - IDOMFixedPage, 451
 - IDOMFont, 459
 - IDOMFontOpenType, 465
 - IDOMFontPCL5, 474
 - IDOMFontPCLXL, 477
 - IDOMFontSource, 480
 - IDOMFontSourceFromStream, 483
 - IDOMFontSourceObfuscationConverter, 487
 - IDOMForm, 496
 - IDOMFormInstance, 503
 - IDOMGlyph, 514
 - IDOMGlyphIDEnumerator, 523
 - IDOMGlyphs, 532
 - IDOMGradientStop, 557
 - IDOMGroup, 563
 - IDOMGroupingFunction, 568
 - IDOMICCProfile, 573
 - IDOMImageBitScalerFilter, 581
 - IDOMImageBleederFilter, 582
 - IDOMImageBrush, 586
 - IDOMImageChannelSelectorFilter, 594
 - IDOMImageColorConverterFilter, 596
 - IDOMImageColorKeyFilter, 597
 - IDOMImageColorSpaceSubstitutionFilter, 598
 - IDOMImageDecodeFilter, 600
 - IDOMImageDeindexerFilter, 601
 - IDOMImageDeviceNToBaseFilter, 602
 - IDOMImageDownsamplerFilter, 603
 - IDOMImageInverterFilter, 604
 - IDOMImageMaskExpanderFilter, 605
 - IDOMImageProperties, 608
 - IDOMImageSpotMergerFilter, 612
 - IDOMJobTk, 618
 - IDOMJobTkContent, 628
 - IDOMJobTkGenericCharacterData, 637
 - IDOMJobTkGenericNode, 642
 - IDOMJobTkNode, 648
 - IDOMJobTkOwner, 654
 - IDOMJobTkValue, 658
 - IDOMJPEGImage, 661
 - IDOMLinearGradientBrush, 667
 - IDOMMaskedBrush, 674
 - IDOMMatrix, 678
 - IDOMNullBrush, 701
 - IDOMPathFigure, 724
 - IDOMPathGeometry, 731
 - IDOMPathGeometryBuilder, 740
 - IDOMPathNode, 750
 - IDOMPCLImage, 776
 - IDOMPDFAlternateImage, 778
 - IDOMPDFImage, 783
 - IDOMPNGImage, 788
 - IDOMPolyBezierSegment, 793
 - IDOMPolyLineSegment, 796
 - IDOMPolyQuadraticBezierSegment, 800
 - IDOMPostScriptCalculatorFunction, 804
 - IDOMPrintTicket, 807
 - IDOMPSDImage, 810
 - IDOMRadialGradientBrush, 817
 - IDOMRawDataFile, 822
 - IDOMRawImage, 825
 - IDOMRecombineAlphaImage, 830
 - IDOMRecombineImage, 834
 - IDOMResourceDictionary, 840
 - IDOMSampledFunction, 843
 - IDOMShadingPatternType1Brush, 855
 - IDOMShadingPatternType2Brush, 861

- IDOMShadingPatternType3Brush, [868](#)
- IDOMShadingPatternType4567Brush, [877](#)
- IDOMShape, [886](#)
- IDOMSoftMaskBrush, [892](#)
- IDOMSolidColorBrush, [897](#)
- IDOMStitchingFunction, [908](#)
- IDOMTIFFImage, [915](#)
- IDOMTilingPatternBrush, [920](#)
- IDOMTransparencyGroup, [932](#)
- IDOMTrivialImage, [941](#)
- IDOMType3Font, [946](#)
- IDOMVisualBrush, [952](#)
- IDOMVisualRoot, [961](#)
- IDOMWMPImage, [963](#)
- IEDLNamespace, [969](#)
- IEDLTempStore, [978](#)
- IFontHeaderWriteSegmentBlockEnumerator, [989](#)
- IFontPCL5WriteSegmentBlockEnumerator, [992](#)
- classify
 - CTransformMatrix< TItem >, [138](#)
- clone
 - IEDLObject, [972](#)
 - JawsMako::IAnnotation, [226](#)
 - JawsMako::IAnnotationAppearance, [233](#)
 - JawsMako::IDocument, [308](#)
 - JawsMako::IDocumentAssembly, [315](#)
 - JawsMako::IForm, [1003](#)
 - JawsMako::IFormField, [1016](#)
 - JawsMako::ILayoutParagraph, [1139](#)
 - JawsMako::INamedDestination, [1173](#)
 - JawsMako::IPage, [1224](#)
 - JawsMako::IPageLabel, [1233](#)
 - JawsMako::IPDFObject, [1307](#)
 - JawsMako::IPDFObjectStore, [1311](#)
- cloneNode
 - IDOMNode, [687](#)
- cloneTree
 - IDOMNode, [687](#)
- cloneTreeAndAppend
 - IDOMNode, [688](#)
- close
 - IDOMPathGeometryBuilder, [740](#)
- Color profiles, [104](#)
- Color space, [102](#)
 - eColorSpaceType, [103](#)
 - eDeviceCMY, [103](#)
 - eDeviceCMYK, [103](#)
 - eDeviceGray, [103](#)
 - eDeviceN, [103](#)
 - eDeviceRGB, [103](#)
 - eICCBased, [103](#)
 - eIndexed, [103](#)
 - eLAB, [103](#)
 - escRGB, [103](#)
 - esGray, [103](#)
 - esRGB, [103](#)
- Color value, [101](#)
- Colors, [101](#)
- colors
 - IDOMShadingPatternType4567Brush::CMeshEntry, [122](#)
- comment
 - JawsMako::IPSCCommentMonitor, [1383](#)
- compare
 - IEDLTime, [984](#)
- completelyContainsShape
 - IDOMShape, [886](#)
- completeRead
 - IInputStream, [1093](#)
- completeReadE
 - IInputStream, [1094](#)
- completeWrite
 - IOutputStream, [1205](#)
- completeWriteE
 - IOutputStream, [1206](#)
- compose
 - CTransformMatrix< TItem >, [139](#)
- containsCompositeObject
 - JawsMako::IPDFArray, [1270](#)
 - JawsMako::IPDFDictionary, [1281](#)
- containsReferences
 - JawsMako::IPDFObject, [1307](#)
- convertColors
 - IColorManager, [259](#)
- convertToCubicBezierSegment
 - IDOMPolyQuadraticBezierSegment, [800](#)
- convertToQName
 - IDOMJobTkContent, [628](#)
- convertToSimpleSegment
 - IDOMArcSegment, [329](#)
- copy
 - IOutputStream, [1206](#)
 - JawsMako::IPDFArray, [1271](#)
 - JawsMako::IPDFDictionary, [1281](#)
- copyNodeData
 - IDOMNode, [688](#)
- CPDFFarReference
 - JawsMako::CPDFFarReference, [125](#)
- CPDFReference
 - JawsMako::CPDFReference, [128](#)
- create
 - IDOMArcSegment, [330](#)
 - IDOMCachedImage, [341](#)
 - IDOMColor, [368–370, 372](#)
 - IDOMColorSpaceDeviceCMY, [388](#)
 - IDOMColorSpaceDeviceCMYK, [391](#)
 - IDOMColorSpaceDeviceGray, [394](#)
 - IDOMColorSpaceDeviceN, [398, 399](#)
 - IDOMColorSpaceDeviceRGB, [404](#)
 - IDOMColorSpaceICCBased, [407](#)
 - IDOMColorSpaceIndexed, [411, 412](#)
 - IDOMColorSpaceLAB, [415](#)
 - IDOMColorSpacesescRGB, [419](#)
 - IDOMColorSpacesGray, [422](#)
 - IDOMColorSpacesRGB, [425](#)
 - IDOMCompositelImage, [428](#)

- IDOMFilteredImage, [443](#), [444](#)
- IDOMFixedPage, [451](#)
- IDOMFontOpenType, [465](#)
- IDOMForm, [496](#)
- IDOMFormInstance, [503](#)
- IDOMGlyph, [514](#), [515](#)
- IDOMGlyphs, [532](#)
- IDOMGradientStop, [557](#)
- IDOMGroup, [563](#)
- IDOMICCProfile, [573](#)
- IDOMImageBitScalerFilter, [581](#)
- IDOMImageBleederFilter, [582](#)
- IDOMImageBrush, [586](#)
- IDOMImageChannelSelectorFilter, [594](#)
- IDOMImageColorConverterFilter, [596](#)
- IDOMImageColorKeyFilter, [597](#)
- IDOMImageColorSpaceSubstitutionFilter, [598](#)
- IDOMImageDecodeFilter, [600](#)
- IDOMImageDeindexerFilter, [601](#)
- IDOMImageDeviceNToBaseFilter, [602](#)
- IDOMImageDownsamplerFilter, [603](#)
- IDOMImageInverterFilter, [604](#)
- IDOMImageMaskExpanderFilter, [605](#)
- IDOMImageProperties, [608](#)
- IDOMImageSpotMergerFilter, [612](#), [613](#)
- IDOMJPEGImage, [661](#)
- IDOMLinearGradientBrush, [667](#)
- IDOMMaskedBrush, [674](#)
- IDOMNullBrush, [701](#)
- IDOMOutline, [703](#)
- IDOMPageRectTarget, [714](#)
- IDOMPathFigure, [724](#)
- IDOMPathGeometry, [731](#), [732](#)
- IDOMPathGeometryBuilder, [740](#)
- IDOMPdFAternateImage, [778](#)
- IDOMPNGImage, [788](#)
- IDOMPolyBezierSegment, [793](#)
- IDOMPolyLineSegment, [797](#)
- IDOMPolyQuadraticBezierSegment, [801](#)
- IDOMPSDImage, [810](#)
- IDOMRadialGradientBrush, [817](#)
- IDOMRawImage, [825](#)
- IDOMRecombineAlphaImage, [830](#)
- IDOMRecombineImage, [834](#)
- IDOMShadingPatternType1Brush, [855](#)
- IDOMShadingPatternType2Brush, [861](#)
- IDOMShadingPatternType3Brush, [868](#)
- IDOMShadingPatternType4567Brush, [877](#), [878](#)
- IDOMSoftMaskBrush, [892](#)
- IDOMSolidColorBrush, [897](#)
- IDOMTIFFImage, [915](#)
- IDOMTilingPatternBrush, [920](#)
- IDOMTransparencyGroup, [932](#)
- IDOMTrivialImage, [941](#)
- IDOMVisualBrush, [952](#)
- JawsMako Library, [48](#)
- JawsMako::IAnnotationAppearance, [233](#)
- JawsMako::IBlendSimplifierTransform, [239](#)
- JawsMako::ICFFCIDSplitterTransform, [245](#)
- JawsMako::ICFFSanitizerTransform, [247](#)
- JawsMako::IColorConverterTransform, [251](#)
- JawsMako::IComplexColorSimplifierTransform, [276](#)
- JawsMako::IDistiller, [286](#)
- JawsMako::IDocument, [308](#)
- JawsMako::IDocumentAssembly, [315](#)
- JawsMako::IFontProcessorDeferredTransform, [996](#)
- JawsMako::IFontProcessorTransform, [999](#)
- JawsMako::IForm, [1003](#)
- JawsMako::IFormField, [1016](#)
- JawsMako::IFormUnpackerTransform, [1026](#)
- JawsMako::IJPDSInput, [1033](#)
- JawsMako::IImageDownsamplerTransform, [1040](#)
- JawsMako::IImageEncoderTransform, [1046](#)
- JawsMako::IImageMergerTransform, [1060](#)
- JawsMako::IInkAnnotation, [1080](#)
- JawsMako::IInput, [1082](#)
- JawsMako::IJawsRenderer, [1111](#)
- JawsMako::ILayout, [1125](#)
- JawsMako::ILayoutFont, [1129](#)
- JawsMako::ILayoutFontWeight, [1132](#)
- JawsMako::ILayoutImageRun, [1135](#)
- JawsMako::ILayoutParagraph, [1139](#)
- JawsMako::ILayoutTextRun, [1146–1148](#)
- JawsMako::ILinkAnnotation, [1158](#)
- JawsMako::INamedDestination, [1173](#)
- JawsMako::IOptionalContentFixerTransform, [1185](#)
- JawsMako::IOutput, [1197](#)
- JawsMako::IOverprintSimulationTransform, [1214](#)
- JawsMako::IOXPSInput, [1217](#)
- JawsMako::IOXPSOutput, [1220](#)
- JawsMako::IPage, [1224](#)
- JawsMako::IPageCropperTransform, [1230](#)
- JawsMako::IPageLabel, [1233](#)
- JawsMako::IPageLayout, [1236](#), [1237](#)
- JawsMako::IPageLayoutData, [1239](#)
- JawsMako::IPageLayoutNode, [1241](#)
- JawsMako::IPatternConverterTransform, [1245](#)
- JawsMako::IPCL5Input, [1248](#)
- JawsMako::IPCL5Output, [1255](#)
- JawsMako::IPCLXLInput, [1260](#)
- JawsMako::IPCLXLOutput, [1267](#)
- JawsMako::IPDFArray, [1271](#)
- JawsMako::IPDFBoolean, [1277](#)
- JawsMako::IPDFDictionary, [1282](#)
- JawsMako::IPDFFarReference, [1288](#)
- JawsMako::IPDFInput, [1292](#)
- JawsMako::IPDFInteger, [1302](#)
- JawsMako::IPDFNull, [1304](#)
- JawsMako::IPDFOutput, [1322](#)
- JawsMako::IPDFPageExtractor, [1343](#)
- JawsMako::IPDFPageInserter, [1347](#)
- JawsMako::IPDFReal, [1351](#)
- JawsMako::IPDFReference, [1354](#)
- JawsMako::IPDFString, [1358](#)
- JawsMako::IPJLParser, [1364](#)
- JawsMako::IPopupAnnotation, [1371](#)

- JawsMako::IPMLInput, [1374](#)
- JawsMako::IPRNInput, [1376](#)
- JawsMako::IProgressMonitor, [1378](#), [1379](#)
- JawsMako::IProgressTick, [1381](#), [1382](#)
- JawsMako::IPSInput, [1389](#)
- JawsMako::IPSOOutput, [1393](#)
- JawsMako::IRedactionAnnotation, [1418](#)
- JawsMako::IRedactorTransform, [1421](#)
- JawsMako::IRendererTransform, [1427](#)
- JawsMako::ISeparator, [1439](#)
- JawsMako::IShapeAnnotation, [1449](#)
- JawsMako::ISkiaRenderer, [1452](#)
- JawsMako::IStampAnnotation, [1461](#)
- JawsMako::IStrokerTransform, [1465](#)
- JawsMako::ISVGGenerator, [1480](#)
- JawsMako::ITextAnnotation, [1487](#)
- JawsMako::ITextSearch, [1495](#)
- JawsMako::ITextSelect, [1497](#)
- JawsMako::ITransformChain, [1506](#)
- JawsMako::IType3UnpackerTransform, [1510](#)
- JawsMako::IWidgetAppearanceCharacteristics, [1530](#)
- JawsMako::IXAMLGenerator, [1536](#)
- JawsMako::IXPSInput, [1545](#)
- JawsMako::IXPSOutput, [1549](#)
- Transforms, [95](#), [96](#)
- createAnnotationReferenceFromTag
 - JawsMako::IAnnotationUtils, [237](#)
- createBasicAppearances
 - JawsMako::IWidgetAnnotation, [1517](#)
- createButton
 - JawsMako::IWidgetAnnotation, [1517](#)
- createCalibratedGrayProfile
 - IColorManager, [261](#)
- createCalibratedRGBProfile
 - IColorManager, [261](#)
- createCheckBox
 - JawsMako::IWidgetAnnotation, [1518](#)
- createChoiceField
 - JawsMako::IWidgetAnnotation, [1519](#)
- createCompositeStream
 - IInputStream, [1094](#)
- createEllipse
 - IDOMPathGeometry, [732](#)
- createFilled
 - IDOMPathNode, [750](#)
- createFrame
 - IImageEncoder, [1043](#)
- createFromArray
 - IDOMColor, [374](#)
- createFromFBox
 - JawsMako::IPDFArray, [1272](#)
- createFromFile
 - IInputStream, [1094](#), [1095](#)
- createFromFileShared
 - IInputStream, [1095](#)
- createFromFlateCompressed
 - IInputStream, [1096](#)
- createFromLz4Compressed
 - IInputStream, [1096](#)
- createFromMemory
 - IInputStream, [1097](#)
- createFromNewFileWithContents
 - IInputStream, [1097](#), [1098](#)
- createFromNumbers
 - JawsMako::IPDFArray, [1272](#)
- createFromNumbersArray
 - JawsMako::IPDFArray, [1272](#)
- createFromPredictorCompressed
 - IInputStream, [1098](#)
- createFromRange
 - JawsMako::ILayoutFontWeight, [1132](#)
- createFromRAUserFunc
 - IInputStream, [1099](#)
- createFromUserReadFunc
 - IInputStream, [1099](#)
- createFromUserWriteFunc
 - IOutputStream, [1207](#)
- createFromVect
 - IDOMColor, [374](#)
 - JawsMako::ILayoutFontWeight, [1132](#)
- createGeometry
 - IDOMPathGeometryBuilder, [740](#)
- createGrayProfile
 - IColorManager, [262](#)
- createImage
 - IDOMPathNode, [751](#)
- createImageDecoder
 - IDOMImage, [578](#)
- createImageEncoder
 - IDOMImage, [578](#)
- createInstance
 - IEDLClassFactory, [964](#)
- createInstanceJawsMako
 - IEDLClassFactory, [965](#)
- createMarkedContentReferencePair
 - JawsMako::IStructureElement, [1471](#)
- createNewDOMid
 - IDOMCatalog, [354](#), [355](#)
- createNode
 - IDOMOutlineEntry, [707](#)
- createNormalAppearance
 - JawsMako::IMarkupAnnotation, [1168](#)
- createNumber
 - JawsMako::IPDFObject, [1307](#)
- createPdfBrush
 - IDOMLinearGradientBrush, [668](#)
- createPolygon
 - IDOMPathGeometry, [732](#)
- createPushbackStream
 - IInputStream, [1099](#)
- createRadioButtons
 - JawsMako::IWidgetAnnotation, [1519](#)
- createRandomAccessFromNonRandomAccess
 - IInputStream, [1100](#)
- createReader

- IEDLTempStoreObject, 981
- createRenamedFont
 - IDOMFontOpenType, 465
- createShading
 - IDOMLinearGradientBrush, 668
 - IDOMRadialGradientBrush, 818
- createSharedFromStream
 - IInputStream, 1100
- createSolidCmyk
 - IDOMColor, 375
 - IDOMSolidColorBrush, 897
- createSolidGray
 - IDOMColor, 375
 - IDOMSolidColorBrush, 898
- createSolidRgb
 - IDOMColor, 377
 - IDOMSolidColorBrush, 898
- createStroked
 - IDOMPathNode, 751
- createSubFile
 - IInputStream, 1101
- createSubsetFont
 - IDOMFontOpenType, 465
- createTbcStream
 - IInputStream, 1101
- createTemporaryObject
 - IEDLTempStore, 978
- createTemporaryObjectWithContents
 - IEDLTempStore, 978
- createTemporaryReaderWriter
 - IEDLTempStore, 978
- createTemporaryReaderWriterPair
 - IEDLTempStore, 979
- createTemporaryStreamWithContents
 - IEDLTempStore, 979
- createText
 - JawsMako::IPDFString, 1359
- createTextField
 - JawsMako::IWidgetAnnotation, 1520
- createToFile
 - IOutputStream, 1207
- createToFlateCompressed
 - IOutputStream, 1208
- createToLz4Compressed
 - IOutputStream, 1208
- createTrueTypeOnlyFontVersion
 - IDOMFontOpenType, 466
- createUelStream
 - IInputStream, 1101
- createWithSpaceAndComponents
 - IDOMSolidColorBrush, 899
- createWithSpaceAndComponentsFromArray
 - IDOMSolidColorBrush, 900
- createWithSpaceAndComponentsFromVect
 - IDOMSolidColorBrush, 901
- createWriter
 - IEDLTempStoreObject, 981
- createWriterAndImage
 - IDOMJPEGImage, 662
 - IDOMPNGImage, 788
 - IDOMRawImage, 826
 - IDOMTIFFImage, 915
- createXyzProfile
 - IColorManager, 262
- CTemporaryStoreParameters
 - JawsMako::CTemporaryStoreParameters, 133
- CTransformMatrix
 - CTransformMatrix< TItem >, 137, 138
- CTransformMatrix< TItem >, 135
 - asArray, 138
 - classify, 138
 - compose, 139
 - CTransformMatrix, 137, 138
 - decompose, 139
 - degenerate, 140
 - determinant, 140
 - dx, 140
 - dy, 140
 - equal, 140
 - identity, 141
 - invert, 141
 - iTransform, 141, 142
 - operator=, 142
 - postMul, 142
 - preMul, 143
 - rotate, 143
 - scale, 143
 - set, 144
 - setDX, 144
 - setDY, 144
 - setXX, 145
 - setXY, 145
 - setYX, 145
 - setYY, 145
 - transform, 146
 - transformRect, 146
 - translate, 147
 - xx, 147
 - xy, 147
 - yx, 147
 - yy, 147
- curveTo
 - IDOMPathGeometryBuilder, 741
- customtransform.h, 1558
- dash
 - JawsMako::CAnnotationBorder, 107
- decompose
 - CTransformMatrix< TItem >, 139
- decRef
 - IRCOBJECT, 1414
- deepClone
 - JawsMako::IPDFObject, 1307
- degenerate
 - CTransformMatrix< TItem >, 140
- delIndex
 - IDOMShadingPatternBrush, 848

- deleteDeviceLinkForConversionsBetween
 - IColorManager, [263](#)
- determinant
 - CTransformMatrix< TItem >, [140](#)
- determineUri
 - IDOMFontSource, [480](#)
 - IDOMFontSourceFromStream, [483](#)
 - IDOMFontSourceObfuscationConverter, [487](#)
 - IDOMFontSourceStreamFilter, [491](#)
- difference
 - IDOMShape, [886](#)
 - JawsMako::ILayoutFontWeight, [1133](#)
- dirtyObject
 - JawsMako::IPDFObjectStore, [1311](#)
- distill
 - JawsMako::IDistiller, [287](#)
- distiller.h, [1558](#)
- documentExists
 - JawsMako::IDocumentAssembly, [315](#)
- DOM Objects, [53](#)
- DOMNodeFlags
 - IDOMNodeFlags, [698](#)
- drawNode
 - JawsMako::ISkiaRenderer, [1452](#)
- dx
 - CTransformMatrix< TItem >, [140](#)
- dy
 - CTransformMatrix< TItem >, [140](#)
- eAbsolute
 - Brushes, [57](#)
- eActionArray
 - Outlines, [78](#)
- eActionGoTo3DView
 - Outlines, [78](#)
- eActionGoToE
 - Outlines, [77](#)
- eActionGoToR
 - Outlines, [77](#)
- eActionHide
 - Outlines, [77](#)
- eActionImportData
 - Outlines, [77](#)
- eActionJavaScript
 - Outlines, [78](#)
- eActionLaunch
 - Outlines, [77](#)
- eActionMovie
 - Outlines, [77](#)
- eActionNamed
 - Outlines, [77](#)
- eActionRendition
 - Outlines, [78](#)
- eActionResetForm
 - Outlines, [77](#)
- eActionSetOCGState
 - Outlines, [78](#)
- eActionSound
 - Outlines, [77](#)
- eActionSubmitForm
 - Outlines, [77](#)
- eActionThread
 - Outlines, [77](#)
- eActionTrans
 - Outlines, [78](#)
- eAnnotationEditingAllowed
 - PDF-specific, [72](#)
- eAnnotationType
 - JawsMako::IAnnotation, [225](#)
- eAny
 - JawsMako::ILayoutFontWeight, [1131](#)
- eAppearanceUsage
 - Annotations, [85](#)
- eAT3D
 - JawsMako::IAnnotation, [225](#)
- eATCaret
 - JawsMako::IAnnotation, [225](#)
- eATCircle
 - JawsMako::IAnnotation, [226](#)
- eATFileAttachment
 - JawsMako::IAnnotation, [226](#)
- eATFreeText
 - JawsMako::IAnnotation, [226](#)
- eATHighlight
 - JawsMako::IAnnotation, [226](#)
- eATInk
 - JawsMako::IAnnotation, [226](#)
- eATLine
 - JawsMako::IAnnotation, [226](#)
- eATLink
 - JawsMako::IAnnotation, [226](#)
- eATMovie
 - JawsMako::IAnnotation, [226](#)
- eATOther
 - JawsMako::IAnnotation, [226](#)
- eATPolygon
 - JawsMako::IAnnotation, [226](#)
- eATPolyLine
 - JawsMako::IAnnotation, [226](#)
- eATPopup
 - JawsMako::IAnnotation, [226](#)
- eATPrinterMark
 - JawsMako::IAnnotation, [226](#)
- eATProjection
 - JawsMako::IAnnotation, [226](#)
- eATRedact
 - JawsMako::IAnnotation, [226](#)
- eATRichMedia
 - JawsMako::IAnnotation, [226](#)
- eATScreen
 - JawsMako::IAnnotation, [226](#)
- eATSound
 - JawsMako::IAnnotation, [226](#)
- eATSquare
 - JawsMako::IAnnotation, [226](#)
- eATSquiggly
 - JawsMako::IAnnotation, [226](#)

- eATStamp
 - JawsMako::IAnnotation, [226](#)
- eATStrikeOut
 - JawsMako::IAnnotation, [226](#)
- eATText
 - JawsMako::IAnnotation, [226](#)
- eATTrapNet
 - JawsMako::IAnnotation, [226](#)
- eATUnderline
 - JawsMako::IAnnotation, [226](#)
- eATWatermark
 - JawsMako::IAnnotation, [226](#)
- eATWidget
 - JawsMako::IAnnotation, [226](#)
- eAUDown
 - Annotations, [85](#)
- eAUNormal
 - Annotations, [85](#)
- eAURollover
 - Annotations, [85](#)
- eBevelJoin
 - Geometry, [68](#)
- eBevelLongMitters
 - Geometry, [68](#)
- eBlack
 - JawsMako::ILayoutFontWeight, [1131](#)
- eBlendMode
 - Transparency blend modes, [99](#)
- eBlendModeColor
 - Transparency blend modes, [99](#)
- eBlendModeColorBurn
 - Transparency blend modes, [99](#)
- eBlendModeColorDodge
 - Transparency blend modes, [99](#)
- eBlendModeDarken
 - Transparency blend modes, [99](#)
- eBlendModeDifference
 - Transparency blend modes, [99](#)
- eBlendModeExclusion
 - Transparency blend modes, [99](#)
- eBlendModeHardLight
 - Transparency blend modes, [99](#)
- eBlendModeHue
 - Transparency blend modes, [99](#)
- eBlendModeLighten
 - Transparency blend modes, [99](#)
- eBlendModeLuminosity
 - Transparency blend modes, [99](#)
- eBlendModeMultiply
 - Transparency blend modes, [99](#)
- eBlendModeNormal
 - Transparency blend modes, [99](#)
- eBlendModeOverlay
 - Transparency blend modes, [99](#)
- eBlendModeSaturation
 - Transparency blend modes, [99](#)
- eBlendModeScreen
 - Transparency blend modes, [99](#)
- eBlendModeSoftLight
 - Transparency blend modes, [99](#)
- eBlendModeUnspecified
 - Transparency blend modes, [99](#)
- eBold
 - JawsMako::ILayoutFontWeight, [1131](#)
- eBorderStyle
 - JawsMako::CAnnotationBorder, [107](#)
- eBrushType
 - Brushes, [56](#)
- eBrushUsage
 - Transforms, [94](#)
- eBTBeveled
 - JawsMako::CAnnotationBorder, [107](#)
- eBTDashed
 - JawsMako::CAnnotationBorder, [107](#)
- eBTInset
 - JawsMako::CAnnotationBorder, [107](#)
- eBTSolid
 - JawsMako::CAnnotationBorder, [107](#)
- eBTUnderline
 - JawsMako::CAnnotationBorder, [107](#)
- eBUAlternateImage
 - Transforms, [95](#)
- eBUFill
 - Transforms, [95](#)
- eBUGeneral
 - Transforms, [95](#)
- eBUNone
 - Transforms, [95](#)
- eBUOpacityMask
 - Transforms, [95](#)
- eBUStroke
 - Transforms, [95](#)
- eChangeEncryptionAllowed
 - IDOMPublicKeyPDFSecurityInfo, [812](#)
- eClipLongMitters
 - Geometry, [68](#)
- eColorInterpolationMode
 - Brushes, [56](#)
- eColorSpaceType
 - Color space, [103](#)
- eContentAccessibilityExtractionAllowed
 - PDF-specific, [72](#)
- eCopyingAllowed
 - PDF-specific, [72](#)
- eCoreProperties
 - Metadata, [79](#)
- eDBMLongEdge
 - JawsMako::IPJLParser, [1364](#)
- eDBMShortEdge
 - JawsMako::IPJLParser, [1364](#)
- eDecimal
 - JawsMako::IPageLabel, [1233](#)
- eDeobfuscate
 - Fonts, [61](#)
- eDeviceCMY
 - Color space, [103](#)

- eDeviceCMYK
 - Color space, [103](#)
- eDeviceGray
 - Color space, [103](#)
- eDeviceN
 - Color space, [103](#)
- eDeviceRGB
 - Color space, [103](#)
- eDIAuthor
 - JawsMako::IDistiller, [284](#)
- eDICreator
 - JawsMako::IDistiller, [284](#)
- eDIN
 - JawsMako::IDistiller, [284](#)
- eDISubject
 - JawsMako::IDistiller, [284](#)
- eDITitle
 - JawsMako::IDistiller, [284](#)
- EDL_ERR_ABORTED
 - Error handling, [50](#)
- EDL_ERR_ADD_OBJECT_RESIDICT
 - Error handling, [52](#)
- EDL_ERR_APPEND_OUTLINE_TO_CLOSED_NODE
 - Error handling, [53](#)
- EDL_ERR_BAD_ARGUMENTS
 - Error handling, [50](#)
- EDL_ERR_BAD_BRUSH
 - Error handling, [51](#)
- EDL_ERR_BAD_COLOR_SPACE
 - Error handling, [53](#)
- EDL_ERR_BAD_DIMENSIONS
 - Error handling, [51](#)
- EDL_ERR_BAD_FONT
 - Error handling, [51](#)
- EDL_ERR_BAD_FONT_MAP
 - Error handling, [50](#)
- EDL_ERR_BAD_GEOMETRY
 - Error handling, [51](#)
- EDL_ERR_BOUNDS_CALCULATION_FAILED
 - Error handling, [51](#)
- EDL_ERR_BROKEN_ELEMENT_SEQUENCE
 - Error handling, [52](#)
- EDL_ERR_COLOR_CONVERSION_FAILURE
 - Error handling, [53](#)
- EDL_ERR_COMPRESS
 - Error handling, [50](#)
- EDL_ERR_CONFIG
 - Error handling, [50](#)
- EDL_ERR_CORRUPT_PNG_IMAGE
 - Error handling, [50](#)
- EDL_ERR_DEFLATE
 - Error handling, [52](#)
- EDL_ERR_DELETE_OUTLINE_WITH_CLOSED_PARENT
 - Error handling, [53](#)
- EDL_ERR_DEVICE_OUTOFMEMORY
 - Error handling, [52](#)
- EDL_ERR_DICT_ITEM_PRESENT
 - Error handling, [52](#)
- EDL_ERR_DOM_CREATION_FAILED
 - Error handling, [51](#)
- EDL_ERR_DTD_CONTENT
 - Error handling, [51](#)
- EDL_ERR_DUPLICATE_URI
 - Error handling, [52](#)
- EDL_ERR_FAILS_TO_ADD_OUTLINE_TO_METADATA_STORAGE
 - Error handling, [53](#)
- EDL_ERR_FALLBACK_BEFORE_CHOICE
 - Error handling, [52](#)
- EDL_ERR_FILENOTFOUND
 - Error handling, [50](#)
- EDL_ERR_IMAGE_DECODE_FAILURE
 - Error handling, [53](#)
- EDL_ERR_IMAGE_ENCODE_FAILURE
 - Error handling, [53](#)
- EDL_ERR_IMAGE_PARAMETER_OUT_OF_RANGE
 - Error handling, [53](#)
- EDL_ERR_IMMUTABLE_GEOMETRY
 - Error handling, [51](#)
- EDL_ERR_INCOMPATIBLE_IMAGE_PARAMETER
 - Error handling, [53](#)
- EDL_ERR_INCOMPATIBLE_PDFA
 - Error handling, [53](#)
- EDL_ERR_INCOMPATIBLE_PDFX
 - Error handling, [53](#)
- EDL_ERR_INTERNAL_RIP
 - Error handling, [53](#)
- EDL_ERR_INVALID_ABBR_PATH
 - Error handling, [52](#)
- EDL_ERR_INVALID_ATTRIBUTE
 - Error handling, [51](#)
- EDL_ERR_INVALID_COLOR_SPECIFICATION
 - Error handling, [52](#)
- EDL_ERR_INVALID_DOM_ID
 - Error handling, [51](#)
- EDL_ERR_INVALID_ELEMENT
 - Error handling, [52](#)
- EDL_ERR_INVALID_FONT
 - Error handling, [53](#)
- EDL_ERR_INVALID_GLYPHS
 - Error handling, [51](#)
- EDL_ERR_INVALID_IMAGE
 - Error handling, [52](#)
- EDL_ERR_INVALID_PDF_PASSWORD
 - Error handling, [53](#)
- EDL_ERR_INVALID_THUMBNAIL_TYPE
 - Error handling, [52](#)
- EDL_ERR_INVALID_TYPE3_GLYPH
 - Error handling, [51](#)
- EDL_ERR_INVALID_URI
 - Error handling, [52](#)
- EDL_ERR_INVALID_ZIP
 - Error handling, [51](#)
- EDL_ERR_IOERROR
 - Error handling, [50](#)
- EDL_ERR_KEY_SET_NON_RESOURCE
 - Error handling, [52](#)

- EDL_ERR_LARGE_FILE
 - Error handling, [50](#)
- EDL_ERR_LICENSE
 - Error handling, [50](#)
- EDL_ERR_LICENSE_BAD_CONFIG
 - Error handling, [53](#)
- EDL_ERR_LICENSE_INVALID
 - Error handling, [53](#)
- EDL_ERR_LICENSE_MISSING
 - Error handling, [53](#)
- EDL_ERR_MISSING_REQ_ATTRIBUTE
 - Error handling, [51](#)
- EDL_ERR_MISSING_RESOURCE
 - Error handling, [51](#)
- EDL_ERR_MISSING_XPS_PART
 - Error handling, [52](#)
- EDL_ERR_MORE_THAN_ONE_ELEMENT
 - Error handling, [52](#)
- EDL_ERR_MORE_THAN_ONE_GROUP_ELEMENT
 - Error handling, [52](#)
- EDL_ERR_MULTIPLE_COREPROPERTIES_RELS
 - Error handling, [52](#)
- EDL_ERR_MULTIPLE_DOCSTRUCTURE_RELS
 - Error handling, [52](#)
- EDL_ERR_MULTIPLE_STARTPARTS_RELS
 - Error handling, [52](#)
- EDL_ERR_MULTIPLE_THUMBNAIL_RELS
 - Error handling, [52](#)
- EDL_ERR_NAMESPACE_NOT_UNDERSTOOD
 - Error handling, [52](#)
- EDL_ERR_NO_COLOR_NAME
 - Error handling, [51](#)
- EDL_ERR_NO_CONTENT_TYPE
 - Error handling, [51](#)
- EDL_ERR_NO_KEY_RESIDICT_OBJECT
 - Error handling, [52](#)
- EDL_ERR_NO_REQ_ELEMENT
 - Error handling, [52](#)
- EDL_ERR_NO_REQ_GROUP_ELEMENT
 - Error handling, [52](#)
- EDL_ERR_NODE_ERROR
 - Error handling, [51](#)
- EDL_ERR_NULL_NODE
 - Error handling, [51](#)
- EDL_ERR_NUMBER_REQ_ELEMENT
 - Error handling, [52](#)
- EDL_ERR_OPEN_PDF
 - Error handling, [53](#)
- EDL_ERR_OPENFORREAD
 - Error handling, [50](#)
- EDL_ERR_OPENFORWRITE
 - Error handling, [50](#)
- EDL_ERR_OUTOFMEMORY
 - Error handling, [50](#)
- EDL_ERR_PAGE_TOO_LARGE
 - Error handling, [50](#)
- EDL_ERR_PANIC
 - Error handling, [50](#)
- EDL_ERR_PDFOUT_GENERAL_FAILURE
 - Error handling, [53](#)
- EDL_ERR_PREFIX_NOT_DEFINED
 - Error handling, [52](#)
- EDL_ERR_PREFIXED_ATTRIBUTE
 - Error handling, [52](#)
- EDL_ERR_PRESERVED_ITEMS_NOT_DECLARED_IGNOREABLE
 - Error handling, [52](#)
- EDL_ERR_PRINTTICKET_EXISTS
 - Error handling, [52](#)
- EDL_ERR_PROCESS_CONTENT_NOT_DECLARED_IGNOREABLE
 - Error handling, [52](#)
- EDL_ERR_PROCESS_PART
 - Error handling, [51](#)
- EDL_ERR_PROPERTY_ALREADY_SET
 - Error handling, [51](#)
- EDL_ERR_PSOUT_GENERAL_FAILURE
 - Error handling, [53](#)
- EDL_ERR_READ
 - Error handling, [50](#)
- EDL_ERR_REMOTERESOURCE_REF
 - Error handling, [51](#)
- EDL_ERR_RESTRICTED_FONT
 - Error handling, [50](#)
- EDL_ERR_TYPE3_GLYPH_NOT_FOUND
 - Error handling, [51](#)
- EDL_ERR_UNDEFINED
 - Error handling, [50](#)
- EDL_ERR_UNEXPECTED_RESOURCE
 - Error handling, [52](#)
- EDL_ERR_UNICODE
 - Error handling, [50](#)
- EDL_ERR_UNPREFIXED_ATTRIBUTE
 - Error handling, [52](#)
- EDL_ERR_UNSUPPORTED_GLYPHS
 - Error handling, [52](#)
- EDL_ERR_VECTOR_ERROR
 - Error handling, [50](#)
- EDL_ERR_WRITE
 - Error handling, [50](#)
- EDL_ERR_XML_PARSER
 - Error handling, [52](#)
- EDL_ERR_XPS_FROM_ENCRYPTED_SOURCE
 - Error handling, [53](#)
- EDL_ERR_XPS_PROVIDER
 - Error handling, [51](#)
- EDL_ERR_ZIP_PATH_NOT_FOUND
 - Error handling, [53](#)
- EDL_ERR_ZIP_WRITE_ONLY
 - Error handling, [53](#)
- edlblend.h, [1558](#)
- EDLErrorCode
 - Error handling, [49](#)
- edlExclusiveMakeTempDir
 - Platform-dependent, [88](#)
- edlFopen
 - Platform-dependent, [89](#)
- edlGetProcessId

- Platform-dependent, [89](#)
- edlGetTemporaryDirectory
 - Platform-dependent, [89](#)
- EDLIFStream, [216](#)
- edlMakeTempDir
 - Platform-dependent, [89](#)
- edlMakeTempDirProvidingSubDirPath
 - Platform-dependent, [90](#)
- edlmath.h, [1558](#)
- edlMkdir
 - Platform-dependent, [90](#)
- edlnamespaces.h, [1558](#)
- EDLOFStream, [217](#)
- edlproperty.h, [1558](#)
- EDLQName, [217](#)
 - EDLQName, [218](#)
 - getName, [218](#)
 - getNamespace, [218](#)
 - getNameWithPrefix, [219](#)
 - isEmpty, [219](#)
 - operator=, [219](#)
 - operator==, [219](#)
 - setName, [220](#)
 - setNamespace, [220](#)
- edlqname.h, [1558](#)
- edlquartz.h, [1558](#)
- edlRmdir
 - Platform-dependent, [91](#)
- edlSnprintf
 - Platform-dependent, [91](#)
- edlstring.h, [1558](#)
- edlStrtod
 - Platform-dependent, [91](#)
- edltime.h, [1558](#)
- edltypes.h, [1558](#)
- edlvector.h, [1558](#)
- edlversion.h, [1558](#)
- edlVsnprintf
 - Platform-dependent, [92](#)
- eDocumentAssemblyAllowed
 - PDF-specific, [72](#)
- eDocumentInfo
 - JawsMako::IDistiller, [284](#)
 - Metadata, [79](#)
- eDoesRotate
 - Geometry, [67](#)
- eDoesScale
 - Geometry, [67](#)
- eDoesTranslate
 - Geometry, [67](#)
- eDOMAnnotationAppearanceNode
 - Nodes, [63](#)
- eDOMCanvasNode
 - Nodes, [63](#)
- eDOMCharPathGroupNode
 - Nodes, [63](#)
- eDOMContentRootNode
 - Nodes, [63](#)
- eDOMDocumentNode
 - Nodes, [63](#)
- eDOMDocumentSequenceNode
 - Nodes, [63](#)
- eDOMFixedDocumentNode
 - Nodes, [63](#)
- eDOMFixedPageNode
 - Nodes, [63](#)
- eDOMFormInstanceNode
 - Nodes, [63](#)
- eDOMFormNode
 - Nodes, [63](#)
- eDOMFragmentNode
 - Nodes, [63](#)
- eDOMGlyphNode
 - Nodes, [63](#)
- eDOMGlyphsNode
 - Nodes, [63](#)
- eDOMGroupNode
 - Nodes, [63](#)
- eDOMJobTkContentNode
 - Nodes, [63](#)
- eDOMJobTkGenericCharacterDataNode
 - Nodes, [63](#)
- eDOMJobTkGenericNodeNode
 - Nodes, [63](#)
- eDOMJobTkNodeNode
 - Nodes, [63](#)
- eDOMJobTkValueNode
 - Nodes, [63](#)
- eDOMNodeType
 - Nodes, [63](#)
- eDOMNodeTypeCnt
 - Nodes, [63](#)
- eDOMOutlineEntryNode
 - Nodes, [63](#)
- eDOMOutlineNode
 - Nodes, [63](#)
- eDOMPageNode
 - Nodes, [63](#)
- eDOMPathNode
 - Nodes, [63](#)
- eDOMRefNode
 - Nodes, [63](#)
- eDOMTileNode
 - Nodes, [63](#)
- eDOMTransparencyGroupNode
 - Nodes, [63](#)
- eDOMVisualRootNode
 - Nodes, [63](#)
- eDuplexBindingMode
 - JawsMako::IPJLParser, [1364](#)
- eEditingAllowed
 - PDF-specific, [72](#)
- eEFJPEG
 - JawsMako::IImageEncoderTransform, [1046](#)
- eEncodeFormat
 - JawsMako::IImageEncoderTransform, [1046](#)

- eEverythingAllowed
 - PDF-specific, [72](#)
- eExternal
 - Outlines, [77](#)
- eExtraBlack
 - JawsMako::ILayoutFontWeight, [1131](#)
- eExtraBold
 - JawsMako::ILayoutFontWeight, [1131](#)
- eExtraLight
 - JawsMako::ILayoutFontWeight, [1131](#)
- effectiveTransformationOfNode
 - IDOMNode, [688](#)
- eFieldType
 - Forms, [86](#)
- eFillRule
 - Geometry, [67](#)
- eFlatCap
 - Geometry, [68](#)
- eFlipX
 - Brushes, [57](#)
- eFlipXY
 - Brushes, [57](#)
- eFlipY
 - Brushes, [57](#)
- eFontProportion
 - JawsMako::ILayoutFont, [1128](#)
- eFontSerifStyle
 - JawsMako::ILayoutFont, [1128](#)
- eFontSourceType
 - Fonts, [59](#)
- eFontSourceTypeFont
 - Fonts, [60](#)
- eFontSourceTypeNone
 - Fonts, [60](#)
- eFontSourceTypeStream
 - Fonts, [60](#)
- eFontSourceTypeStreamFilter
 - Fonts, [60](#)
- eFontStreamFilterType
 - Fonts, [60](#)
- eFontStreamFilterTypeNone
 - Fonts, [60](#)
- eFontStreamFilterTypeObfuscation
 - Fonts, [60](#)
- eFontStyle
 - JawsMako::ILayoutFont, [1128](#)
- eFontType
 - Fonts, [60](#)
- eFontType3
 - Fonts, [60](#)
- eFontTypeOpenType
 - Fonts, [60](#)
- eFontTypePCL5
 - Fonts, [60](#)
- eFontTypePCLXL
 - Fonts, [60](#)
- eFontTypeUnknown
 - Fonts, [60](#)
- eFontWeight
 - JawsMako::ILayoutFontWeight, [1131](#)
- eFormFillingAllowed
 - PDF-specific, [72](#)
- eFPAny
 - JawsMako::ILayoutFont, [1128](#)
- eFPMonoSpaced
 - JawsMako::ILayoutFont, [1128](#)
- eFPProportional
 - JawsMako::ILayoutFont, [1128](#)
- eFREvenOdd
 - Geometry, [67](#)
- eFRNonZero
 - Geometry, [67](#)
- eFSAny
 - JawsMako::ILayoutFont, [1128](#)
- eFSItalic
 - JawsMako::ILayoutFont, [1128](#)
- eFSRegular
 - JawsMako::ILayoutFont, [1128](#)
- eFSSAny
 - JawsMako::ILayoutFont, [1128](#)
- eFSSSansSerif
 - JawsMako::ILayoutFont, [1128](#)
- eFSSSerif
 - JawsMako::ILayoutFont, [1128](#)
- eFTButton
 - Forms, [87](#)
- eFTChoice
 - Forms, [87](#)
- eFTInherited
 - Forms, [87](#)
- eFTSignature
 - Forms, [87](#)
- eFTText
 - Forms, [87](#)
- eGlyphIDNotdef
 - Glyphs, [65](#)
- eGlyphIDSpecial
 - Glyphs, [65](#)
- eHACenter
 - JawsMako::ILayoutParagraph, [1138](#)
- eHAJustified
 - JawsMako::ILayoutParagraph, [1138](#)
- eHALeft
 - JawsMako::ILayoutParagraph, [1138](#)
- eHARight
 - JawsMako::ILayoutParagraph, [1138](#)
- eHighQualityPrintAllowed
 - PDF-specific, [72](#)
- eHorizontalAlignment
 - JawsMako::ILayoutParagraph, [1138](#)
- eCAuto
 - JawsMako::IDistiller, [284](#)
- eCCBased
 - Color space, [103](#)
- eCCCITT
 - JawsMako::IDistiller, [284](#)

- eICDCT
 - JawsMako::IDistiller, 284
 - JawsMako::IPDFOutput, 1320
- eICFlate
 - JawsMako::IDistiller, 284
- eICFlatePredict
 - JawsMako::IDistiller, 284
- eICLZW
 - JawsMako::IDistiller, 284
- eICNone
 - JawsMako::IDistiller, 284
- eICRLE
 - JawsMako::IPDFOutput, 1320
- eIDAverage
 - JawsMako::IDistiller, 284
- eIDBicubic
 - JawsMako::IDistiller, 284
- eIDNone
 - JawsMako::IDistiller, 284
- eIDSubsample
 - JawsMako::IDistiller, 284
- elImage
 - Brushes, 56
- elImageCompression
 - JawsMako::IDistiller, 284
 - JawsMako::IPDFOutput, 1319
- elImageDownsamplingMethod
 - JawsMako::IDistiller, 284
- eIndexed
 - Color space, 103
- eInternal
 - Outlines, 77
- elsComplex
 - Geometry, 67
- eJpegQuality
 - JawsMako::IDistiller, 284
- eJQHigh
 - JawsMako::IDistiller, 285
- eJQLow
 - JawsMako::IDistiller, 285
- eJQMedium
 - JawsMako::IDistiller, 285
- eJQUser
 - JawsMako::IDistiller, 285
- eLAB
 - Color space, 103
- eLabelStyle
 - JawsMako::IPageLabel, 1232
- eLetterLowercase
 - JawsMako::IPageLabel, 1233
- eLetterUppercase
 - JawsMako::IPageLabel, 1233
- eLight
 - JawsMako::ILayoutFontWeight, 1131
- eLinearGradient
 - Brushes, 56
- eMasked
 - Brushes, 56
- eMedium
 - JawsMako::ILayoutFontWeight, 1131
- eMetadataTypeCnt
 - Metadata, 79
- eMiterJoin
 - Geometry, 68
- enablePSInput
 - JawsMako::IJawsMako, 1107
- enableUnencapsulatedMode
 - JawsMako::IPCL5Input, 1248
 - JawsMako::IPCLXLInput, 1261
 - JawsMako::IPSInput, 1389
- encode
 - IDOMJPEGImage, 662, 663
 - IDOMPNGImage, 789, 790
 - IDOMTIFFImage, 916, 917
- encryptMetadata
 - IDOMStandardPDFSecurityInfo, 903
- end
 - JawsMako::IPDFDictionary, 1282
- endEnumeration
 - IDOMGlyphIDEnumerator, 523
- eNodeDirtyFlag
 - IDOMNodeFlags, 698
- eNodeInterestingFlag
 - IDOMNodeFlags, 698
- eNodeRenderFlag
 - IDOMNodeFlags, 698
- eNodeSelectedFlag
 - IDOMNodeFlags, 698
- eNodeUnpackFlag
 - IDOMNodeFlags, 698
- eNodeUserFlag
 - IDOMNodeFlags, 698
- eNone
 - JawsMako::IPageLabel, 1233
- eNormal
 - JawsMako::ILayoutFontWeight, 1131
- eNoSpread
 - Brushes, 57
- eNoTile
 - Brushes, 57
- eNull
 - Brushes, 56
- eObfuscate
 - Fonts, 61
- eof
 - IInputStream, 1102
- eOpenTypeFontType
 - Fonts, 60
- eOpenTypeFontTypeCFF
 - Fonts, 60
- eOpenTypeFontTypeTTC
 - Fonts, 60
- eOpenTypeFontTypeTTF
 - Fonts, 60
- eOpenTypeFontTypeUnknown
 - Fonts, 60

- eOperation
 - Fonts, [60](#)
- eOperationTypes
 - Geometry, [67](#)
- eOriginalFontType
 - Fonts, [61](#)
- eOriginalPclTrueType
 - Fonts, [61](#)
- eOriginalType1
 - Fonts, [61](#)
- eOriginalType11
 - Fonts, [61](#)
- eOriginalType2
 - Fonts, [61](#)
- eOriginalType42
 - Fonts, [61](#)
- eOriginalType9
 - Fonts, [61](#)
- eOriginalTypeOpenType
 - Fonts, [61](#)
- ePad
 - Brushes, [57](#)
- ePage
 - Outlines, [77](#)
- ePageRect
 - Outlines, [77](#)
- ePageView
 - Metadata, [79](#)
- ePAIHDefault
 - JawsMako::IPDFOutput, [1320](#)
- ePAIHFailIfAlternatesFound
 - JawsMako::IPDFOutput, [1320](#)
- ePAIHFailIfDefaultForPrintingFound
 - JawsMako::IPDFOutput, [1320](#)
- ePCLXLFontType
 - Fonts, [61](#)
- ePCLXLFontTypeBitmap
 - Fonts, [61](#)
- ePCLXLFontTypeTTF
 - Fonts, [61](#)
- ePDF1_3
 - JawsMako::IDistiller, [285](#)
 - JawsMako::IPDFOutput, [1321](#)
- ePDF1_4
 - JawsMako::IDistiller, [285](#)
 - JawsMako::IPDFOutput, [1321](#)
- ePDF1_5
 - JawsMako::IDistiller, [285](#)
 - JawsMako::IPDFOutput, [1321](#)
- ePDF1_6
 - JawsMako::IDistiller, [285](#)
 - JawsMako::IPDFOutput, [1321](#)
- ePDF1_7
 - JawsMako::IDistiller, [285](#)
 - JawsMako::IPDFOutput, [1322](#)
- ePDF2_0
 - JawsMako::IPDFOutput, [1322](#)
- ePDFA1b
 - JawsMako::IPDFOutput, [1322](#)
- ePDFA2b
 - JawsMako::IPDFOutput, [1322](#)
- ePDFA2u
 - JawsMako::IPDFOutput, [1322](#)
- ePDFA3b
 - JawsMako::IPDFOutput, [1322](#)
- ePdfAlternateImageHandling
 - JawsMako::IPDFOutput, [1320](#)
- ePdfDeviceNHandling
 - JawsMako::IPDFOutput, [1320](#)
- ePdfExtendedGraphicsStateHandling
 - JawsMako::IPDFOutput, [1321](#)
- ePDFInfo
 - Metadata, [79](#)
- ePDFObjectType
 - PDF Objects, [75](#)
- ePdfOptionalContentHandling
 - JawsMako::IPDFOutput, [1321](#)
- ePDFUA
 - JawsMako::IPDFOutput, [1322](#)
- ePDFVersion
 - JawsMako::IDistiller, [285](#)
 - JawsMako::IPDFOutput, [1321](#)
- ePDFX1a
 - JawsMako::IPDFOutput, [1322](#)
- ePDFX4
 - JawsMako::IPDFOutput, [1322](#)
- ePDNFailOnMismatchedDeviceNProcessSpace
 - JawsMako::IPDFOutput, [1320](#)
- ePDNFailOnNonUtf8ColorantName
 - JawsMako::IPDFOutput, [1320](#)
- ePDNHDefault
 - JawsMako::IPDFOutput, [1320](#)
- ePDNHFailIfBetterColorantInformationFoundAfterFirstUse
 - JawsMako::IPDFOutput, [1320](#)
- ePDNHFailIfDotGainFound
 - JawsMako::IPDFOutput, [1320](#)
- ePDNHFailOnInconsistentColorantInformation
 - JawsMako::IPDFOutput, [1320](#)
- ePDNHFailOnMissingColorantInformation
 - JawsMako::IPDFOutput, [1320](#)
- ePEGSHDefault
 - JawsMako::IPDFOutput, [1321](#)
- ePEGSHFailOnTransferFunction
 - JawsMako::IPDFOutput, [1321](#)
- ePermissionsFlags
 - PDF-specific, [72](#)
- ePjIResult
 - JawsMako::IPJLParser, [1364](#)
- ePOCHDefault
 - JawsMako::IPDFOutput, [1321](#)
- ePOCHFailIfAutoStateFound
 - JawsMako::IPDFOutput, [1321](#)
- ePOCHFailOnInconsistentConfigurationName
 - JawsMako::IPDFOutput, [1321](#)
- ePOCHFailOnInvalidOrder
 - JawsMako::IPDFOutput, [1321](#)

- ePOTArray
 - PDF Objects, [75](#)
- ePOTBoolean
 - PDF Objects, [75](#)
- ePOTDictionary
 - PDF Objects, [75](#)
- ePOTFarReference
 - PDF Objects, [75](#)
- ePOTInteger
 - PDF Objects, [75](#)
- ePOTName
 - PDF Objects, [75](#)
- ePOTNull
 - PDF Objects, [75](#)
- ePOTOperator
 - PDF Objects, [75](#)
- ePOTReal
 - PDF Objects, [75](#)
- ePOTReference
 - PDF Objects, [75](#)
- ePOTStream
 - PDF Objects, [75](#)
- ePOTString
 - PDF Objects, [75](#)
- ePREndOfFile
 - JawsMako::IPJLParser, [1364](#)
- ePREnterHpgl2
 - JawsMako::IPJLParser, [1364](#)
- ePREnterPcl
 - JawsMako::IPJLParser, [1364](#)
- ePREnterPclXI
 - JawsMako::IPJLParser, [1364](#)
- ePREnterPdf
 - JawsMako::IPJLParser, [1364](#)
- ePREnterPostScript
 - JawsMako::IPJLParser, [1364](#)
- ePrintAllowed
 - PDF-specific, [72](#)
- ePublicKeyPermissionsFlags
 - IDOMPublicKeyPDFSecurityInfo, [812](#)
- equal
 - CClassID, [112](#)
 - CTransformMatrix< TItem >, [140](#)
- equals
 - IDOMColorSpace, [383](#)
 - JawsMako::IAnnotationReference, [236](#)
 - JawsMako::ILayoutFontWeight, [1133](#)
- eRadialGradient
 - Brushes, [56](#)
- eReflect
 - Brushes, [57](#)
- eRepeat
 - Brushes, [57](#)
- eRomanLowercase
 - JawsMako::IPageLabel, [1233](#)
- eRomanUppercase
 - JawsMako::IPageLabel, [1233](#)
- eRoundCap
 - Geometry, [68](#)
- eRoundJoin
 - Geometry, [68](#)
- Error handling, [49](#)
 - EDL_ERR_ABORTED, [50](#)
 - EDL_ERR_ADD_OBJECT_RES_DICT, [52](#)
 - EDL_ERR_APPEND_OUTLINE_TO_CLOSED_NODE, [53](#)
 - EDL_ERR_BAD_ARGUMENTS, [50](#)
 - EDL_ERR_BAD_BRUSH, [51](#)
 - EDL_ERR_BAD_COLOR_SPACE, [53](#)
 - EDL_ERR_BAD_DIMENSIONS, [51](#)
 - EDL_ERR_BAD_FONT, [51](#)
 - EDL_ERR_BAD_FONT_MAP, [50](#)
 - EDL_ERR_BAD_GEOMETRY, [51](#)
 - EDL_ERR_BOUNDS_CALCULATION_FAILED, [51](#)
 - EDL_ERR_BROKEN_ELEMENT_SEQUENCE, [52](#)
 - EDL_ERR_COLOR_CONVERSION_FAILURE, [53](#)
 - EDL_ERR_COMPRESS, [50](#)
 - EDL_ERR_CONFIG, [50](#)
 - EDL_ERR_CORRUPT_PNG_IMAGE, [50](#)
 - EDL_ERR_DEFLATE, [52](#)
 - EDL_ERR_DELETE_OUTLINE_WITH_CLOSED_PARENT, [53](#)
 - EDL_ERR_DEVICE_OUTOFMEMORY, [52](#)
 - EDL_ERR_DICT_ITEM_PRESENT, [52](#)
 - EDL_ERR_DOM_CREATION_FAILED, [51](#)
 - EDL_ERR_DTD_CONTENT, [51](#)
 - EDL_ERR_DUPLICATE_URI, [52](#)
 - EDL_ERR_FAILS_TO_ADD_OUTLINE_TO_METADATA_STORAGE, [53](#)
 - EDL_ERR_FALLBACK_BEFORE_CHOICE, [52](#)
 - EDL_ERR_FILENOTFOUND, [50](#)
 - EDL_ERR_IMAGE_DECODE_FAILURE, [53](#)
 - EDL_ERR_IMAGE_ENCODE_FAILURE, [53](#)
 - EDL_ERR_IMAGE_PARAMETER_OUT_OF_RANGE, [53](#)
 - EDL_ERR_IMMUTABLE_GEOMETRY, [51](#)
 - EDL_ERR_INCOMPATIBLE_IMAGE_PARAMETER, [53](#)
 - EDL_ERR_INCOMPATIBLE_PDFA, [53](#)
 - EDL_ERR_INCOMPATIBLE_PDFX, [53](#)
 - EDL_ERR_INTERNAL_RIP, [53](#)
 - EDL_ERR_INVALID_ABBR_PATH, [52](#)
 - EDL_ERR_INVALID_ATTRIBUTE, [51](#)
 - EDL_ERR_INVALID_COLOR_SPECIFICATION, [52](#)
 - EDL_ERR_INVALID_DOM_ID, [51](#)
 - EDL_ERR_INVALID_ELEMENT, [52](#)
 - EDL_ERR_INVALID_FONT, [53](#)
 - EDL_ERR_INVALID_GLYPHS, [51](#)
 - EDL_ERR_INVALID_IMAGE, [52](#)
 - EDL_ERR_INVALID_PDF_PASSWORD, [53](#)
 - EDL_ERR_INVALID_THUMBNAIL_TYPE, [52](#)
 - EDL_ERR_INVALID_TYPE3_GLYPH, [51](#)
 - EDL_ERR_INVALID_URI, [52](#)
 - EDL_ERR_INVALID_ZIP, [51](#)
 - EDL_ERR_IOERROR, [50](#)

- EDL_ERR_KEY_SET_NON_RESOURCE, 52
- EDL_ERR_LARGE_FILE, 50
- EDL_ERR_LICENSE, 50
- EDL_ERR_LICENSE_BAD_CONFIG, 53
- EDL_ERR_LICENSE_INVALID, 53
- EDL_ERR_LICENSE_MISSING, 53
- EDL_ERR_MISSING_REQ_ATTRIBUTE, 51
- EDL_ERR_MISSING_RESOURCE, 51
- EDL_ERR_MISSING_XPS_PART, 52
- EDL_ERR_MORE_THAN_ONE_ELEMENT, 52
- EDL_ERR_MORE_THAN_ONE_GROUP_ELEMENT, 52
- EDL_ERR_MULTIPLE_COREPROPERTIES_RELS, 52
- EDL_ERR_MULTIPLE_DOCSTRUCTURE_RELS, 52
- EDL_ERR_MULTIPLE_STARTPARTS_RELS, 52
- EDL_ERR_MULTIPLE_THUMBNAIL_RELS, 52
- EDL_ERR_NAMESPACE_NOT_UNDERSTOOD, 52
- EDL_ERR_NO_COLOR_NAME, 51
- EDL_ERR_NO_CONTENT_TYPE, 51
- EDL_ERR_NO_KEY_RESIDICT_OBJECT, 52
- EDL_ERR_NO_REQ_ELEMENT, 52
- EDL_ERR_NO_REQ_GROUP_ELEMENT, 52
- EDL_ERR_NODE_ERROR, 51
- EDL_ERR_NULL_NODE, 51
- EDL_ERR_NUMBER_REQ_ELEMENT, 52
- EDL_ERR_OPEN_PDF, 53
- EDL_ERR_OPENFORREAD, 50
- EDL_ERR_OPENFORWRITE, 50
- EDL_ERR_OUTOFMEMORY, 50
- EDL_ERR_PAGE_TOO_LARGE, 50
- EDL_ERR_PANIC, 50
- EDL_ERR_PDFOUT_GENERAL_FAILURE, 53
- EDL_ERR_PREFIX_NOT_DEFINED, 52
- EDL_ERR_PREFIXED_ATTRIBUTE, 52
- EDL_ERR_PRESERVED_ITEMS_NOT_DECLARED_IGNOREABLE, 52
- EDL_ERR_PRINTTICKET_EXISTS, 52
- EDL_ERR_PROCESS_CONTENT_NOT_DECLARED_IGNOREABLE, 52
- EDL_ERR_PROCESS_PART, 51
- EDL_ERR_PROPERTY_ALREADY_SET, 51
- EDL_ERR_PSOUT_GENERAL_FAILURE, 53
- EDL_ERR_READ, 50
- EDL_ERR_REMOTERESOURCE_REF, 51
- EDL_ERR_RESTRICTED_FONT, 50
- EDL_ERR_TYPE3_GLYPH_NOT_FOUND, 51
- EDL_ERR_UNDEFINED, 50
- EDL_ERR_UNEXPECTED_RESOURCE, 52
- EDL_ERR_UNICODE, 50
- EDL_ERR_UNPREFIXED_ATTRIBUTE, 52
- EDL_ERR_UNSUPPORTED_GLYPHS, 52
- EDL_ERR_VECTOR_ERROR, 50
- EDL_ERR_WRITE, 50
- EDL_ERR_XML_PARSER, 52
- EDL_ERR_XPS_FROM_ENCRYPTED_SOURCE, 53
- EDL_ERR_XPS_PROVIDER, 51
- EDL_ERR_ZIP_PATH_NOT_FOUND, 53
- EDL_ERR_ZIP_WRITE_ONLY, 53
- EDLErrorCode, 49
- JM_DUPLICATE_FIELD_PARTIAL_NAME, 50
- JM_ERR_ANNOTATION_NOT_FOUND, 50
- JM_ERR_APPEARANCE_NOT_FOUND, 50
- JM_ERR_ASSEMBLY_WRITE_FORBIDDEN, 50
- JM_ERR_ATTEMPTED_WRITE_ON_OPEN_INPUT, 51
- JM_ERR_BAD_CONFIGURATION, 50
- JM_ERR_BAD_UNICODE_CMAP, 51
- JM_ERR_DIRECTORY_DOESNT_EXIST, 50
- JM_ERR_DOCUMENT_NOT_FOUND, 50
- JM_ERR_DUPLICATE_WIDGET, 50
- JM_ERR_FONT_NOT_FOUND, 50
- JM_ERR_FORM_FIELD_NOT_FOUND, 50
- JM_ERR_GENERAL, 50
- JM_ERR_IJPCS, 51
- JM_ERR_INCORRECT_PDF_OBJECT_TYPE, 51
- JM_ERR_INFORMATION_NOT_AVAILABLE, 50
- JM_ERR_INVALID_OPTIONAL_CONTENT, 51
- JM_ERR_INVALID_PDF_OBJECT, 51
- JM_ERR_OPTIONAL_CONTENT_GROUP_NOT_FOUND, 51
- JM_ERR_PAGE_NOT_FOUND, 50
- JM_ERR_PCL, 51
- JM_ERR_PCLXL, 51
- JM_ERR_PDF_OBJECT_NOT_FOUND, 51
- JM_ERR_PJL, 51
- JM_ERR_RANGE_ERROR, 50
- JM_ERR_RESOURCE_NOT_FOUND, 51
- JM_ERR_REVERT_FAILED, 50
- JM_ERR_SYMBOLSET_NOT_FOUND, 51
- JM_ERR_TARGET_NOT_FOUND, 50
- JM_ERR_TOO_MANY_PDFOUT_WRITERS, 50
- JM_ERR_TTF_INSTRUCTIONS, 51
- JM_ERR_UNSUPPORTED, 50
- JM_ERR_UNSUPPORTED_WIDGET_NOT_FOUND, 50
- escRGB
 - Color space, 103
- eSCRgbLinearInterpolation
 - Brushes, 56
- eSemiBold
 - JawsMako::ILayoutFontWeight, 1131
- eSemiLight
 - JawsMako::ILayoutFontWeight, 1131
- esGray
 - Color space, 103
- eSoftMask
 - Brushes, 56
- eSolidColor
 - Brushes, 56
- eSpreadMethod
 - Brushes, 57
- eSquareCap

- Geometry, [68](#)
- esRGB
 - Color space, [103](#)
- eSRgbLinearInterpolation
 - Brushes, [56](#)
- eStrokeLineCap
 - Geometry, [67](#)
- eStrokeLineJoin
 - Geometry, [68](#)
- eStrokeMiterLimitTreatment
 - Geometry, [68](#)
- eSweepDirection
 - Geometry, [68](#)
- eTargetType
 - Outlines, [77](#)
- eTCNone
 - IDOMTIFFImage, [915](#)
- eTextStyle
 - Outlines, [78](#)
- eTextStyleBold
 - Outlines, [78](#)
- eTextStyleBoldItalic
 - Outlines, [78](#)
- eTextStyleItalic
 - Outlines, [78](#)
- eTextStyleNone
 - Outlines, [78](#)
- eTHColor
 - JawsMako::IDistiller, [285](#)
- eThin
 - JawsMako::ILayoutFontWeight, [1131](#)
- eTHMono
 - JawsMako::IDistiller, [285](#)
- eTHNone
 - JawsMako::IDistiller, [285](#)
- eThumbnailType
 - JawsMako::IDistiller, [285](#)
- eTIFFCompression
 - IDOMTIFFImage, [914](#)
- eTile
 - Brushes, [57](#)
- eTilingMode
 - Brushes, [57](#)
- eTilingPattern
 - Brushes, [56](#)
- eTransferFunctionMethod
 - JawsMako::IDistiller, [285](#)
- eTRApply
 - JawsMako::IDistiller, [286](#)
- eTriangleCap
 - Geometry, [68](#)
- eTRPreserve
 - JawsMako::IDistiller, [286](#)
- eTRRemove
 - JawsMako::IDistiller, [286](#)
- eType
 - Metadata, [79](#)
- eType1ShadingPattern
 - Brushes, [56](#)
- eType2ShadingPattern
 - Brushes, [56](#)
- eType3ShadingPattern
 - Brushes, [56](#)
- eType4567ShadingPattern
 - Brushes, [56](#)
- evaluate
 - IDOMFunction, [507](#)
 - JawsMako::IOptionalContentVisibilityExpression, [1195](#)
- eVEOAnd
 - JawsMako::IOptionalContentVisibilityExpression, [1195](#)
- eVEOGroupState
 - JawsMako::IOptionalContentVisibilityExpression, [1195](#)
- eVEONot
 - JawsMako::IOptionalContentVisibilityExpression, [1195](#)
- eVEOR
 - JawsMako::IOptionalContentVisibilityExpression, [1195](#)
- eViewerPreferences
 - Metadata, [79](#)
- eViewUnits
 - Brushes, [57](#)
- eVisibilityExpressionOperation
 - JawsMako::IOptionalContentVisibilityExpression, [1195](#)
- eVisibilityPolicy
 - JawsMako::IOptionalContentDetails, [1182](#)
- eVisual
 - Brushes, [56](#)
- eVPAIOff
 - JawsMako::IOptionalContentDetails, [1183](#)
- eVPAIOn
 - JawsMako::IOptionalContentDetails, [1183](#)
- eVPAAnyOff
 - JawsMako::IOptionalContentDetails, [1183](#)
- eVPAAnyOn
 - JawsMako::IOptionalContentDetails, [1183](#)
- eXmpContainer_Alt
 - Metadata, [80](#)
- eXmpContainer_Bag
 - Metadata, [80](#)
- eXmpContainer_None
 - Metadata, [80](#)
- eXmpContainer_Seq
 - Metadata, [80](#)
- eXmpContainerType
 - Metadata, [79](#)
- extract
 - JawsMako::IPDFPageExtractor, [1344](#)
- extractChild
 - IDOMNode, [689](#)
- extractFrom
 - JawsMako::IPDFPageExtractor, [1344](#)

- extractTo
 - JawsMako::IPDFPageExtractor, [1345](#)
- fieldInSubtree
 - JawsMako::IFormField, [1017](#)
- fieldInTree
 - JawsMako::IForm, [1004](#)
- findChild
 - IDOMJobTk, [618](#), [620](#)
 - IDOMJobTkContent, [629](#)
 - IDOMJobTkNode, [648](#)
- findChildrenOfType
 - IDOMNode, [689](#)
- findFont
 - JawsMako::IJawsMako, [1107](#), [1108](#)
 - JawsMako::ILayoutFont, [1129](#)
- findFonts
 - JawsMako::ILayoutFont, [1130](#)
- findNamedClass
 - IEDLClassFactory, [965](#)
- findNamespaceByName
 - IDOMJobTkContent, [629](#)
- findNamespaceByPrefix
 - IDOMJobTkContent, [630](#)
- flushCaches
 - JawsMako::ISkiaRenderer, [1452](#)
- font
 - JawsMako::IPDFInput::CPdfFontInfo, [127](#)
- Fonts, [58](#)
 - eDeobfuscate, [61](#)
 - eFontSourceType, [59](#)
 - eFontSourceTypeFont, [60](#)
 - eFontSourceTypeNone, [60](#)
 - eFontSourceTypeStream, [60](#)
 - eFontSourceTypeStreamFilter, [60](#)
 - eFontStreamFilterType, [60](#)
 - eFontStreamFilterTypeNone, [60](#)
 - eFontStreamFilterTypeObfuscation, [60](#)
 - eFontType, [60](#)
 - eFontType3, [60](#)
 - eFontTypeOpenType, [60](#)
 - eFontTypePCL5, [60](#)
 - eFontTypePCLXL, [60](#)
 - eFontTypeUnknown, [60](#)
 - eObfuscate, [61](#)
 - eOpenTypeFontType, [60](#)
 - eOpenTypeFontTypeCFF, [60](#)
 - eOpenTypeFontTypeTTC, [60](#)
 - eOpenTypeFontTypeTTF, [60](#)
 - eOpenTypeFontTypeUnknown, [60](#)
 - eOperation, [60](#)
 - eOriginalFontType, [61](#)
 - eOriginalPclTrueType, [61](#)
 - eOriginalType1, [61](#)
 - eOriginalType11, [61](#)
 - eOriginalType2, [61](#)
 - eOriginalType42, [61](#)
 - eOriginalType9, [61](#)
 - eOriginalTypeOpenType, [61](#)
 - ePCLXLFontType, [61](#)
 - ePCLXLFontTypeBitmap, [61](#)
 - ePCLXLFontTypeTTF, [61](#)
- forceGroupState
 - JawsMako::IOptionalContent, [1176](#)
- Form XObjects, [61](#)
- Forms, [86](#)
 - eFieldType, [86](#)
 - eFTButton, [87](#)
 - eFTChoice, [87](#)
 - eFTInherited, [87](#)
 - eFTSignature, [87](#)
 - eFTText, [87](#)
- fromPDFDate
 - IEDLTime, [984](#)
- fromW3CDTF
 - IEDLTime, [984](#)
- generateSVG
 - JawsMako::ISVGGenerator, [1480](#), [1481](#)
- generateXAML
 - JawsMako::IXAMLGenerator, [1537](#)
- generateXAMLForAppearance
 - JawsMako::IXAMLGenerator, [1537](#), [1538](#)
- generateXAMLForPageAndAnnotationAppearances
 - JawsMako::IXAMLGenerator, [1538](#)
- generateXAMLForPageAnnotationAppearances
 - JawsMako::IXAMLGenerator, [1539](#)
- generateXMLForDocument
 - JawsMako::IAnnotationUtils, [237](#)
- Generators, [83](#)
- Geometry, [65](#)
 - eBevelJoin, [68](#)
 - eBevelLongMiters, [68](#)
 - eClipLongMiters, [68](#)
 - eDoesRotate, [67](#)
 - eDoesScale, [67](#)
 - eDoesTranslate, [67](#)
 - eFillRule, [67](#)
 - eFlatCap, [68](#)
 - eFREvenOdd, [67](#)
 - eFRNonZero, [67](#)
 - eIsComplex, [67](#)
 - eMiterJoin, [68](#)
 - eOperationTypes, [67](#)
 - eRoundCap, [68](#)
 - eRoundJoin, [68](#)
 - eSquareCap, [68](#)
 - eStrokeLineCap, [67](#)
 - eStrokeLineJoin, [68](#)
 - eStrokeMiterLimitTreatment, [68](#)
 - eSweepDirection, [68](#)
 - eTriangleCap, [68](#)
- get
 - IColorManager, [263](#)
 - IDOMNodeFlags, [698](#)
 - IDOMResourceDictionary, [840](#)
 - JawsMako::IPDFArray, [1273](#)
 - JawsMako::IPDFDictionary, [1282](#), [1283](#)

- getAbort
 - JawsMako::IProgressMonitor, [1379](#)
- getActionOrDest
 - JawsMako::ILinkAnnotation, [1159](#)
- getActionsCount
 - IDOMActionArray, [321](#)
- getActionsDictionary
 - JawsMako::IWidgetAnnotation, [1521](#)
- getActionsEnum
 - IDOMActionArray, [321](#)
- getAdditionalActionsDictionary
 - JawsMako::IWidgetAnnotation, [1521](#)
- getAdjustedForUseInTransformedNode
 - IDOMBrush, [337](#)
- getAdvanceX
 - IDOMGlyph, [515](#)
- getAdvanceY
 - IDOMGlyph, [515](#)
- getAllocatorId
 - JawsMako::CPDFFarReference, [125](#)
- getAlpha
 - IDOMColor, [377](#)
- getAlphaDetails
 - IDOMPDFImage, [783](#)
- getAlternateCaption
 - JawsMako::IWidgetAppearanceCharacteristics, [1530](#)
- getAlternateColorSpace
 - IDOMColorSpaceDeviceN, [399](#)
 - IDOMColorSpaceICCBased, [408](#)
 - IDOMDeviceNColorant, [433](#)
- getAlternateImages
 - IDOMImageBrush, [587](#)
- getAntiAlias
 - IDOMShadingPatternBrush, [849](#)
- getAppearance
 - JawsMako::IAnnotation, [227](#)
- getAppearanceCharacteristics
 - JawsMako::IWidgetAnnotation, [1521](#)
- getAppearances
 - JawsMako::IAnnotation, [227](#)
- getAsDOMImage
 - JawsMako::IPageRaster, [1243](#)
- getAsGeometry
 - IDOMShape, [887](#)
- getAsImage
 - IDOMShape, [887](#)
- getAttributeAtIndex
 - IDOMJobTkGenericNode, [642](#)
- getAttributes
 - JawsMako::IPJLParser, [1364](#)
 - JawsMako::IStructureElement, [1471](#)
- getAuthor
 - JawsMako::IMarkupAnnotation, [1168](#)
- getAutomationPropertiesHelpText
 - IDOMCanvas, [349](#)
 - IDOMPathNode, [752](#)
- getAutomationPropertiesName
 - IDOMCanvas, [349](#)
 - IDOMPathNode, [753](#)
- getBackdropColor
 - IDOMSoftMaskBrush, [893](#)
- getBackgroundColor
 - IDOMShadingPatternBrush, [849](#)
 - JawsMako::IWidgetAppearanceCharacteristics, [1531](#)
- getBBox
 - IDOMShadingPatternBrush, [849](#)
 - IDOMTilingPatternBrush, [921](#)
 - IDOMType3Font, [946](#)
- getBidiLevel
 - IDOMGlyphs, [533](#)
- getBitsPerComponent
 - IDOMPDFImage, [783](#)
 - IDOMShadingPatternType4567Brush, [879](#)
- getBitsPerCoordinate
 - IDOMShadingPatternType4567Brush, [879](#)
- getBitsPerFlag
 - IDOMShadingPatternType4567Brush, [879](#)
- getBitsPerSample
 - IDOMSampledFunction, [843](#)
- getBlackPoint
 - IDOMColorSpaceLAB, [416](#)
- getBlackPointCompensation
 - IDOMGlyphs, [533](#)
 - IDOMPathNode, [753](#)
 - IDOMTransparencyGroup, [933](#)
- getBleedBox
 - IDOMFixedPage, [451](#)
- getBlendMode
 - IDOMFormInstance, [503](#)
 - IDOMGlyphs, [533](#)
 - IDOMPathNode, [753](#)
 - IDOMTransparencyGroup, [933](#)
- getBoolProperty
 - IDOMImageProperties, [608](#)
- getBorder
 - JawsMako::IAnnotation, [227](#)
- getBorderColor
 - JawsMako::IWidgetAppearanceCharacteristics, [1531](#)
- getBottom
 - IDOMPageRectTarget, [715](#)
- getBounds
 - IDOMGlyph, [516](#)
 - IDOMNode, [689](#)
 - IDOMPathFigure, [725](#)
 - IDOMPathGeometry, [733](#)
 - IDOMPathSegment, [772](#)
 - IDOMShape, [887](#)
- getBoundsOnPage
 - JawsMako::ITextRun, [1493](#)
- getBoundsVector
 - IDOMStitchingFunction, [908](#)
- getBox
 - JawsMako::IPDFArray, [1273](#)

- getBPS
 - ImageFrame, 1051
- getBrush
 - IDOMMaskedBrush, 675
- getBrushType
 - IDOMBrush, 338
- getCalculator
 - IDOMPostScriptCalculatorFunction, 804
- getCalculatorAsPostScriptStream
 - IDOMPostScriptCalculatorFunction, 804
- getCalloutLine
 - JawsMako::IFreeTextAnnotation, 1030
- getCanonicalSourceFilePath
 - IInputStream, 1102
- getCaption
 - JawsMako::IWidgetAppearanceCharacteristics, 1531
- getCaptionOffset
 - JawsMako::ILineAnnotation, 1153
- getCaretStops
 - IDOMGlyphs, 533
- getCenter
 - IDOMRadialGradientBrush, 818
- getCharacterData
 - IDOMJobTkGenericCharacterData, 637
- getCharacterMap
 - IDOMFont, 459
- getCharPathType
 - IDOMCharPathGroup, 363
- getChecksum
 - JawsMako::IOutputIntent, 1202
- getChildFields
 - JawsMako::IFormField, 1017
- getChildValue
 - IDOMJobTkNode, 648
- getChildWidgets
 - JawsMako::IFormField, 1017
- getCidMap
 - IDOMFontOpenType, 466
- getClassID
 - IEDLObject, 972
- getClip
 - IDOMGlyphs, 534
 - IDOMGroup, 563
 - IDOMPathNode, 753
- getClippedGroup
 - IDOMCharPathGroup, 363
- getClusters
 - IDOMGlyphs, 534
- getCMYKSWOPProfile
 - IColorManager, 263
- getColor
 - IDOMGradientStop, 557
 - IDOMSolidColorBrush, 901
 - JawsMako::IAnnotation, 227
 - JawsMako::ILayoutTextRun, 1148
- getColorant
 - IDOMColorSpaceDeviceN, 399
- getColorantInfo
 - IDOMColorSpaceDeviceN, 400
- getColorantName
 - IDOMColorSpace, 383
- getColorized
 - IDOMGlyph, 516
- getColorizedTilingPatternBrush
 - IDOMTilingPatternBrush, 921
- getColorInterpolationMode
 - IDOMGradientBrush, 552
- getColorKey
 - IDOMPDFImage, 783
- getColorSpace
 - IDOMColor, 377
 - IDOMPDFImage, 783
 - IDOMShadingPatternBrush, 849
 - IDOMTransparencyGroup, 933
 - ImageFrame, 1051
- getColorSpaceType
 - IDOMColorSpace, 383
- getCombinedContent
 - IDOMJobTk, 620
- getComments
 - JawsMako::IPSCCommentMonitor, 1384
- GetComponentRange
 - IDOMColorSpace, 384
- getComponentsHaveSameRange
 - IDOMColorSpace, 384
- GetComponentValue
 - IDOMColor, 378
- getContent
 - IDOMJobTk, 620
 - JawsMako::IPage, 1224
- getContentBox
 - IDOMFixedPage, 451
- getContents
 - JawsMako::IAnnotation, 228
- getCount
 - IDOMCatalog, 355
- getCreationTime
 - JawsMako::IMarkupAnnotation, 1168
- getCropBox
 - IDOMFixedPage, 452
- getDataSource
 - IDOMShadingPatternType4567Brush, 879
- getDay
 - IEDLTime, 985
- getDecode
 - IDOMPDFImage, 784
 - IDOMShadingPatternType4567Brush, 879
- getDecodeParameters
 - IDOMPDFImage, 784
- getDecodeParametersAtIndex
 - IDOMPDFImage, 784
- getDefaultAppearanceString
 - JawsMako::IFormField, 1018
 - JawsMako::IWidgetAnnotation, 1521
- getDefaultBlackPointCompensation

- IColorManager, 264
- getDefaultColor
 - JawsMako::ILayoutParagraph, 1140
- getDefaultFont
 - JawsMako::ILayoutParagraph, 1140
- getDefaultFontIndex
 - JawsMako::ILayoutParagraph, 1140
- getDefaultFontSize
 - JawsMako::ILayoutParagraph, 1141
- getDefaultForPrinting
 - IDOMPdFAIternateImage, 779
- getDefaultIntentForICCBasedSpace
 - IColorManager, 264
- getDefaultOtherBlackPreservation
 - IColorManager, 264
- getDefaultRenderingIntent
 - IDOMColorSpace, 384
- getDefaultTextBlackPreservation
 - IColorManager, 264
- getDefaultValue
 - JawsMako::IFormField, 1018
 - JawsMako::IWidgetAnnotation, 1522
- getDescription
 - IDOMOutlineEntry, 707
- getDeviceCMYKIntercept
 - IColorManager, 265
- getDeviceFontName
 - IDOMGlyphs, 534
- getDeviceGrayIntercept
 - IColorManager, 265
- getDeviceLinkICCPProfile
 - IColorManager, 265
- getDeviceRGBIntercept
 - IColorManager, 266
- getDocument
 - JawsMako::IDocumentAssembly, 316
- getDocumentId
 - JawsMako::CPDFFarReference, 126
- getDomain
 - IDOMShadingPatternType1Brush, 856
 - IDOMShadingPatternType2Brush, 862
 - IDOMShadingPatternType3Brush, 869
- getDOMid
 - IDOMNode, 690
- getDotGainFunction
 - IDOMDeviceNColorant, 433
- getEdgeMode
 - IDOMCanvas, 349
 - IDOMPathNode, 754
- getEfficientlySkippable
 - IImageFrame, 1051
- getEmbedded
 - IDOMFontOpenType, 467
- getEmbeddedPSCSAForICCBasedSpace
 - IColorManager, 266
- getEncodeVector
 - IDOMStitchingFunction, 909
- getEndCircleCenter
 - IDOMShadingPatternType3Brush, 869
- getEndCircleRadius
 - IDOMShadingPatternType3Brush, 870
- getEndPoint
 - IDOMLinearGradientBrush, 668
 - IDOMPathSegment, 772
 - IDOMShadingPatternType2Brush, 862
- getEnumerationCount
 - IFontPCL5WriteSegmentBlockEnumerator, 992
- getEnumerationItemBlockSize
 - IFontPCL5WriteSegmentBlockEnumerator, 993
- getEquivalentPath
 - IDOMGlyphs, 535
- getEquivalentSimpleBrush
 - IDOMShadingPatternType2Brush, 862
 - IDOMShadingPatternType3Brush, 870
- getEquivalentSimpleVisualBrush
 - IDOMVisualBrush, 953
- getEquivalentTilingBrush
 - IDOMImageBrush, 587
 - IDOMVisualBrush, 953
- getEquivalentVisualBrush
 - IDOMImageBrush, 587
 - IDOMTilingPatternBrush, 922
- getEquivalentXPSBrush
 - IDOMMaskedBrush, 675
- getExpanded
 - IDOMOutlineEntry, 708
- getExponent
 - IDOMExponentialFunction, 436
- getExtend
 - IDOMShadingPatternType2Brush, 863
 - IDOMShadingPatternType3Brush, 870
- getExtraChannelType
 - IImageFrame, 1051
- getFactory
 - ISession, 1443
- getFarReferenceForReference
 - JawsMako::IPDFObjectStore, 1311
- getFieldDefaultValue
 - JawsMako::IForm, 1004
- getFieldFlags
 - JawsMako::IFormField, 1018
 - JawsMako::IWidgetAnnotation, 1522
- getFieldId
 - JawsMako::IFormField, 1019
- getFields
 - JawsMako::IForm, 1004
- getFieldType
 - JawsMako::IFormField, 1019
 - JawsMako::IWidgetAnnotation, 1522
- getFieldValue
 - JawsMako::IForm, 1005
- getFigures
 - IDOMPathGeometry, 733
- getFiguresCount
 - IDOMPathGeometry, 733
- getFiguresImmutable

- IDOMPathGeometry, [734](#)
- getFile
 - IDOMActionLaunch, [323](#)
- getFill
 - IDOMGlyphs, [535](#)
 - IDOMPathNode, [754](#)
- getFillOverprints
 - IDOMGlyphs, [535](#)
 - IDOMPathNode, [754](#)
- getFillRule
 - IDOMPathGeometry, [734](#)
- getFilterAtIndex
 - IDOMFilteredImage, [444](#)
- getFirstChild
 - IDOMNode, [690](#)
- getFirstVisibleOption
 - JawsMako::IWidgetAnnotation, [1523](#)
- getFitType
 - IDOMPageRectTarget, [715](#)
- getFlags
 - IDOMNode, [690](#)
 - JawsMako::IAnnotation, [228](#)
- getFlattenedGeometry
 - IDOMPathGeometry, [734](#)
- getFlattenedGlyphInfo
 - IDOMGlyphs, [535](#)
- getFont
 - IDOMGlyphs, [536](#)
 - JawsMako::ILayoutTextRun, [1149](#)
- getFontBaseStream
 - IDOMFont, [459](#)
- getFontDictionary
 - IDOMType3Font, [946](#)
- getFontHeaderSegmentBlockEnumerator
 - IDOMFontPCLXL, [477](#)
- getFontIndex
 - IDOMGlyphs, [536](#)
 - JawsMako::ILayoutTextRun, [1149](#)
- getFontLicenseFromOS2Table
 - IDOMFontOpenType, [467](#)
- getFontName
 - IDOMFontPCL5, [474](#)
 - IDOMFontPCLXL, [478](#)
- getFontNames
 - JawsMako::IDistiller, [287](#)
- getFontOpenTypeTableAccessor
 - IDOMFontOpenType, [467](#)
- getFontRenderingEmSize
 - IDOMGlyphs, [536](#)
- getFontSource
 - IDOMFont, [460](#)
- getFontSourceType
 - IDOMFontSource, [481](#)
- getFontStreamFilterType
 - IDOMFontSourceStreamFilter, [491](#)
- getFontTrueTypeGlyphAccessor
 - IDOMFontOpenType, [468](#)
 - IDOMFontPCL5, [474](#)
- IDOMFontPCLXL, [478](#)
- getFontType
 - IDOMFont, [460](#)
- getForm
 - IDOMFormInstance, [503](#)
 - JawsMako::IAnnotationAppearance, [234](#)
 - JawsMako::IDocument, [309](#)
- getFormat15FontBlockEnumerator
 - IDOMFontPCL5, [475](#)
- getFormat16FontBlockEnumerator
 - IDOMFontPCL5, [475](#)
- getFormId
 - IDOMForm, [496](#)
- getFrame
 - IImageDecoder, [1037](#)
- getFromPcl
 - IDOMFontSourceFromStream, [483](#)
- getFullName
 - IDOMFontOpenType, [468](#)
- getFunction
 - IDOMShadingPatternBrush, [849](#)
- getFunctionAtIndex
 - IDOMGroupingFunction, [568](#)
 - IDOMStitchingFunction, [909](#)
- getFunctionType
 - IDOMFunction, [508](#)
- getGeneration
 - JawsMako::CPDFFarReference, [126](#)
 - JawsMako::CPDFReference, [128](#)
 - JawsMako::IPDFReference, [1355](#)
- getGlyph
 - IDOMType3Font, [947](#)
- getGlyphID
 - IDOMGlyph, [516](#)
 - IDOMGlyphIDEnumerator, [523](#)
- getGlyphName
 - IDOMGlyph, [517](#)
- getGlyphs
 - IDOMType3Font, [948](#)
- getGlyphUnicode
 - IDOMGlyph, [517](#)
- getGradientOrigin
 - IDOMRadialGradientBrush, [818](#)
- getGradientStops
 - IDOMGradientBrush, [553](#)
- getGroup
 - IDOMSoftMaskBrush, [893](#)
- getHandlerRevision
 - IDOMStandardPDFSecurityInfo, [903](#)
- getHasAlphaChannel
 - IImageFrame, [1051](#)
- getHasCustomAdvance
 - IDOMGlyph, [519](#)
- getHasMask
 - IImageFrame, [1052](#)
- getHasRealColorant
 - IDOMColorSpaceDeviceN, [400](#)
- getHeight

- IDOMFixedPage, [452](#)
- IImageFrame, [1052](#)
- JawsMako::ILayoutImageRun, [1135](#)
- getHighlightMode
 - JawsMako::ILinkAnnotation, [1159](#)
 - JawsMako::IWidgetAnnotation, [1523](#)
- getHorizontalAlignment
 - JawsMako::ILayoutParagraph, [1141](#)
- getICCProfile
 - IDOMColorSpaceICCBased, [408](#)
 - IDOMImageBrush, [588](#)
- getIconName
 - JawsMako::ITextAnnotation, [1487](#)
- getId
 - IDOMType3Font, [948](#)
- getIdByIndex
 - IDOMCatalog, [355](#)
- getIdByNumbers
 - IDOMCatalog, [356](#)
- getIdByURI
 - IDOMCatalog, [356](#)
- getImage
 - JawsMako::ILayoutImageRun, [1136](#)
- getImageBrush
 - IDOMPDFAlternateImage, [779](#)
- getImageFrame
 - IDOMImage, [578](#)
- getImageIndex
 - IDOMTIFFImage, [917](#)
- getImageProperties
 - IDOMImage, [579](#)
- getImageSource
 - IDOMImageBrush, [588](#)
- getImageType
 - IDOMImage, [579](#)
- getImageTypes
 - IDOMPDFImage, [784](#)
- getImageWithSubstitutedColorSpace
 - IDOMImage, [579](#)
- getIncrementalSaves
 - JawsMako::IPDFInput, [1292](#), [1293](#)
- getIndices
 - IDOMGlyphs, [536](#)
- getInkList
 - JawsMako::ILinkAnnotation, [1080](#)
- getInputDomain
 - IDOMFunction, [508](#)
- getInputEncode
 - IDOMSampledFunction, [843](#)
- getInputFontSource
 - IDOMFontSourceObfuscationConverter, [488](#)
 - IDOMFontSourceStreamFilter, [491](#)
- getInteger
 - JawsMako::IPDFArray, [1273](#)
- getIntent
 - JawsMako::IStampAnnotation, [1461](#)
- getInteriorColor
 - JawsMako::ILineAnnotation, [1154](#)
 - JawsMako::IRedactionAnnotation, [1418](#)
 - JawsMako::IShapeAnnotation, [1450](#)
- getInterpolationMethod
 - IDOMSampledFunction, [844](#)
- getIntProperty
 - IDOMImageProperties, [609](#)
- getIsCharPath
 - IDOMGlyphs, [537](#)
- getIsClosed
 - IDOMPathFigure, [725](#)
- getIsDashed
 - IDOMPathNode, [754](#)
- getIsEmpty
 - IDOMShape, [888](#)
- getIsExecutable
 - JawsMako::IPDFObject, [1307](#)
- getIsFilled
 - IDOMPathFigure, [725](#)
- getIsImmutable
 - IDOMPathFigure, [725](#)
 - IDOMPathSegment, [772](#)
- getIsInvisible
 - IDOMGlyphs, [537](#)
- getIsIsolated
 - IDOMTransparencyGroup, [934](#)
- getIsKnockout
 - IDOMTransparencyGroup, [934](#)
- getIsLabColorSpace
 - IDOMColorSpaceICCBased, [408](#)
- getIsLargeArc
 - IDOMArcSegment, [330](#)
- getIsMap
 - IDOMExternalTarget, [439](#)
- getIsNChannel
 - IDOMColorSpaceDeviceN, [400](#)
- getIsPDFStandardFont
 - IDOMFontOpenType, [468](#)
- getIsPSStandardFont
 - IDOMFontOpenType, [469](#)
- getIsRect
 - IDOMPathGeometry, [735](#)
 - IDOMShape, [888](#)
- getIsRendered
 - IDOMImage, [580](#)
- getIsRestricted
 - IDOMFontOpenType, [469](#)
- getIsSideways
 - IDOMGlyphs, [537](#)
- getIsSoftMask
 - IDOMMaskedBrush, [675](#)
- getIsStroked
 - IDOMPathSegment, [772](#)
- getIsVisible
 - JawsMako::IOptionalContentDetails, [1183](#)
- getJobMetadata
 - JawsMako::IDocumentAssembly, [316](#)
- getJobTicket
 - IDOMJobTkOwner, [654](#)

- JawsMako::IDocument, 309
- JawsMako::IDocumentAssembly, 316
- JawsMako::IPage, 1224
- getJobTkContent
 - IDOMJobTkNode, 649
- getJobTkNodeType
 - IDOMJobTkNode, 649
- getKeyAtIndex
 - JawsMako::IPDFDictionary, 1283
- getLanguage
 - IDOMCanvas, 349
 - IDOMFixedPage, 452
 - IDOMGlyphs, 537
 - IDOMOutline, 704
 - IDOMOutlineEntry, 708
 - IDOMPathNode, 755
- getLastChild
 - IDOMNode, 690
- getLayerIndex
 - IDOMPSDImage, 810
- getLeaderLineExtensionsLength
 - JawsMako::ILineAnnotation, 1154
- getLeaderLineLength
 - JawsMako::ILineAnnotation, 1154
- getLeaderLineOffset
 - JawsMako::ILineAnnotation, 1154
- getLeading
 - JawsMako::ILayoutParagraph, 1141
- getLeft
 - IDOMPageRectTarget, 715
- getLetterSpacing
 - JawsMako::ILayoutTextRun, 1149
- getLevel
 - IDOMJobTkContent, 630
- getLineEndpoints
 - JawsMako::ILineAnnotation, 1154
- getLiteMessageHandler
 - ISession, 1443
- getLocalBounds
 - JawsMako::ITextRun, 1493
- getMacParameters
 - IDOMActionLaunch, 323
- getMapDeviceGrayToCMYKBlack
 - IColorManager, 266
- getMappingFunction
 - IDOMColorSpaceIndexed, 412
- getMarkedContentDetails
 - IDOMGroup, 563
- getMatrix
 - IDOMForm, 497
 - IDOMShadingPatternType1Brush, 856
- getMediaBox
 - JawsMako::IPage, 1224
- getMeshEntries
 - IDOMShadingPatternType4567Brush, 880
- getMessageHandler
 - ISession, 1444
- getModificationTime
 - JawsMako::IAnnotation, 228
- getModified
 - IDOMJobTkContent, 630
- getMonth
 - IEDLTime, 985
- getMutableGeometry
 - IDOMPathGeometry, 735
- getName
 - EDLQName, 218
 - IDOMDeviceNColorant, 433
 - JawsMako::INamedDestination, 1173
 - JawsMako::IStampAnnotation, 1461
- getNamespace
 - EDLQName, 218
 - IEDLNamespace, 969
- getNamespaceCollectionEnum
 - IDOMJobTkContent, 630
- getNamespacesCount
 - IDOMJobTkContent, 631
- getNameWithPrefix
 - EDLQName, 219
- getNavigateLink
 - IDOMCanvas, 350
 - IDOMGlyphs, 538
 - IDOMPathNode, 755
- getNeedAppearances
 - JawsMako::IForm, 1005
- getNewWindow
 - IDOMActionLaunch, 324
- getNextChild
 - IDOMNode, 691
- getNextSibling
 - IDOMNode, 691
- getNodeType
 - IDOMNode, 691
- getNumAttributes
 - IDOMJobTkGenericNode, 643
- getNumber
 - JawsMako::IPageLabel, 1233
 - JawsMako::IPDFObject, 1308
- getNumbers
 - JawsMako::IPDFArray, 1274
- getNumChannels
 - IImageFrame, 1052
- getNumComponents
 - IDOMColorSpace, 385
- getNumComponentsForICCBasedSpace
 - IColorManager, 266
- getNumComponentsForICCPProfile
 - IColorManager, 267
- getNumDeviceLinkICCPProfiles
 - IColorManager, 267
- getNumDocuments
 - JawsMako::IDocumentAssembly, 316
- getNumExtraChannels
 - IImageFrame, 1052
- getNumFilters
 - IDOMFilteredImage, 444

- getNumFunctions
 - IDOMGroupingFunction, 568
 - IDOMStitchingFunction, 909
- getNumIncrementalSaves
 - JawsMako::IPDFInput, 1293, 1294
- getNumInputValues
 - IDOMFunction, 508
- getNumOutputValues
 - IDOMFunction, 508
- getNumPages
 - JawsMako::IDocument, 309
 - JawsMako::IPDFPageExtractor, 1345
 - JawsMako::IPDFPageInserter, 1348
- getNumSeparations
 - JawsMako::ISeparator, 1440
- getObfuscated
 - IDOMFontOpenType, 469
- getObject
 - IDOMCatalog, 356
- getObjectNum
 - JawsMako::CPDFFarReference, 126
 - JawsMako::CPDFReference, 128
 - JawsMako::IPDFReference, 1355
- getObjectProperty
 - IDOMImageProperties, 609
- getObjectStore
 - JawsMako::IDocument, 309
 - JawsMako::IPage, 1225
 - JawsMako::IStructureElement, 1471
- getOffset
 - IDOMGradientStop, 557
- getOpacity
 - IDOMBrush, 338
 - IDOMFormInstance, 503
 - IDOMGlyphs, 538
 - IDOMPathNode, 755
 - IDOMTransparencyGroup, 934
 - JawsMako::IMarkupAnnotation, 1169
- getOpacityMask
 - IDOMFormInstance, 504
 - IDOMGlyphs, 538
 - IDOMPathNode, 755
 - IDOMTransparencyGroup, 934
- getOpen
 - JawsMako::IPopupAnnotation, 1372
 - JawsMako::ITextAnnotation, 1487
- getOpenTypeFontType
 - IDOMFontOpenType, 469
- getOptionalContentDetails
 - IDOMGroup, 563
 - IDOMImageBrush, 588
 - IDOMPdFAternateImage, 779
- getOptions
 - JawsMako::IFormField, 1019
 - JawsMako::IWidgetAnnotation, 1523
- getOriginalAdvanceX
 - IDOMGlyph, 519
- getOriginalFontType
 - IDOMFontOpenType, 469
- getOriginalOS2Table
 - IDOMFontOpenType, 470
- getOriginX
 - IDOMGlyphs, 538
- getOriginY
 - IDOMGlyphs, 539
- getOutline
 - JawsMako::IDocument, 309
- getOutlineTree
 - IDOMOutline, 704
- getOutputC0
 - IDOMExponentialFunction, 436
- getOutputC1
 - IDOMExponentialFunction, 437
- getOutputDecode
 - IDOMSampledFunction, 844
- getOutputRange
 - IDOMFunction, 509
- getOverprintMode
 - IDOMGlyphs, 539
 - IDOMPathNode, 756
- getOwner
 - IDOMJobTk, 621
- getPA
 - JawsMako::ILinkAnnotation, 1159
- getPage
 - JawsMako::IDocument, 310
- getPageGroup
 - IDOMFixedPage, 452
- getPageId
 - IDOMPageRectTarget, 715
- getPageLabel
 - JawsMako::IPage, 1225
- getPageRaster
 - JawsMako::IPage, 1225
- getPaintType
 - IDOMTilingPatternBrush, 922
- getParentNode
 - IDOMNode, 691
- getPartialName
 - JawsMako::IFormField, 1019
 - JawsMako::IWidgetAnnotation, 1523
- getPathData
 - IDOMPathNode, 756
- getPathToField
 - JawsMako::IForm, 1005
- getPathToWidget
 - JawsMako::IForm, 1006
- getPatternColor
 - IDOMTilingPatternBrush, 922
- getPatternType
 - IDOMTilingPatternBrush, 922
- getPCL5FontType
 - IDOMFontPCL5, 475
- getPCLXLFontType
 - IDOMFontPCLXL, 478
- getPdfPropertiesDictionary

- IDOMForm, [497](#)
- IDOMImageBrush, [588](#)
- getPoint
 - IDOMArcSegment, [330](#)
- getPoints
 - IDOMPolyBezierSegment, [794](#)
 - IDOMPolyLineSegment, [797](#)
 - IDOMPolyQuadraticBezierSegment, [801](#)
 - JawsMako::IPolyAnnotation, [1368](#)
- getPointsCount
 - IDOMPolyBezierSegment, [794](#)
 - IDOMPolyLineSegment, [797](#)
 - IDOMPolyQuadraticBezierSegment, [801](#)
- getPopupReference
 - JawsMako::IMarkupAnnotation, [1169](#)
- getPos
 - IEDLStream, [975](#)
- getPostScriptCSAForICCBasedSpace
 - IColorManager, [267](#)
- getPostScriptName
 - IDOMFontOpenType, [470](#)
- getPrefix
 - IEDLNamespace, [969](#)
 - JawsMako::IPageLabel, [1233](#)
- getPreviousChild
 - IDOMNode, [692](#)
- getPreviousSibling
 - IDOMNode, [692](#)
- getPrintingOrder
 - IDOMColorSpaceDeviceN, [400](#)
- getProcessColorSpace
 - IDOMColorSpaceDeviceN, [401](#)
- getProcessComponentNames
 - IDOMColorSpaceDeviceN, [401](#)
- getProfileClass
 - IDOMICCProfile, [573](#)
- getProfileClassForICCBasedSpace
 - IColorManager, [268](#)
- getProfileColorSpace
 - IDOMICCProfile, [574](#)
- getProfileColorSpaceForICCBasedSpace
 - IColorManager, [268](#)
- getProfileConnectionSpace
 - IDOMICCProfile, [574](#)
- getProfileConnectionSpaceForICCBasedSpace
 - IColorManager, [268](#)
- getProfileFileSpecifications
 - JawsMako::IOutputIntent, [1202](#)
- getProfileForSpace
 - IColorManager, [268](#)
- getProfileHasSupportForIntent
 - IDOMICCProfile, [574](#)
- getProfileHasSupportForIntentForICCBasedSpace
 - IColorManager, [269](#)
- getProfileName
 - IDOMICCProfile, [574](#)
- getProfileNameForICCBasedSpace
 - IColorManager, [269](#)
- getProfileVersion
 - IDOMICCProfile, [575](#)
- getProfileVersionForICCBasedSpace
 - IColorManager, [270](#)
- getProgressTick
 - JawsMako::IProgressMonitor, [1379](#)
- getProperty
 - IDOMImageProperties, [609](#)
 - IDOMMetadata, [682](#)
 - IDOMNode, [692](#)
- getPropertyCollectionEnum
 - IDOMMetadata, [683](#)
 - IDOMNode, [693](#)
- getQName
 - IDOMJobTkGenericNode, [643](#)
 - IDOMJobTkNode, [649](#)
- getQNameAsString
 - IDOMJobTkNode, [649](#)
- getQuadding
 - JawsMako::IFormField, [1020](#)
 - JawsMako::IWidgetAnnotation, [1524](#)
- getQuadPoints
 - JawsMako::ILinkAnnotation, [1159](#)
 - JawsMako::IRedactionAnnotation, [1418](#)
 - JawsMako::ITextMarkupAnnotation, [1491](#)
- getRadiusX
 - IDOMArcSegment, [331](#)
 - IDOMRadialGradientBrush, [818](#)
- getRadiusY
 - IDOMArcSegment, [331](#)
 - IDOMRadialGradientBrush, [819](#)
- getRangeAB
 - IDOMColorSpaceLAB, [416](#)
- getRawBytesPerRow
 - IImageFrame, [1053](#)
- getRect
 - JawsMako::IAnnotation, [228](#)
- getRectInset
 - JawsMako::ICaretAnnotation, [243](#)
 - JawsMako::IFreeTextAnnotation, [1030](#)
 - JawsMako::IShapeAnnotation, [1450](#)
- getRefCount
 - IRCOobject, [1414](#)
- getReference
 - JawsMako::IAnnotation, [228](#)
- getRenderingIntent
 - IDOMGlyphs, [539](#)
 - IDOMPathNode, [756](#)
 - IDOMTransparencyGroup, [934](#)
- getRenderTransform
 - IDOMFormInstance, [504](#)
 - IDOMGlyphs, [539](#)
 - IDOMGroup, [564](#)
 - IDOMMatrix, [678](#)
 - IDOMPathGeometry, [736](#)
 - IDOMPathNode, [756](#)
 - IDOMTransformableBrush, [927](#)
- getRequestedFontName

- IDOMFontOpenType, 470
- getResolution
 - IDOMShape, 888
- getResource
 - JawsMako::ISVGGenerator, 1481
 - JawsMako::IXAMLGenerator, 1539
- getResourceDictionary
 - IDOMCanvas, 350
 - IDOMFixedPage, 453
- getResources
 - JawsMako::ISVGGenerator, 1481
 - JawsMako::IXAMLGenerator, 1539
- getRight
 - IDOMPageRectTarget, 716
- getRolloverCaption
 - JawsMako::IWidgetAppearanceCharacteristics, 1531
- getRootNode
 - IDOMJobTkContent, 631
- getRootObject
 - JawsMako::IPDFObjectStore, 1312
- getRootObjectReference
 - JawsMako::IPDFObjectStore, 1312
- getRotation
 - JawsMako::IWidgetAppearanceCharacteristics, 1532
- getRotationAngle
 - IDOMArcSegment, 331
- getRuns
 - JawsMako::ILayoutParagraph, 1141
- getScaledAppearance
 - JawsMako::IAnnotationAppearance, 234
- getscRGBProfile
 - IColorManager, 270
- getSecurityInfo
 - JawsMako::IDocumentAssembly, 317
- getSegmentBlockItem
 - IFontHeaderWriteSegmentBlockEnumerator, 989
- getSegmentBlockSize
 - IFontHeaderWriteSegmentBlockEnumerator, 990
- getSegmentItem
 - IFontHeaderWriteSegmentBlockEnumerator, 990
- getSegments
 - IDOMPathFigure, 726
- getSegmentsCount
 - IDOMPathFigure, 726
- getSelectedOptions
 - JawsMako::IWidgetAnnotation, 1524
- getSeparation
 - JawsMako::ISeparator, 1440
- getsGrayProfile
 - IColorManager, 270
- getShadingType
 - IDOMShadingPatternBrush, 850
- getShape
 - IDOMPathGeometry, 736
 - IDOMPathNode, 757
- getShapeDetails
 - IDOMShape, 888
- getShouldZeroWidthLinesBeVisible
 - IDOMPathNode, 757
- getSimpleImageBrush
 - IDOMMaskedBrush, 676
- getSimplifiedGeometry
 - IDOMPathGeometry, 736
- getSimplifiedGradient
 - IDOMRadialGradientBrush, 819
- getSingleton
 - IEDLClassFactory, 965
- getSize
 - JawsMako::ILayoutTextRun, 1149
 - JawsMako::IPDFArray, 1274
 - JawsMako::IPDFDictionary, 1283
- getSnapsToDevicePixels
 - IDOMPathNode, 757
- getSoftMaskType
 - IDOMSoftMaskBrush, 893
- getSolidity
 - IDOMDeviceNColorant, 433
- getSoundAsWav
 - JawsMako::ISoundAnnotation, 1457
- getSourceFilePath
 - IInputStream, 1102
- getSourceImage
 - IDOMCachedImage, 342
 - IDOMFilteredImage, 445
- getSpacing
 - JawsMako::ILayoutParagraph, 1141
- getSpreadMethod
 - IDOMGradientBrush, 553
- getsRGBProfile
 - IColorManager, 270
- getStartCircleCenter
 - IDOMShadingPatternType3Brush, 871
- getStartCircleRadius
 - IDOMShadingPatternType3Brush, 871
- getStartPoint
 - IDOMLinearGradientBrush, 669
 - IDOMPathFigure, 726
 - IDOMShadingPatternType2Brush, 863
- getStartupDirectory
 - ISession, 1444
- getState
 - JawsMako::IAnnotation, 229
 - JawsMako::IAnnotationAppearance, 234
- getStream
 - IDOMCachedImage, 342
 - IDOMCompositedImage, 430
 - IDOMFilteredImage, 445
 - IDOMFontSourceFromStream, 484
 - IDOMFontSourceObfuscationConverter, 488
 - IDOMFontSourceStreamFilter, 491
 - IDOMRecombineAlphaImage, 830
 - IDOMRecombineImage, 834
 - IDOMResource, 837
 - IDOMTrivialImage, 942

- getStreamLength
 - IDOMFontSourceFromStream, 484
 - IDOMResource, 837
- getString
 - JawsMako::IPDFObject, 1309
- getStroke
 - IDOMPathNode, 757
- getStrokeAdjust
 - IDOMPathNode, 758
- getStrokeDashLineCap
 - IDOMPathNode, 758
- getStrokeDashOffset
 - IDOMPathNode, 758
- getStrokeDashPattern
 - IDOMPathNode, 758
- getStrokeDashesCount
 - IDOMPathNode, 759
- getStrokeEndLineCap
 - IDOMPathNode, 759
- getStrokeLineJoin
 - IDOMPathNode, 759
- getStrokeMiterLimit
 - IDOMPathNode, 760
- getStrokeMiterLimitTreatment
 - IDOMPathNode, 760
- getStrokeOverprints
 - IDOMPathNode, 760
- getStrokePath
 - IDOMCharPathGroup, 363
- getStrokeStartLineCap
 - IDOMPathNode, 760
- getStrokeThickness
 - IDOMPathNode, 761
- getStructureElement
 - IDOMOutlineEntry, 708
- getStructureElementReference
 - IDOMForm, 497
- getStyle
 - JawsMako::IPageLabel, 1234
- getStyleSimulations
 - IDOMGlyphs, 540
- getSubsetted
 - IDOMFontOpenType, 471
- getSweepDirection
 - IDOMArcSegment, 331
- getSymbolSetID
 - IDOMFontPCL5, 475
- getSymbolSetIDCode
 - IDOMFontPCL5, 476
- getSynthetic
 - IDOMRawImage, 826
- getTableData
 - IDOMSampledFunction, 844
- getTableDimension
 - IDOMSampledFunction, 844
- getTarget
 - IDOMOutlineEntry, 709
 - JawsMako::INamedDestination, 1173
- getTargetId
 - IDOMInternalTarget, 616
- getTargetPage
 - IDOMPageTarget, 720
- getTargetType
 - IDOMActionArray, 321
 - IDOMActionLaunch, 324
 - IDOMExternalTarget, 439
 - IDOMInternalTarget, 616
 - IDOMPageRectTarget, 716
 - IDOMPageTarget, 720
 - IDOMTarget, 911
- getTargetUri
 - IDOMExternalTarget, 440
- getTemporaryDirectory
 - ISession, 1444
- getTempStore
 - ISession, 1445
- getText
 - JawsMako::ILayoutTextRun, 1149
- getTextColor
 - IDOMOutlineEntry, 709
- getTextStyle
 - IDOMOutlineEntry, 709
- getTextValue
 - JawsMako::IPDFString, 1359
- getThreads
 - JawsMako::IDocument, 310
- getThumbnail
 - IDOMFixedPage, 453
 - JawsMako::IDocumentAssembly, 317
- getTileMode
 - IDOMImageBrush, 589
 - IDOMVisualBrush, 954
- getTilingStep
 - IDOMTilingPatternBrush, 923
- getTilingType
 - IDOMTilingPatternBrush, 923
- getTime
 - IEDLTime, 985
- getTintTransform
 - IDOMColorSpaceDeviceN, 401
 - IDOMDeviceNColorant, 434
- getTop
 - IDOMPageRectTarget, 716
- getTransferFunction
 - IDOMSoftMaskBrush, 894
- getTransforms
 - JawsMako::ITransformChain, 1506
- getTrimBox
 - IDOMFixedPage, 453
- getType
 - JawsMako::IAnnotation, 229
 - JawsMako::IPDFObject, 1309
- getTypeAtIndex
 - JawsMako::IPDFArray, 1274
- getUnderlyingColorSpace
 - IDOMColorSpaceIndexed, 412

- getUnicodeString
 - IDOMGlyphs, [540](#)
- getUnixParameters
 - IDOMActionLaunch, [324](#)
- getUOffset
 - IDOMGlyph, [519](#)
- getURI
 - IDOMCatalog, [357](#)
- getUri
 - IDOMResource, [837](#)
- getUsage
 - JawsMako::IAnnotationAppearance, [234](#)
- getUserPassword
 - IDOMPublicKeyPDFSecurityInfo, [812](#)
 - IDOMStandardPDFSecurityInfo, [903](#)
- getUsers
 - JawsMako::IOptionalContentGroupUsage, [1190](#)
- getUserUnit
 - JawsMako::IPage, [1225](#)
- getValue
 - IDOMJobTkValue, [658](#)
 - JawsMako::IFormField, [1020](#)
 - JawsMako::IPDFBoolean, [1278](#)
 - JawsMako::IPDFFarReference, [1289](#)
 - JawsMako::IPDFInteger, [1302](#)
 - JawsMako::IPDFReal, [1352](#)
 - JawsMako::IPDFReference, [1355](#)
 - JawsMako::IPDFString, [1359](#)
 - JawsMako::IWidgetAnnotation, [1524](#)
- getValueAtIndex
 - JawsMako::IPDFDictionary, [1283](#)
- getValueTypeAtIndex
 - JawsMako::IPDFDictionary, [1284](#)
- getVersion
 - IDOMJobTkContent, [631](#)
- getVerticesPerRow
 - IDOMShadingPatternType4567Brush, [880](#)
- getViewBox
 - IDOMImageBrush, [589](#)
 - IDOMVisualBrush, [954](#)
- getViewBoxUnits
 - IDOMImageBrush, [590](#)
 - IDOMVisualBrush, [954](#)
- getViewPort
 - IDOMImageBrush, [590](#)
 - IDOMVisualBrush, [954](#)
- getViewPortUnits
 - IDOMImageBrush, [590](#)
 - IDOMVisualBrush, [955](#)
- getVisual
 - IDOMTilingPatternBrush, [923](#)
 - IDOMVisualBrush, [955](#)
- getVOffset
 - IDOMGlyph, [519](#)
- getWasExhausted
 - IEDLTempStore, [979](#)
- getWhitePoint
 - IDOMColorSpaceLAB, [417](#)
- getWidgetDefaultAppearanceString
 - JawsMako::IForm, [1007](#)
- getWidgetDefaultValue
 - JawsMako::IForm, [1007](#)
- getWidgetFieldFlags
 - JawsMako::IForm, [1007](#)
- getWidgetFieldType
 - JawsMako::IForm, [1008](#)
- getWidgetOptions
 - JawsMako::IForm, [1008](#)
- getWidgetQuadding
 - JawsMako::IForm, [1008](#)
- getWidgets
 - JawsMako::IForm, [1009](#)
- getWidgetValue
 - JawsMako::IForm, [1009](#)
- getWidth
 - IDOMFixedPage, [453](#)
 - IImageFrame, [1053](#)
 - JawsMako::ILayoutImageRun, [1136](#)
- getWinParameters
 - IDOMActionLaunch, [324](#)
- getXfaPacketData
 - JawsMako::IForm, [1009](#)
- getXmpPacket
 - JawsMako::IDocumentAssembly, [317](#)
- getXResolution
 - IImageFrame, [1053](#)
- getYear
 - IEDLTime, [985](#)
- getYResolution
 - IImageFrame, [1053](#)
- getZoom
 - IDOMPageRectTarget, [716](#)
- GlyphID
 - Glyphs, [65](#)
- Glyphs, [64](#)
 - eGlyphIDNotdef, [65](#)
 - eGlyphIDSpecial, [65](#)
 - GlyphID, [65](#)
- Grouping, [63](#)
- groupsVisible
 - JawsMako::IOptionalContent, [1176](#), [1177](#)
- groupShouldBeIgnored
 - JawsMako::IOptionalContent, [1177](#)
- Halftones, [99](#)
- hardSplit
 - IDOMGlyphs, [540](#)
- hasChildNodes
 - IDOMNode, [693](#)
- hasGlyph
 - IDOMType3Font, [948](#), [949](#)
- hash
 - IDOMHashable, [570](#)
 - IInputStream, [1103](#)
 - JawsMako::IHashable, [1031](#)
- hashE
 - IDOMHashable, [570](#)

- hasNormalAppearance
 - JawsMako::IAnnotation, 229
- hasReferencedObject
 - JawsMako::IPDFObjectStore, 1312
- haveMoreEnumerationItems
 - IFontHeaderWriteSegmentBlockEnumerator, 990
 - IFontPCL5WriteSegmentBlockEnumerator, 993
- IAbort, 222
- IColorManager, 254
 - convertColors, 259
 - createCalibratedGrayProfile, 261
 - createCalibratedRGBProfile, 261
 - createGrayProfile, 262
 - createXyzProfile, 262
 - deleteDeviceLinkForConversionsBetween, 263
 - get, 263
 - getCMYKSWOPProfile, 263
 - getDefaultBlackPointCompensation, 264
 - getDefaultIntentForICCBasedSpace, 264
 - getDefaultOtherBlackPreservation, 264
 - getDefaultTextBlackPreservation, 264
 - getDeviceCMYKIntercept, 265
 - getDeviceGrayIntercept, 265
 - getDeviceLinkICCPProfile, 265
 - getDeviceRGBIntercept, 266
 - getEmbeddedPSCSAForICCBasedSpace, 266
 - getMapDeviceGrayToCMYKBlack, 266
 - getNumComponentsForICCBasedSpace, 266
 - getNumComponentsForICCPProfile, 267
 - getNumDeviceLinkICCPProfiles, 267
 - getPostScriptCSAForICCBasedSpace, 267
 - getProfileClassForICCBasedSpace, 268
 - getProfileColorSpaceForICCBasedSpace, 268
 - getProfileConnectionSpaceForICCBasedSpace, 268
 - getProfileForSpace, 268
 - getProfileHasSupportForIntentForICCBasedSpace, 269
 - getProfileNameForICCBasedSpace, 269
 - getProfileVersionForICCBasedSpace, 270
 - getscRGBProfile, 270
 - getGrayProfile, 270
 - getRGBProfile, 270
 - interceptSpace, 270
 - setDefaultBlackPointCompensation, 271
 - setDefaultBlackPreservation, 271
 - setDeviceCMYKIntercept, 272
 - setDeviceGrayIntercept, 272
 - setDeviceRGBIntercept, 272
 - setMapDeviceGrayToCMYKBlack, 272
 - testGamut, 273
 - useDeviceLinkForConversionsBetween, 273, 274
- identity
 - CTransformMatrix< TItem >, 141
- IDOMActionArray, 318
 - addAction, 320
 - getActionsCount, 321
 - getActionsEnum, 321
 - getTargetType, 321
- IDOMActionLaunch, 322
 - getFile, 323
 - getMacParameters, 323
 - getNewWindow, 324
 - getTargetType, 324
 - getUnixParameters, 324
 - getWinParameters, 324
 - setFile, 324
 - setMacParameters, 325
 - setNewWindow, 325
 - setUnixParameters, 325
 - setWinParameters, 325
- IDOMArcSegment, 326
 - classID, 329
 - convertToSimpleSegment, 329
 - create, 330
 - getIsLargeArc, 330
 - getPoint, 330
 - getRadiusX, 331
 - getRadiusY, 331
 - getRotationAngle, 331
 - getSweepDirection, 331
 - setIsLargeArc, 332
 - setPoint, 332
 - setRadiusX, 332
 - setRadiusY, 333
 - setRotationAngle, 333
 - setSweepDirection, 333
- IDOMArcSegment::Data, 151
- IDOMAudioFile, 333
 - classID, 335
- IDOMAudioFile::Data, 151
- IDOMBrush, 336
 - getAdjustedForUseInTransformedNode, 337
 - getBrushType, 338
 - getOpacity, 338
 - setOpacity, 338
- idombrush.h, 1558
- IDOMCachedImage, 339
 - classID, 341
 - create, 341
 - getSourceImage, 342
 - getStream, 342
 - setStream, 342
- IDOMCachedImage::Data, 152
- IDOMCanvas, 343
 - classID, 349
 - getAutomationPropertiesHelpText, 349
 - getAutomationPropertiesName, 349
 - getEdgeMode, 349
 - getLanguage, 349
 - getNavigateLink, 350
 - getResourceDictionary, 350
 - setAutomationPropertiesHelpText, 350
 - setAutomationPropertiesName, 351
 - setEdgeMode, 351
 - setLanguage, 351

- setNavigateLink, 352
 - setResourceDictionary, 352
- IDOMCanvas::Data, 153
- IDOMCatalog, 352
 - classID, 354
 - createNewDOMid, 354, 355
 - getCount, 355
 - getByIdIndex, 355
 - getByIdNumbers, 356
 - getByIdURI, 356
 - getObject, 356
 - getURI, 357
 - registerNumbers, 357
 - registerObject, 357
 - unregisterObject, 358
- IDOMCharPathGroup, 358
 - classID, 363
 - getCharPathType, 363
 - getClippedGroup, 363
 - getStrokePath, 363
 - setCharPathType, 364
 - setClippedGroup, 365
 - setStrokePath, 365
- IDOMCharPathGroup::Data, 153
- IDOMColor, 365
 - classID, 368
 - create, 368–370, 372
 - createFromArray, 374
 - createFromVect, 374
 - createSolidCmyk, 375
 - createSolidGray, 375
 - createSolidRgb, 377
 - getAlpha, 377
 - getColorSpace, 377
 - getComponentValue, 378
 - setAlpha, 378
 - setColorSpace, 378, 379
 - setComponentValue, 379
 - testGamut, 380
- IDOMColorSpace, 381
 - equals, 383
 - getColorantName, 383
 - getColorSpaceType, 383
 - getComponentRange, 384
 - getComponentsHaveSameRange, 384
 - getDefaultRenderingIntent, 384
 - getNumComponents, 385
 - similar, 385
- IDOMColorSpaceDeviceCMY, 386
 - classID, 388
 - create, 388
- IDOMColorSpaceDeviceCMYK, 388
 - classID, 391
 - create, 391
- IDOMColorSpaceDeviceGray, 391
 - classID, 394
 - create, 394
- IDOMColorSpaceDeviceN, 394
 - classID, 398
 - create, 398, 399
 - getAlternateColorSpace, 399
 - getColorant, 399
 - getColorantInfo, 400
 - getHasRealColorant, 400
 - getIsNChannel, 400
 - getPrintingOrder, 400
 - getProcessColorSpace, 401
 - getProcessComponentNames, 401
 - getTintTransform, 401
- IDOMColorSpaceDeviceN::Data, 154
- IDOMColorSpaceDeviceRGB, 402
 - classID, 404
 - create, 404
- IDOMColorSpaceICCBased, 404
 - classID, 407
 - create, 407
 - getAlternateColorSpace, 408
 - getICCProfile, 408
 - getIsLabColorSpace, 408
 - setICCProfile, 408
- IDOMColorSpaceICCBased::Data, 155
- IDOMColorSpaceIndexed, 409
 - classID, 411
 - create, 411, 412
 - getMappingFunction, 412
 - getUnderlyingColorSpace, 412
- IDOMColorSpaceIndexed::Data, 155
- IDOMColorSpaceLAB, 413
 - classID, 415
 - create, 415
 - getBlackPoint, 416
 - getRangeAB, 416
 - getWhitePoint, 417
- IDOMColorSpaceLAB::Data, 156
- IDOMColorSpacescRGB, 417
 - classID, 419
 - create, 419
- IDOMColorSpacesGray, 420
 - classID, 422
 - create, 422
- IDOMColorSpacesRGB, 423
 - classID, 425
 - create, 425
- IDOMCompositeImage, 426
 - classID, 428
 - create, 428
 - getStream, 430
 - setStream, 430
- IDOMCompositeImage::Data, 156
- IDOMDePremultiplierFilter::Data, 157
- IDOMDePremultiplyFilter, 430
- IDOMDeviceNColorant, 431
 - classID, 433
 - getAlternateColorSpace, 433
 - getDotGainFunction, 433
 - getName, 433

- getSolidity, 433
- getTintTransform, 434
- IDOMDeviceNColorant::Data, 158
- IDOMExponentialFunction, 434
 - classID, 436
 - getExponent, 436
 - getOutputC0, 436
 - getOutputC1, 437
- IDOMExponentialFunction::Data, 159
- IDOMExternalTarget, 437
 - getIsMap, 439
 - getTargetType, 439
 - getTargetUri, 440
 - setIsMap, 440
 - setTargetUri, 440
- IDOMFilteredImage, 441
 - classID, 443
 - create, 443, 444
 - getFilterAtIndex, 444
 - getNumFilters, 444
 - getSourceImage, 445
 - getStream, 445
 - pushFilter, 445
 - setStream, 445
- IDOMFilteredImage::Data, 159
- IDOMFixedPage, 446
 - classID, 451
 - create, 451
 - getBleedBox, 451
 - getContentBox, 451
 - getCropBox, 452
 - getHeight, 452
 - getLanguage, 452
 - getPageGroup, 452
 - getResourceDictionary, 453
 - getThumbnail, 453
 - getTrimBox, 453
 - getWidth, 453
 - setBleedBox, 453
 - setContentBox, 454
 - setCropBox, 454
 - setHeight, 454
 - setLanguage, 454
 - setPageGroup, 455
 - setResourceDictionary, 455
 - setThumbnail, 455
 - setTrimBox, 456
 - setWidth, 456
- IDOMFixedPage::Data, 160
- IDOMFont, 456
 - classID, 459
 - getCharacterMap, 459
 - getFontBaseStream, 459
 - getFontSource, 460
 - getFontType, 460
 - setFontSource, 460
- idomfont.h, 1558
- IDOMFont::Data, 160
- IDOMFontOpenType, 461
 - classID, 465
 - create, 465
 - createRenamedFont, 465
 - createSubsetFont, 465
 - createTrueTypeOnlyFontVersion, 466
 - getCidMap, 466
 - getEmbedded, 467
 - getFontLicenseFromOS2Table, 467
 - getFontOpenTypeTableAccessor, 467
 - getFontTrueTypeGlyphAccessor, 468
 - getFullName, 468
 - getIsPDFStandardFont, 468
 - getIsPSStandardFont, 469
 - getIsRestricted, 469
 - getObfuscated, 469
 - getOpenTypeFontType, 469
 - getOriginalFontType, 469
 - getOriginalOS2Table, 470
 - getPostScriptName, 470
 - getRequestedFontName, 470
 - getSubsetted, 471
 - setCidMap, 471
 - setEmbedded, 471
 - stripInstructions, 471
 - validateInstructions, 472
- IDOMFontOpenType::CCIDMap, 108
 - mapping, 109
 - ordering, 109
 - registry, 109
 - supplement, 109
- IDOMFontOpenType::Data, 161
- IDOMFontOpenTypeTT::Data, 162
- IDOMFontOTFTrueType, 472
- idomfontpcl.h, 1558
- IDOMFontPCL5, 473
 - classID, 474
 - getFontName, 474
 - getFontTrueTypeGlyphAccessor, 474
 - getFormat15FontBlockEnumerator, 475
 - getFormat16FontBlockEnumerator, 475
 - getPCL5FontType, 475
 - getSymbolSetID, 475
 - getSymbolSetIDCode, 476
 - setSymbolSetIDCode, 476
- IDOMFontPCL5::Data, 163
- IDOMFontPCLXL, 476
 - classID, 477
 - getFontHeaderSegmentBlockEnumerator, 477
 - getFontName, 478
 - getFontTrueTypeGlyphAccessor, 478
 - getPCLXLFontType, 478
- IDOMFontPCLXL::Data, 164
- IDOMFontSource, 479
 - classID, 480
 - determineUri, 480
 - getFontSourceType, 481
- IDOMFontSource::Data, 165

- IDOMFontSourceFromStream, 481
 - classID, 483
 - determineUri, 483
 - getFromPcl, 483
 - getStream, 484
 - getStreamLength, 484
- IDOMFontSourceFromStream::Data, 166
- IDOMFontSourceObfuscationConverter, 485
 - classID, 487
 - determineUri, 487
 - getInputFontSource, 488
 - getStream, 488
- IDOMFontSourceObfuscationConverter::Data, 167
- IDOMFontSourceStreamFilter, 489
 - determineUri, 491
 - getFontStreamFilterType, 491
 - getInputFontSource, 491
 - getStream, 491
- IDOMFontSourceStreamFilter::Data, 168
- IDOMForm, 492
 - classID, 496
 - create, 496
 - getFormId, 496
 - getMatrix, 497
 - getPdfPropertiesDictionary, 497
 - getStructureElementReference, 497
 - setBounds, 497
 - setMatrix, 498
 - setPdfPropertiesDictionary, 498
 - setStructureElementReference, 498
- idomform.h, 1558
- IDOMForm::Data, 169
- IDOMFormInstance, 499
 - classID, 503
 - create, 503
 - getBlendMode, 503
 - getForm, 503
 - getOpacity, 503
 - getOpacityMask, 504
 - getRenderTransform, 504
 - setBlendMode, 504
 - setForm, 504
 - setOpacity, 505
 - setOpacityMask, 505
 - setRenderTransform, 505
- IDOMFormInstance::Data, 170
- IDOMFunction, 505
 - evaluate, 507
 - getFunctionType, 508
 - getInputDomain, 508
 - getNumInputValues, 508
 - getNumOutputValues, 508
 - getOutputRange, 509
- IDOMGlyph, 509
 - classID, 514
 - create, 514, 515
 - getAdvanceX, 515
 - getAdvanceY, 515
 - getBounds, 516
 - getColored, 516
 - getGlyphID, 516
 - getGlyphName, 517
 - getGlyphUnicode, 517
 - getHasCustomAdvance, 519
 - getOriginalAdvanceX, 519
 - getUOffset, 519
 - getVOffset, 519
 - setColored, 520
 - setHasCustomAdvance, 520
- IDOMGlyph::Data, 171
- IDOMGlyphIDEnumerator, 521
 - add, 522
 - beginEnumeration, 523
 - classID, 523
 - endEnumeration, 523
 - getGlyphID, 523
 - nextEnumerationItem, 523
- IDOMGlyphName, 524
- IDOMGlyphs, 525
 - classID, 532
 - create, 532
 - getBidiLevel, 533
 - getBlackPointCompensation, 533
 - getBlendMode, 533
 - getCaretStops, 533
 - getClip, 534
 - getClusters, 534
 - getDeviceFontName, 534
 - getEquivalentPath, 535
 - getFill, 535
 - getFillOverprints, 535
 - getFlattenedGlyphInfo, 535
 - getFont, 536
 - getFontIndex, 536
 - getFontRenderingEmSize, 536
 - getIndices, 536
 - getIsCharPath, 537
 - getIsInvisible, 537
 - getIsSideways, 537
 - getLanguage, 537
 - getNavigateLink, 538
 - getOpacity, 538
 - getOpacityMask, 538
 - getOriginX, 538
 - getOriginY, 539
 - getOverprintMode, 539
 - getRenderingIntent, 539
 - getRenderTransform, 539
 - getStyleSimulations, 540
 - getUnicodeString, 540
 - hardSplit, 540
 - setBidiLevel, 541
 - setBlackPointCompensation, 541
 - setBlendMode, 541
 - setCaretStops, 542
 - setClip, 542

- setClusters, [543](#)
- setDeviceFontName, [543](#)
- setFill, [543](#)
- setFillOverprints, [543](#)
- setFont, [544](#)
- setFontIndex, [544](#)
- setFontRenderingEmSize, [544](#)
- setIndices, [544](#)
- setIsCharPath, [545](#)
- setIsInvisible, [545](#)
- setIsSideways, [545](#)
- setLanguage, [545](#)
- setNavigateLink, [546](#)
- setOpacity, [546](#)
- setOpacityMask, [546](#)
- setOriginX, [546](#)
- setOriginY, [547](#)
- setOverprintMode, [547](#)
- setRenderingIntent, [547](#)
- setRenderTransform, [548](#)
- setStyleSimulations, [548](#)
- setUnicodeString, [548](#)
- split, [548](#)
- validateInstructions, [549](#)
- IDOMGlyphs::Data, [171](#)
- IDOMGradientBrush, [550](#)
 - addGradientStop, [552](#)
 - getColorInterpolationMode, [552](#)
 - getGradientStops, [553](#)
 - getSpreadMethod, [553](#)
 - normalizeStops, [553](#)
 - setColorInterpolationMode, [554](#)
 - setGradientStops, [554](#)
 - setSpreadMethod, [554](#)
- IDOMGradientStop, [555](#)
 - classID, [557](#)
 - create, [557](#)
 - getColor, [557](#)
 - getOffset, [557](#)
 - setColor, [557](#)
 - setOffset, [558](#)
- IDOMGradientStop::Data, [172](#)
- IDOMGroup, [558](#)
 - classID, [563](#)
 - create, [563](#)
 - getClip, [563](#)
 - getMarkedContentDetails, [563](#)
 - getOptionalContentDetails, [563](#)
 - getRenderTransform, [564](#)
 - setClip, [564](#)
 - setMarkedContentDetails, [564](#)
 - setOptionalContentDetails, [564](#)
 - setRenderTransform, [565](#)
- IDOMGroup::Data, [172](#)
- IDOMGroupingFunction, [565](#)
 - classID, [568](#)
 - getFunctionAtIndex, [568](#)
 - getNumFunctions, [568](#)
- IDOMGroupingFunction::Data, [173](#)
- IDOMHashable, [569](#)
 - hash, [570](#)
 - hashE, [570](#)
- IDOMICCProfile, [571](#)
 - classID, [573](#)
 - create, [573](#)
 - getProfileClass, [573](#)
 - getProfileColorSpace, [574](#)
 - getProfileConnectionSpace, [574](#)
 - getProfileHasSupportForIntent, [574](#)
 - getProfileName, [574](#)
 - getProfileVersion, [575](#)
- IDOMICCProfile::Data, [174](#)
- IDOMImage, [575](#)
 - createImageDecoder, [578](#)
 - createImageEncoder, [578](#)
 - getImageFrame, [578](#)
 - getImageProperties, [579](#)
 - getImageType, [579](#)
 - getImageWithSubstitutedColorSpace, [579](#)
 - getIsRendered, [580](#)
- IDOMImage::Data, [174](#)
- IDOMImageBitScalerFilter, [580](#)
 - classID, [581](#)
 - create, [581](#)
- IDOMImageBitScalerFilter::Data, [175](#)
- IDOMImageBleederFilter, [581](#)
 - classID, [582](#)
 - create, [582](#)
- IDOMImageBleederFilter::Data, [176](#)
- IDOMImageBrush, [583](#)
 - classID, [586](#)
 - create, [586](#)
 - getAlternateImages, [587](#)
 - getEquivalentTilingBrush, [587](#)
 - getEquivalentVisualBrush, [587](#)
 - getICCProfile, [588](#)
 - getImageSource, [588](#)
 - getOptionalContentDetails, [588](#)
 - getPdfPropertiesDictionary, [588](#)
 - getTileMode, [589](#)
 - getViewBox, [589](#)
 - getViewBoxUnits, [590](#)
 - getViewPort, [590](#)
 - getViewPortUnits, [590](#)
 - setAlternateImages, [591](#)
 - setICCProfile, [591](#)
 - setImageSource, [591](#)
 - setOptionalContentDetails, [592](#)
 - setPdfPropertiesDictionary, [592](#)
 - setTileMode, [592](#)
 - setViewBox, [592](#)
 - setViewBoxUnits, [593](#)
 - setViewPort, [593](#)
 - setViewPortUnits, [593](#)
- IDOMImageBrush::Data, [177](#)
- IDOMImageChannelSelectorFilter, [594](#)

- classID, 594
- create, 594
- IDOMImageChannelSelectorFilter::Data, 177
- IDOMImageColorConverterFilter, 595
 - classID, 596
 - create, 596
- IDOMImageColorConverterFilter::Data, 178
- IDOMImageColorKeyFilter, 596
 - classID, 597
 - create, 597
- IDOMImageColorKeyFilter::Data, 179
- IDOMImageColorSpaceSubstitutionFilter, 598
 - classID, 598
 - create, 598
- IDOMImageColorSpaceSubstitutionFilter::Data, 179
- IDOMImageDecodeFilter, 599
 - classID, 600
 - create, 600
- IDOMImageDecodeFilter::Data, 180
- IDOMImageDeindexerFilter, 600
 - classID, 601
 - create, 601
- IDOMImageDeindexerFilter::Data, 181
- IDOMImageDeviceNToBaseFilter, 601
 - classID, 602
 - create, 602
- IDOMImageDeviceNToBaseFilter::Data, 181
- IDOMImageDownsamplerFilter, 603
 - classID, 603
 - create, 603
- IDOMImageDownsamplerFilter::Data, 182
- IDOMImageInverterFilter, 604
 - classID, 604
 - create, 604
- IDOMImageInverterFilter::Data, 183
- IDOMImageMaskExpanderFilter, 605
 - classID, 605
 - create, 605
- IDOMImageMaskExpanderFilter::Data, 183
- IDOMImageMatteRemoverFilter::Data, 184
- IDOMImageProperties, 606
 - classID, 608
 - create, 608
 - getBoolProperty, 608
 - getIntProperty, 609
 - getObjectProperty, 609
 - getProperty, 609
 - setBoolProperty, 610
 - setIntProperty, 610
 - setObjectProperty, 610
 - setProperty, 611
- IDOMImageSpotMergerFilter, 611
 - classID, 612
 - create, 612, 613
- IDOMImageSpotMergerFilter::Data, 185
- IDOMInternalTarget, 614
 - getTargetId, 616
 - getTargetType, 616
 - setTargetId, 616
- IDOMJobTk, 617
 - classID, 618
 - findChild, 618, 620
 - getCombinedContent, 620
 - getContent, 620
 - getOwner, 621
 - setContent, 621
 - setOwner, 621
- IDOMJobTkContent, 622
 - addFeature, 626
 - addNamespace, 626, 627
 - addParameterInit, 627
 - addToInitString, 627
 - classID, 628
 - convertToQName, 628
 - findChild, 629
 - findNamespaceByName, 629
 - findNamespaceByPrefix, 630
 - getLevel, 630
 - getModified, 630
 - getNamespaceCollectionEnum, 630
 - getNamespacesCount, 631
 - getRootNode, 631
 - getVersion, 631
 - isValid, 631
 - loadFromFile, 631
 - loadFromInitString, 632
 - loadFromStream, 632
 - setLevel, 632
 - setModified, 633
 - setVersion, 633
- IDOMJobTkContent::Data, 185
- IDOMJobTkGenericCharacterData, 633
 - appendCharacterData, 637
 - classID, 637
 - getCharacterData, 637
 - setCharacterData, 638
- IDOMJobTkGenericCharacterData::Data, 186
- IDOMJobTkGenericNode, 638
 - addAttribute, 642
 - classID, 642
 - getAttributeAtIndex, 642
 - getNumAttributes, 643
 - getQName, 643
- IDOMJobTkGenericNode::Data, 187
- IDOMJobTkNode, 644
 - classID, 648
 - findChild, 648
 - getChildValue, 648
 - getJobTkContent, 649
 - getJobTkNodeType, 649
 - getQName, 649
 - getQNameAsString, 649
 - setJobTkNodeType, 649
 - setQName, 650
- IDOMJobTkNode::Data, 187
- IDOMJobTkOwner, 650

- classID, 654
- getJobTicket, 654
- setJobTicket, 654
- IDOMJobTkValue, 655
 - classID, 658
 - getValue, 658
 - isValid, 658
 - setValue, 658
- IDOMJobTkValue::Data, 188
- IDOMJPEGImage, 659
 - classID, 661
 - create, 661
 - createWriterAndImage, 662
 - encode, 662, 663
- IDOMLinearGradientBrush, 664
 - classID, 667
 - create, 667
 - createPdfBrush, 668
 - createShading, 668
 - getEndPoint, 668
 - getStartPoint, 669
 - setEndPoint, 669
 - setStartPoint, 669
- IDOMLinearGradientBrush::Data, 189
- IDOMMaskedBrush, 670
 - classID, 674
 - create, 674
 - getBrush, 675
 - getEquivalentXPSBrush, 675
 - getIsSoftMask, 675
 - getSimpleImageBrush, 676
 - setBrush, 676
- IDOMMaskedBrush::Data, 189
- IDOMMatrix, 676
 - classID, 678
 - getRenderTransform, 678
 - setRenderTransform, 678
- IDOMMatrix::Data, 190
- IDOMMatteRemoverFilter, 679
- IDOMMetadata, 679
 - getProperty, 682
 - getPropertyCollectionEnum, 683
 - removeProperty, 683
 - setProperty, 683
- IDOMNode, 684
 - appendChild, 687
 - cloneNode, 687
 - cloneTree, 687
 - cloneTreeAndAppend, 688
 - copyNodeData, 688
 - effectiveTransformationOfNode, 688
 - extractChild, 689
 - findChildrenOfType, 689
 - getBounds, 689
 - getDOMid, 690
 - getFirstChild, 690
 - getFlags, 690
 - getLastChild, 690
 - getNextChild, 691
 - getNextSibling, 691
 - getNodeType, 691
 - getParentNode, 691
 - getPreviousChild, 692
 - getPreviousSibling, 692
 - getProperty, 692
 - getPropertyCollectionEnum, 693
 - hasChildNodes, 693
 - insertChild, 693
 - isAncestor, 693
 - isComplete, 694
 - notifyOnDestruct, 694
 - removeProperty, 694
 - replaceChild, 695
 - setDOMid, 695
 - setNextSibling, 695
 - setParentNode, 695
 - setPreviousSibling, 696
 - setProperty, 696
 - unregisterNotify, 696
 - walkTree, 696
- idomnode.h, 1558
- IDOMNodeFlags, 697
 - DOMNodeFlags, 698
 - eNodeDirtyFlag, 698
 - eNodeInterestingFlag, 698
 - eNodeRenderFlag, 698
 - eNodeSelectedFlag, 698
 - eNodeUnpackFlag, 698
 - eNodeUserFlag, 698
 - get, 698
 - set, 699
- IDOMNullBrush, 699
 - classID, 701
 - create, 701
- IDOMNullBrush::Data, 190
- IDOMOutline, 702
 - create, 703
 - getLanguage, 704
 - getOutlineTree, 704
 - setLanguage, 704
- IDOMOutline::Data, 191
- IDOMOutlineEntry, 705
 - createNode, 707
 - getDescription, 707
 - getExpanded, 708
 - getLanguage, 708
 - getStructureElement, 708
 - getTarget, 709
 - getTextColor, 709
 - getTextStyle, 709
 - setDescription, 710
 - setExpanded, 710
 - setLanguage, 710
 - setStructureElement, 711
 - setTarget, 711
 - setTextColor, 711

- setTextStyle, [712](#)
- IDOMOutlineEntry::Data, [191](#)
- IDOMPageRectTarget, [712](#)
 - create, [714](#)
 - getBottom, [715](#)
 - getFitType, [715](#)
 - getLeft, [715](#)
 - getPagelId, [715](#)
 - getRight, [716](#)
 - getTargetType, [716](#)
 - getTop, [716](#)
 - getZoom, [716](#)
 - setBottom, [716](#)
 - setFitType, [717](#)
 - setLeft, [717](#)
 - setPagelId, [717](#)
 - setRight, [717](#)
 - setTop, [718](#)
 - setZoom, [718](#)
- IDOMPageTarget, [718](#)
 - getTargetPage, [720](#)
 - getTargetType, [720](#)
 - setTargetPage, [721](#)
- IDOMPathFigure, [721](#)
 - addSegment, [724](#)
 - classID, [724](#)
 - create, [724](#)
 - getBounds, [725](#)
 - getIsClosed, [725](#)
 - getIsFilled, [725](#)
 - getIsImmutable, [725](#)
 - getSegments, [726](#)
 - getSegmentsCount, [726](#)
 - getStartPoint, [726](#)
 - setIsClosed, [726](#)
 - setIsFilled, [727](#)
 - setStartPoint, [727](#)
- IDOMPathFigure::Data, [192](#)
- IDOMPathGeometry, [727](#)
 - addFigure, [730](#)
 - classID, [731](#)
 - create, [731](#), [732](#)
 - createEllipse, [732](#)
 - createPolygon, [732](#)
 - getBounds, [733](#)
 - getFigures, [733](#)
 - getFiguresCount, [733](#)
 - getFiguresImmutable, [734](#)
 - getFillRule, [734](#)
 - getFlattenedGeometry, [734](#)
 - getIsRect, [735](#)
 - getMutableGeometry, [735](#)
 - getRenderTransform, [736](#)
 - getShape, [736](#)
 - getSimplifiedGeometry, [736](#)
 - setFillRule, [737](#)
 - setRenderTransform, [737](#)
- IDOMPathGeometry::Data, [192](#)
- IDOMPathGeometryBuilder, [737](#)
 - arcTo, [739](#)
 - classID, [740](#)
 - close, [740](#)
 - create, [740](#)
 - createGeometry, [740](#)
 - curveTo, [741](#)
 - lineTo, [741](#)
 - moveTo, [742](#)
 - quadCurveTo, [742](#)
- IDOMPathNode, [742](#)
 - addStrokeDash, [750](#)
 - classID, [750](#)
 - createFilled, [750](#)
 - createImage, [751](#)
 - createStroked, [751](#)
 - getAutomationPropertiesHelpText, [752](#)
 - getAutomationPropertiesName, [753](#)
 - getBlackPointCompensation, [753](#)
 - getBlendMode, [753](#)
 - getClip, [753](#)
 - getEdgeMode, [754](#)
 - getFill, [754](#)
 - getFillOverprints, [754](#)
 - getIsDashed, [754](#)
 - getLanguage, [755](#)
 - getNavigateLink, [755](#)
 - getOpacity, [755](#)
 - getOpacityMask, [755](#)
 - getOverprintMode, [756](#)
 - getPathData, [756](#)
 - getRenderingIntent, [756](#)
 - getRenderTransform, [756](#)
 - getShape, [757](#)
 - getShouldZeroWidthLinesBeVisible, [757](#)
 - getSnapsToDevicePixels, [757](#)
 - getStroke, [757](#)
 - getStrokeAdjust, [758](#)
 - getStrokeDashLineCap, [758](#)
 - getStrokeDashOffset, [758](#)
 - getStrokeDashPattern, [758](#)
 - getStrokeDashesCount, [759](#)
 - getStrokeEndLineCap, [759](#)
 - getStrokeLineJoin, [759](#)
 - getStrokeMiterLimit, [760](#)
 - getStrokeMiterLimitTreatment, [760](#)
 - getStrokeOverprints, [760](#)
 - getStrokeStartLineCap, [760](#)
 - getStrokeThickness, [761](#)
 - setAutomationPropertiesHelpText, [761](#)
 - setAutomationPropertiesName, [761](#)
 - setBlackPointCompensation, [762](#)
 - setBlendMode, [762](#)
 - setClip, [762](#)
 - setEdgeMode, [762](#)
 - setFill, [763](#)
 - setFillOverprints, [763](#)
 - setLanguage, [763](#)

- setNavigateLink, [764](#)
- setOpacity, [764](#)
- setOpacityMask, [764](#)
- setOverprintMode, [764](#)
- setPathData, [765](#)
- setRenderingIntent, [765](#)
- setRenderTransform, [765](#)
- setShouldZeroWidthLinesBeVisible, [766](#)
- setSnapsToDevicePixels, [766](#)
- setStroke, [766](#)
- setStrokeAdjust, [766](#)
- setStrokeDashLineCap, [767](#)
- setStrokeDashOffset, [767](#)
- setStrokeDashPattern, [767](#)
- setStrokeEndLineCap, [768](#)
- setStrokeLineJoin, [768](#)
- setStrokeMiterLimit, [768](#)
- setStrokeMiterLimitTreatment, [769](#)
- setStrokeOverprints, [769](#)
- setStrokeStartLineCap, [769](#)
- setStrokeThickness, [769](#)
- split, [770](#)
- IDOMPPathNode::Data, [193](#)
- IDOMPPathSegment, [770](#)
 - getBounds, [772](#)
 - getEndPoint, [772](#)
 - getIsImmutable, [772](#)
 - getIsStroked, [772](#)
 - setIsStroked, [773](#)
- IDOMPCLImage, [773](#)
 - classID, [776](#)
- IDOMPCLImage::Data, [193](#)
- IDOMPDFAlternateImage, [776](#)
 - classID, [778](#)
 - create, [778](#)
 - getDefaultForPrinting, [779](#)
 - getImageBrush, [779](#)
 - getOptionalContentDetails, [779](#)
- IDOMPDFAlternateImage::Data, [194](#)
- IDOMPDFImage, [780](#)
 - classID, [783](#)
 - getAlphaDetails, [783](#)
 - getBitsPerComponent, [783](#)
 - getColorKey, [783](#)
 - getColorSpace, [783](#)
 - getDecode, [784](#)
 - getDecodeParameters, [784](#)
 - getDecodeParametersAtIndex, [784](#)
 - getImageTypes, [784](#)
- IDOMPDFImage::CCITTFaxParams, [110](#)
- IDOMPDFImage::Data, [195](#)
- IDOMPDFImage::DCTParams, [215](#)
- IDOMPDFImage::FlateLZWParams, [221](#)
- IDOMPDFImage::IDeCodeParams, [279](#)
- IDOMPDFImage::JBig2Params, [1554](#)
- IDOMPNGImage, [785](#)
 - classID, [788](#)
 - create, [788](#)
 - createWriterAndImage, [788](#)
 - encode, [789](#), [790](#)
- IDOMPolyBezierSegment, [790](#)
 - addPoint, [793](#)
 - classID, [793](#)
 - create, [793](#)
 - getPoints, [794](#)
 - getPointsCount, [794](#)
- IDOMPolyBezierSegment::Data, [196](#)
- IDOMPolyLineSegment, [794](#)
 - addPoint, [796](#)
 - classID, [796](#)
 - create, [797](#)
 - getPoints, [797](#)
 - getPointsCount, [797](#)
- IDOMPolyLineSegment::Data, [196](#)
- IDOMPolyQuadraticBezierSegment, [798](#)
 - addPoint, [800](#)
 - classID, [800](#)
 - convertToCubicBezierSegment, [800](#)
 - create, [801](#)
 - getPoints, [801](#)
 - getPointsCount, [801](#)
- IDOMPolyQuadraticBezierSegment::Data, [197](#)
- IDOMPPostScriptCalculatorFunction, [802](#)
 - classID, [804](#)
 - getCalculator, [804](#)
 - getCalculatorAsPostScriptStream, [804](#)
- IDOMPPostScriptCalculatorFunction::Data, [198](#)
- IDOMPPrintTicket, [805](#)
 - classID, [807](#)
- IDOMPPrintTicket::Data, [198](#)
- IDOMPSPDImage, [807](#)
 - classID, [810](#)
 - create, [810](#)
 - getLayerIndex, [810](#)
- IDOMPSPDImage::Data, [199](#)
- IDOMPPublicKeyPDFSecurityInfo, [811](#)
 - eChangeEncryptionAllowed, [812](#)
 - ePublicKeyPermissionsFlags, [812](#)
 - getUserPassword, [812](#)
 - isOwnerAccess, [813](#)
- IDOMRadialGradientBrush, [813](#)
 - classID, [817](#)
 - create, [817](#)
 - createShading, [818](#)
 - getCenter, [818](#)
 - getGradientOrigin, [818](#)
 - getRadiusX, [818](#)
 - getRadiusY, [819](#)
 - getSimplifiedGradient, [819](#)
 - setCenter, [819](#)
 - setGradientOrigin, [820](#)
 - setRadiusX, [820](#)
 - setRadiusY, [820](#)
- IDOMRadialGradientBrush::Data, [200](#)
- IDOMRawDataFile, [820](#)
 - classID, [822](#)

- IDOMRawDataFile::Data, 200
- IDOMRawImage, 823
 - classID, 825
 - create, 825
 - createWriterAndImage, 826
 - getSynthetic, 826
- IDOMRawImage::Data, 201
- IDOMRecombineAlphaImage, 827
 - classID, 830
 - create, 830
 - getStream, 830
 - setStream, 830
- IDOMRecombineAlphaImage::Data, 202
- IDOMRecombineImage, 831
 - classID, 834
 - create, 834
 - getStream, 834
 - setStream, 835
- IDOMRecombineImage::Data, 202
- IDOMResource, 835
 - getStream, 837
 - getStreamLength, 837
 - getUri, 837
 - setStream, 837
 - setUri, 837
- IDOMResourceDictionary, 838
 - classID, 840
 - get, 840
 - put, 840
- IDOMResourceDictionary::Data, 203
- idomresources.h, 1558
- IDOMSampledFunction, 841
 - classID, 843
 - getBitsPerSample, 843
 - getInputEncode, 843
 - getInterpolationMethod, 844
 - getOutputDecode, 844
 - getTableData, 844
 - getTableDimension, 844
- IDOMSampledFunction::Data, 204
- IDOMSecurityInfo, 845
- IDOMShadingPatternBrush, 846
 - delIndex, 848
 - getAntiAlias, 849
 - getBackgroundColor, 849
 - getBBox, 849
 - getColorSpace, 849
 - getFunction, 849
 - getShadingType, 850
 - setAntiAlias, 850
 - setBackgroundColor, 850
 - setBBox, 850
 - setColorSpace, 851
 - setFunction, 851
- IDOMShadingPatternType1Brush, 852
 - classID, 855
 - create, 855
 - getDomain, 856
 - getMatrix, 856
 - setDomain, 856
 - setMatrix, 857
- IDOMShadingPatternType1Brush::Data, 205
- IDOMShadingPatternType2Brush, 857
 - classID, 861
 - create, 861
 - getDomain, 862
 - getEndPoint, 862
 - getEquivalentSimpleBrush, 862
 - getExtend, 863
 - getStartPoint, 863
 - setDomain, 863
 - setEndPoint, 863
 - setExtend, 864
 - setStartPoint, 864
- IDOMShadingPatternType2Brush::Data, 205
- IDOMShadingPatternType3Brush, 864
 - classID, 868
 - create, 868
 - getDomain, 869
 - getEndCircleCenter, 869
 - getEndCircleRadius, 870
 - getEquivalentSimpleBrush, 870
 - getExtend, 870
 - getStartCircleCenter, 871
 - getStartCircleRadius, 871
 - setDomain, 871
 - setEndCircleCenter, 871
 - setEndCircleRadius, 872
 - setExtend, 872
 - setStartCircleCenter, 872
 - setStartCircleRadius, 872
- IDOMShadingPatternType3Brush::Data, 206
- IDOMShadingPatternType4567Brush, 873
 - classID, 877
 - create, 877, 878
 - getBitsPerComponent, 879
 - getBitsPerCoordinate, 879
 - getBitsPerFlag, 879
 - getDataSource, 879
 - getDecode, 879
 - getMeshEntries, 880
 - getVerticesPerRow, 880
 - setBitsPerComponent, 880
 - setBitsPerCoordinate, 880
 - setBitsPerFlag, 881
 - setDataSource, 881
 - setDecode, 881
 - setMeshEntries, 881
 - setShadingType, 883
 - setVerticesPerRow, 883
- IDOMShadingPatternType4567Brush::CMeshEntry, 122
 - colors, 122
 - points, 122
- IDOMShadingPatternType4567Brush::Data, 207
- IDOMShape, 883
 - classID, 886

- completelyContainsShape, 886
- difference, 886
- getAsGeometry, 887
- getAsImage, 887
- getBounds, 887
- getIsEmpty, 888
- getIsRect, 888
- getResolution, 888
- getShapeDetails, 888
- intersect, 888
- intersects, 889
- isEqualTo, 889
- offset, 889
- unite, 890
- IDOMShape::CShapeDetails, 131
- IDOMShape::CShapeDetails::CSpan, 132
- IDOMShape::Data, 207
- IDOMSoftMaskBrush, 890
 - classID, 892
 - create, 892
 - getBackdropColor, 893
 - getGroup, 893
 - getSoftMaskType, 893
 - getTransferFunction, 894
- IDOMSoftMaskBrush::Data, 208
- IDOMSolidColorBrush, 894
 - classID, 897
 - create, 897
 - createSolidCmyk, 897
 - createSolidGray, 898
 - createSolidRgb, 898
 - createWithSpaceAndComponents, 899
 - createWithSpaceAndComponentsFromArray, 900
 - createWithSpaceAndComponentsFromVect, 901
 - getColor, 901
 - setColor, 901
- IDOMSolidColorBrush::Data, 208
- IDOMStandardPDFSecurityInfo, 902
 - encryptMetadata, 903
 - getHandlerRevision, 903
 - getUserPassword, 903
 - isAnnotationEditingAllowed, 904
 - isContentAccessibilityExtractionAllowed, 904
 - isCopyingAllowed, 904
 - isDocumentAssemblyAllowed, 904
 - isEditingAllowed, 905
 - isFillingFormsAllowed, 905
 - isHighQualityPrintingAllowed, 905
 - isOwnerAccess, 905
 - isPrintingAllowed, 905
- IDOMStitchingFunction, 906
 - classID, 908
 - getBoundsVector, 908
 - getEncodeVector, 909
 - getFunctionAtIndex, 909
 - getNumFunctions, 909
- IDOMStitchingFunction::Data, 209
- IDOMTarget, 910
 - getTargetType, 911
- IDOMTIFFImage, 912
 - classID, 915
 - create, 915
 - createWriterAndImage, 915
 - encode, 916, 917
 - eTCNone, 915
 - eTIFFCompression, 914
 - getImageIndex, 917
- IDOMTIFFImage::Data, 210
- IDOMTilingPatternBrush, 918
 - classID, 920
 - create, 920
 - getBBox, 921
 - getColoredTilingPatternBrush, 921
 - getEquivalentVisualBrush, 922
 - getPaintType, 922
 - getPatternColor, 922
 - getPatternType, 922
 - getTilingStep, 923
 - getTilingType, 923
 - getVisual, 923
 - setBBox, 923
 - setPaintType, 924
 - setPatternColor, 924
 - setTilingStep, 924
 - setTilingType, 924
 - setVisual, 925
- IDOMTilingPatternBrush::Data, 210
- IDOMTransformableBrush, 925
 - getRenderTransform, 927
 - setRenderTransform, 927
- IDOMTransparencyGroup, 927
 - classID, 932
 - create, 932
 - getBlackPointCompensation, 933
 - getBlendMode, 933
 - getColorSpace, 933
 - getIsIsolated, 934
 - getIsKnockout, 934
 - getOpacity, 934
 - getOpacityMask, 934
 - getRenderingIntent, 934
 - setBlackPointCompensation, 935
 - setBlendMode, 935
 - setColorSpace, 935
 - setIsIsolated, 937
 - setIsKnockout, 937
 - setOpacity, 937
 - setOpacityMask, 937
 - setRenderingIntent, 938
- IDOMTransparencyGroup::Data, 211
- IDOMTrivialImage, 938
 - classID, 941
 - create, 941
 - getStream, 942
 - setStream, 942
- IDOMTrivialImage::Data, 212

- IDOMType3Font, 943
 - addGlyph, 946
 - classID, 946
 - getBBox, 946
 - getFontDictionary, 946
 - getGlyph, 947
 - getGlyphs, 948
 - getId, 948
 - hasGlyph, 948, 949
- IDOMType3Font::Data, 212
- IDOMVisualBrush, 949
 - classID, 952
 - create, 952
 - getEquivalentSimpleVisualBrush, 953
 - getEquivalentTilingBrush, 953
 - getTileMode, 954
 - getViewBox, 954
 - getViewBoxUnits, 954
 - getViewPort, 954
 - getViewPortUnits, 955
 - getVisual, 955
 - setTileMode, 955
 - setViewBox, 956
 - setViewBoxUnits, 956
 - setViewPort, 956
 - setViewPortUnits, 957
 - setVisual, 957
- IDOMVisualBrush::Data, 213
- IDOMVisualRoot, 957
 - classID, 961
- IDOMWMPImage, 961
 - classID, 963
- IEDLClassFactory, 964
 - createInstance, 964
 - createInstanceJawsMako, 965
 - findNamedClass, 965
 - getSingleton, 965
 - registerClass, 966
 - registerNamedClass, 966
- iedlcollection.h, 1558
- iedlenum.h, 1558
- IEDLError, 967
- IEDLNamespace, 967
 - classID, 969
 - getNamespace, 969
 - getPrefix, 969
 - setNamespace, 969
 - setPrefix, 970
- IEDLNamespace::Data, 214
- IEDLObject, 970
 - clone, 972
 - getClassID, 972
 - init, 973
- IEDLStream, 973
 - getPos, 975
 - isValid, 975
 - open, 975
- IEDLTempStore, 976
 - classID, 978
 - createTemporaryObject, 978
 - createTemporaryObjectWithContents, 978
 - createTemporaryReaderWriter, 978
 - createTemporaryReaderWriterPair, 979
 - createTemporaryStreamWithContents, 979
 - getWasExhausted, 979
- IEDLTempStore::Data, 214
- IEDLTempStoreObject, 980
 - createReader, 981
 - createWriter, 981
- IEDLTime, 982
 - compare, 984
 - fromPDFDate, 984
 - fromW3CDTF, 984
 - getDay, 985
 - getMonth, 985
 - getTime, 985
 - getYear, 985
 - isEqualTo, 986
 - setDay, 986
 - setMonth, 986
 - setTime, 986
 - setYear, 987
 - toPDFDate, 987
 - toW3CDTF, 987
- IEDLTime::Data, 215
- iedltree.h, 1558
- ifilespec.h, 1558
- IFontHeaderWriteSegmentBlockEnumerator, 988
 - classID, 989
 - getSegmentBlockItem, 989
 - getSegmentBlockSize, 990
 - getSegmentItem, 990
 - haveMoreEnumerationItems, 990
 - nextEnumerationItem, 990
 - writeSegmentBlock, 990
- IFontPCL5WriteSegmentBlockEnumerator, 991
 - classID, 992
 - getEnumerationCount, 992
 - getEnumerationItemBlockSize, 993
 - haveMoreEnumerationItems, 993
 - nextEnumerationItem, 993
 - writeEnumerationItemBlock, 993
- IFrame, 1026
- ignoreAttribute
 - JawsMako::IPCLXLAttributeHandler, 1258
- iimagecodec.h, 1558
- IImageDecoder, 1036
 - getFrame, 1037
- IImageEncoder, 1042
 - createFrame, 1043
- IImageFrame, 1049
 - getBPS, 1051
 - getColorSpace, 1051
 - getEfficientlySkippable, 1051
 - getExtraChannelType, 1051
 - getHasAlphaChannel, 1051

- getHasMask, 1052
- getHeight, 1052
- getNumChannels, 1052
- getNumExtraChannels, 1052
- getRawBytesPerRow, 1053
- getWidth, 1053
- getXResolution, 1053
- getYResolution, 1053
- readScanLine, 1053
- skipScanLines, 1054
- IImageFrameWriter, 1054
 - setBPS, 1056
 - setColorSpace, 1057
 - setHeight, 1057
 - setImageExtraChannelType, 1057
 - setWidth, 1057
 - setXResolution, 1058
 - setYResolution, 1058
 - writeScanLine, 1058
- IInputEnum< typename T >, 1085
- IInputEnumRC< typename T >, 1085
- IInputPushbackStream, 1086
- IInputStream, 1090
 - completeRead, 1093
 - completeReadE, 1094
 - createCompositeStream, 1094
 - createFromFile, 1094, 1095
 - createFromFileShared, 1095
 - createFromFlateCompressed, 1096
 - createFromLz4Compressed, 1096
 - createFromMemory, 1097
 - createFromNewFileWithContents, 1097, 1098
 - createFromPredictorCompressed, 1098
 - createFromRAUserFunc, 1099
 - createFromUserReadFunc, 1099
 - createPushbackStream, 1099
 - createRandomAccessFromNonRandomAccess, 1100
 - createSharedFromStream, 1100
 - createSubFile, 1101
 - createTbcpStream, 1101
 - createUelStream, 1101
 - eof, 1102
 - getCanonicalSourceFilePath, 1102
 - getSourceFilePath, 1102
 - hash, 1103
 - read, 1103
 - skip, 1104
- ijpdsinput.h, 1558
- Images, 69
- init
 - IEDLObject, 973
- inkInfoToColorantInfo
 - JawsMako::IRendererTransform, 1427
- Input, 82
- Input/Output, 81
- insert
 - JawsMako::IPDFArray, 1275
 - JawsMako::IPDFPageInserter, 1348
- insertChild
 - IDOMNode, 693
- insertDocument
 - JawsMako::IDocumentAssembly, 317
- insertFrom
 - JawsMako::IPDFPageInserter, 1348
- insertPage
 - JawsMako::IDocument, 310
- insertTo
 - JawsMako::IPDFPageInserter, 1349
- Interactive features, 83
- interactive.h, 1558
- interceptSpace
 - IColorManager, 270
- intersect
 - IDOMShape, 888
- intersects
 - IDOMShape, 889
- inUncoloredTilingBrush
 - JawsMako::CTransformState, 150
- invert
 - CTransformMatrix< TItem >, 141
- IOutputStream, 1203
 - completeWrite, 1205
 - completeWriteE, 1206
 - copy, 1206
 - createFromUserWriteFunc, 1207
 - createToFile, 1207
 - createToFlateCompressed, 1208
 - createToLz4Compressed, 1208
 - write, 1209
 - writeFormatted, 1209
 - writeFormattedE, 1209
 - writeStream, 1210
- IPushbackStream, 1397
 - pushBack, 1398
- IRAInputPushbackStream, 1399
- IRAInputStream, 1403
- IRAOutputStream, 1408
- IRASStream, 1410
 - length, 1411
 - setPos, 1411
 - setPosE, 1412
- IRCOobject, 1412
 - decRef, 1414
 - getRefCount, 1414
- ircobject.h, 1558
- IRunnable, 1437
 - run, 1438
- isAncestor
 - IDOMNode, 693
- isAnnotationEditingAllowed
 - IDOMStandardPDFSecurityInfo, 904
- isComplete
 - IDOMNode, 694
- isContentAccessibilityExtractionAllowed
 - IDOMStandardPDFSecurityInfo, 904

- isCopyingAllowed
 - IDOMStandardPDFSecurityInfo, 904
- isDocumentAssemblyAllowed
 - IDOMStandardPDFSecurityInfo, 904
- isEditingAllowed
 - IDOMStandardPDFSecurityInfo, 905
- isEmpty
 - EDLQName, 219
- isEqualTo
 - IDOMShape, 889
 - IEDLTime, 986
- ISession, 1442
 - getFactory, 1443
 - getLiteMessageHandler, 1443
 - getMessageHandler, 1444
 - getStartupDirectory, 1444
 - getTemporaryDirectory, 1444
 - getTempStore, 1445
 - setFactory, 1445
 - setStartupDirectory, 1445
 - setTemporaryDirectory, 1446
- isFillingFormsAllowed
 - IDOMStandardPDFSecurityInfo, 905
- isHighQualityPrintingAllowed
 - IDOMStandardPDFSecurityInfo, 905
- isOwnerAccess
 - IDOMPublicKeyPDFSecurityInfo, 813
 - IDOMStandardPDFSecurityInfo, 905
- isPrintingAllowed
 - IDOMStandardPDFSecurityInfo, 905
- isUtf8Encoded
 - JawsMako::IPDFString, 1359
- isValid
 - IDOMJobTkContent, 631
 - IDOMJobTkValue, 658
 - IEDLStream, 975
- Iterator, 1483
- iTransform
 - CTransformMatrix< TItem >, 141, 142
- JawsMako API, 45
- JawsMako Library, 46
 - create, 48
- jawsmako.h, 1558
- JawsMako::CAnnotationBorder, 106
 - CAnnotationBorder, 107
 - dash, 107
 - eBorderType, 107
 - eBTBeveled, 107
 - eBTDashed, 107
 - eBTInset, 107
 - eBTSolid, 107
 - eBTUnderline, 107
 - width, 107
- JawsMako::CFieldOption, 117
- JawsMako::CPDFFarReference, 124
 - CPDFFarReference, 125
 - getAllocatorId, 125
 - getDocumentId, 126
 - getGeneration, 126
 - getObjectNum, 126
- JawsMako::CPDFReference, 127
 - CPDFReference, 128
 - getGeneration, 128
 - getObjectNum, 128
- JawsMako::CQuadPoint, 130
- JawsMako::CRectInset, 130
- JawsMako::CTemporaryStoreParameters, 133
 - CTemporaryStoreParameters, 133
- JawsMako::CTransformState, 148
 - inUncoloredTilingBrush, 150
 - stateInsideBrush, 149
 - stateInsideNode, 149
 - transformPriv, 150
- JawsMako::CXFAPacket, 150
- JawsMako::IAnnotation, 223
 - addAppearance, 226
 - clone, 226
 - eAnnotationType, 225
 - eAT3D, 225
 - eATCaret, 225
 - eATCircle, 226
 - eATFileAttachment, 226
 - eATFreeText, 226
 - eATHighlight, 226
 - eATInk, 226
 - eATLine, 226
 - eATLink, 226
 - eATMovie, 226
 - eATOther, 226
 - eATPolygon, 226
 - eATPolyLine, 226
 - eATPopup, 226
 - eATPrinterMark, 226
 - eATProjection, 226
 - eATRedact, 226
 - eATRichMedia, 226
 - eATScreen, 226
 - eATSound, 226
 - eATSquare, 226
 - eATSquiggly, 226
 - eATStamp, 226
 - eATStrikeOut, 226
 - eATText, 226
 - eATTrapNet, 226
 - eATUnderline, 226
 - eATWatermark, 226
 - eATWidget, 226
 - getAppearance, 227
 - getAppearances, 227
 - getBorder, 227
 - getColor, 227
 - getContents, 228
 - getFlags, 228
 - getModificationTime, 228
 - getRect, 228
 - getReference, 228

- getState, 229
- getType, 229
- hasNormalAppearance, 229
- matchesReference, 229
- rotate, 230
- setBorder, 230
- setColor, 230
- setContents, 231
- setFlags, 231
- setModificationTime, 231
- setRect, 231
- setState, 232
- JawsMako::IAnnotationAppearance, 232
 - clone, 233
 - create, 233
 - getForm, 234
 - getScaledAppearance, 234
 - getState, 234
 - getUsage, 234
- JawsMako::IAnnotationReference, 235
 - equals, 236
- JawsMako::IAnnotationUtils, 236
 - createAnnotationReferenceFromTag, 237
 - generateXMLForDocument, 237
- JawsMako::IAnnotationUtils::CXMLResource, 150
- JawsMako::IBlendSimplifierTransform, 238
 - create, 239
- JawsMako::ICaretAnnotation, 240
 - getRectInset, 243
- JawsMako::ICFFCIDSplitterTransform, 243
 - create, 245
- JawsMako::ICFFSanitizerTransform, 245
 - create, 247
- JawsMako::IColorConverterTransform, 248
 - create, 251
 - setBlackPreservation, 251
 - setConvertColorsInsideLuminositySoftMasks, 252
 - setConvertThroughTransparencyGroupColorSpaces, 252
 - setDeviceNHandling, 252
 - setKeepOverprintMode, 253
 - setTargetProfile, 254
 - setTargetSpace, 254
- JawsMako::IComplexColorSimplifierTransform, 274
 - create, 276
- JawsMako::ICustomTransform, 277
- JawsMako::ICustomTransform::Implementation, 1061
 - transformAnnotation, 1063
 - transformAnnotationAppearance, 1063
 - transformBrush, 1064
 - transformCanvas, 1064
 - transformCharPathGroup, 1065
 - transformColor, 1065
 - transformColorSpace, 1066
 - transformFixedPage, 1066
 - transformFont, 1067
 - transformForm, 1067
 - transformFormInstance, 1068
 - transformGlyphs, 1068
 - transformGradientBrush, 1069
 - transformGroup, 1069
 - transformImage, 1070
 - transformImageBrush, 1070
 - transformLinearGradientBrush, 1071
 - transformMaskedBrush, 1071
 - transformNode, 1071
 - transformNullBrush, 1072
 - transformPath, 1072
 - transformRadialGradientBrush, 1073
 - transformShadingPatternBrush, 1073
 - transformSoftMaskBrush, 1074
 - transformSolidColorBrush, 1074
 - transformTilingPatternBrush, 1075
 - transformTransparencyGroup, 1075
 - transformVisualBrush, 1076
 - transformVisualRoot, 1076
- JawsMako::IDistiller, 280
 - addFont, 286
 - addFonts, 286
 - create, 286
 - distill, 287
 - eDIAuthor, 284
 - eDICreator, 284
 - eDIN, 284
 - eDISubject, 284
 - eDITitle, 284
 - eDocumentInfo, 284
 - eICAUTO, 284
 - eICCCITT, 284
 - eICDCT, 284
 - eICFlate, 284
 - eICFlatePredict, 284
 - eICLZW, 284
 - eICNone, 284
 - eIDAverage, 284
 - eIDBicubic, 284
 - eIDNone, 284
 - eIDSubsample, 284
 - eImageCompression, 284
 - eImageDownsamplingMethod, 284
 - eJpegQuality, 284
 - eJQHigh, 285
 - eJQLow, 285
 - eJQMedium, 285
 - eJQUser, 285
 - ePDF1_3, 285
 - ePDF1_4, 285
 - ePDF1_5, 285
 - ePDF1_6, 285
 - ePDF1_7, 285
 - ePDFVersion, 285
 - eTHColor, 285
 - eTHMono, 285
 - eTHNone, 285
 - eThumbnailType, 285
 - eTransferFunctionMethod, 285

- eTRApply, [286](#)
- eTRPreserve, [286](#)
- eTRRemove, [286](#)
- getFontNames, [287](#)
- removeFont, [287](#)
- setAscii85EncodePages, [288](#)
- setAutoRotatePages, [288](#)
- setColorImageCompression, [288](#)
- setColorImageDownsampling, [290](#)
- setColorImageResolution, [290](#)
- setColorJPEGQuality, [290](#)
- setColorQFactor, [290](#)
- setCommentMonitor, [291](#)
- setCompressObjects, [291](#)
- setCompressPages, [292](#)
- setConvertCMYKImagesToRGB, [292](#)
- setCropToBBox, [292](#)
- setDefaultPanoseStyle, [292](#)
- setDocumentInformationString, [293](#)
- setEmbedBase14Fonts, [293](#)
- setEmbedDiskBasedCidFonts, [293](#)
- setEmbedFonts, [294](#)
- setEncryption, [294](#)
- setEpilog, [295](#)
- setFontDevicePath, [295](#)
- setGrayImageCompression, [295](#)
- setGrayImageDownsampling, [296](#)
- setGrayImageResolution, [296](#)
- setGrayJPEGQuality, [296](#)
- setGrayQFactor, [296](#)
- setHalfTone, [297](#)
- setHiResBBox, [297](#)
- setJobTicket, [298](#)
- setLinearize, [298](#)
- setMonoImageCompression, [298](#)
- setMonoImageDownsampling, [298](#)
- setMonoImageResolution, [299](#)
- setOPI, [299](#)
- setOverprint, [299](#)
- setOverprintMode, [300](#)
- setPageFilterHigh, [300](#)
- setPageFilterLow, [300](#)
- setPageFilterRange, [301](#)
- setPanose, [301](#)
- setParameter, [301](#)
- setPdfVersion, [302](#)
- setProlog, [302](#)
- setResolution, [303](#)
- setResourceDevicePath, [303](#)
- setSubsetFonts, [303](#)
- setThumbnails, [304](#)
- setTransfers, [304](#)
- setTrimToBBox, [304](#)
- setUseDeviceDependentColor, [304](#)
- JawsMako::IDocument, [305](#)
 - addEmbeddedStream, [307](#)
 - addNamedDestination, [308](#)
 - appendPage, [308](#)
 - clone, [308](#)
 - create, [308](#)
 - getForm, [309](#)
 - getJobTicket, [309](#)
 - getNumPages, [309](#)
 - getObjectStore, [309](#)
 - getOutline, [309](#)
 - getPage, [310](#)
 - getThreads, [310](#)
 - insertPage, [310](#)
 - lookupFarReference, [311](#)
 - pageExists, [311](#)
 - readPdfObject, [311](#)
 - readPdfTrailerDictionary, [312](#)
 - removePage, [312](#)
 - setNamedDestinations, [312](#)
 - setOutline, [312](#)
- JawsMako::IDocumentAssembly, [313](#)
 - appendDocument, [315](#)
 - clone, [315](#)
 - create, [315](#)
 - documentExists, [315](#)
 - getDocument, [316](#)
 - getJobMetadata, [316](#)
 - getJobTicket, [316](#)
 - getNumDocuments, [316](#)
 - getSecurityInfo, [317](#)
 - getThumbnail, [317](#)
 - getXmpPacket, [317](#)
 - insertDocument, [317](#)
 - removeDocument, [318](#)
 - setThumbnail, [318](#)
- JawsMako::IFontProcessorDeferredTransform, [994](#)
 - create, [996](#)
 - shouldMerge, [997](#)
 - shouldSubset, [997](#)
- JawsMako::IFontProcessorTransform, [997](#)
 - create, [999](#)
 - shouldMerge, [1000](#)
 - shouldSubset, [1000](#)
- JawsMako::IForm, [1000](#)
 - addField, [1003](#)
 - addWidget, [1003](#)
 - clone, [1003](#)
 - create, [1003](#)
 - fieldInTree, [1004](#)
 - getFieldDefaultValue, [1004](#)
 - getFields, [1004](#)
 - getFieldValue, [1005](#)
 - getNeedAppearances, [1005](#)
 - getPathToField, [1005](#)
 - getPathToWidget, [1006](#)
 - getWidgetDefaultAppearanceString, [1007](#)
 - getWidgetDefaultValue, [1007](#)
 - getWidgetFieldFlags, [1007](#)
 - getWidgetFieldType, [1008](#)
 - getWidgetOptions, [1008](#)
 - getWidgetQuadding, [1008](#)

- getWidgets, 1009
- getWidgetValue, 1009
- getXfaPacketData, 1009
- removeField, 1009
- removeWidget, 1010
- setNeedAppearances, 1010
- setXfaPacketData, 1010
- widgetInTree, 1011
- JawsMako::IFormDeduplicatorTransform, 1011
- JawsMako::IFormField, 1013
 - addChildField, 1016
 - addChildWidget, 1016
 - clone, 1016
 - create, 1016
 - fieldInSubtree, 1017
 - getChildFields, 1017
 - getChildWidgets, 1017
 - getDefaultAppearanceString, 1018
 - getDefaultValue, 1018
 - getFieldFlags, 1018
 - getFieldId, 1019
 - getFieldType, 1019
 - getOptions, 1019
 - getPartialName, 1019
 - getQuadding, 1020
 - getValue, 1020
 - removeChildField, 1020
 - removeChildWidget, 1020, 1021
 - setDefaultAppearanceString, 1021
 - setDefaultValue, 1021
 - setFieldFlags, 1022
 - setOptions, 1022
 - setPartialName, 1022
 - setQuadding, 1022
 - setValue, 1023
 - widgetInSubtree, 1023, 1024
- JawsMako::IFormUnpackerTransform, 1024
 - create, 1026
- JawsMako::IFreeTextAnnotation, 1027
 - getCalloutLine, 1030
 - getRectInset, 1030
- JawsMako::IHashable, 1031
 - hash, 1031
 - updateHash, 1031
- JawsMako::IJPDSInput, 1032
 - create, 1033
- JawsMako::IJPDSPageRaster, 1034
- JawsMako::IImageDownsamplerTransform, 1038
 - create, 1040
 - setDownsampleMaskedImages, 1041
 - setUseMaskResolutionForMaskedImages, 1041
- JawsMako::IImageEncoderTransform, 1043
 - create, 1046
 - eEFJPEG, 1046
 - eEncodeFormat, 1046
 - setColorTIFFCompression, 1046
 - setGrayTIFFCompression, 1047
 - setJPEGQuality, 1047
 - setMonoTIFFCompression, 1047
 - setPreferredColorFormat, 1047
 - setPreferredFormat, 1048
 - setPreferredGrayFormat, 1048
 - setPreferredMonoFormat, 1048
 - setTIFFCompression, 1048
- JawsMako::IImageMergerTransform, 1058
 - create, 1060
 - setAllowMaskedResults, 1060
- JawsMako::IInkAnnotation, 1077
 - create, 1080
 - getInkList, 1080
 - setInkList, 1080
- JawsMako::IInput, 1081
 - create, 1082
 - open, 1083
 - setParameter, 1083
 - setSequentialMode, 1084
- JawsMako::IJawsMako, 1104
 - enablePSInput, 1107
 - findFont, 1107, 1108
- JawsMako::IJawsRenderer, 1108
 - create, 1111
 - render, 1111
 - renderAntiAliased, 1112
 - renderMonochrome, 1113
 - renderMonochromeToFrameBuffer, 1113
 - renderScreened, 1114
 - renderScreenedToFrameBuffers, 1115
 - renderSeparations, 1116
 - renderSeparationsToFrameBuffers, 1117
 - renderToFrameBuffer, 1119
 - renderToFrameBufferPadAndReverse, 1120
 - setBlackPreservation, 1121
 - setEnableRasterCompression, 1122
 - setEnableTrueTypeNotDef, 1122
 - setIgnoreMatchingDeviceIntercept, 1122
- JawsMako::IJawsRenderer::CColorSpotHalftone, 115
- JawsMako::IJawsRenderer::CColorThresholdArrayHalftone, 115
- JawsMako::IJawsRenderer::CEDSHalftone, 116
- JawsMako::IJawsRenderer::CFrameBufferInfo, 118
 - pixelStride, 118
- JawsMako::IJawsRenderer::CSpotHalftone, 133
- JawsMako::IJawsRenderer::CThresholdArrayHalftone, 134
- JawsMako::IJawsRenderer::CThresholdHalftone, 135
- JawsMako::ILayout, 1123
 - addFrame, 1124
 - create, 1125
 - layout, 1125, 1126
- JawsMako::ILayoutFont, 1126
 - create, 1129
 - eFontProportion, 1128
 - eFontSerifStyle, 1128
 - eFontStyle, 1128
 - eFPAny, 1128
 - eFPMonoSpaced, 1128

- eFPProportional, 1128
- eFSAny, 1128
- eFSItalic, 1128
- eFSRegular, 1128
- eFSSAny, 1128
- eFSSSansSerif, 1128
- eFSSSerif, 1128
- findFont, 1129
- findFonts, 1130
- JawsMako::ILayoutFont::CLayoutFontItem, 122
- JawsMako::ILayoutFontWeight, 1130
 - create, 1132
 - createFromRange, 1132
 - createFromVect, 1132
 - difference, 1133
 - eAny, 1131
 - eBlack, 1131
 - eBold, 1131
 - eExtraBlack, 1131
 - eExtraBold, 1131
 - eExtraLight, 1131
 - eFontWeight, 1131
 - eLight, 1131
 - eMedium, 1131
 - eNormal, 1131
 - equals, 1133
 - eSemiBold, 1131
 - eSemiLight, 1131
 - eThin, 1131
 - match, 1133
- JawsMako::ILayoutImageRun, 1134
 - create, 1135
 - getHeight, 1135
 - getImage, 1136
 - getWidth, 1136
- JawsMako::ILayoutParagraph, 1136
 - addRun, 1138
 - clone, 1139
 - create, 1139
 - eHACenter, 1138
 - eHAJustified, 1138
 - eHALeft, 1138
 - eHARight, 1138
 - eHorizontalAlignment, 1138
 - getDefaultColor, 1140
 - getDefaultFont, 1140
 - getDefaultFontIndex, 1140
 - getDefaultFontSize, 1141
 - getHorizontalAlignment, 1141
 - getLeading, 1141
 - getRuns, 1141
 - getSpacing, 1141
 - setDefaultColor, 1142
 - setDefaultFont, 1142
 - setDefaultFontSize, 1143
 - setLeading, 1143
 - setSpacing, 1143
- JawsMako::ILayoutRun, 1143
- JawsMako::ILayoutTextRun, 1145
 - create, 1146–1148
 - getColor, 1148
 - getFont, 1149
 - getFontIndex, 1149
 - getLetterSpacing, 1149
 - getSize, 1149
 - getText, 1149
 - setColor, 1149
 - setLetterSpacing, 1150
- JawsMako::ILineAnnotation, 1150
 - getCaptionOffset, 1153
 - getInteriorColor, 1154
 - getLeaderLineExtensionsLength, 1154
 - getLeaderLineLength, 1154
 - getLeaderLineOffset, 1154
 - getLineEndpoints, 1154
 - setInteriorColor, 1155
 - setLineEndpoints, 1155
- JawsMako::ILinkAnnotation, 1155
 - create, 1158
 - getActionOrDest, 1159
 - getHighlightMode, 1159
 - getPA, 1159
 - getQuadPoints, 1159
 - setActionOrDest, 1159
 - setHighlightMode, 1160
 - setPA, 1160
 - setQuadPoints, 1160
- JawsMako::IMarkedContentArtifactDetails, 1160
- JawsMako::IMarkedContentDetails, 1162
- JawsMako::IMarkedContentStructureDetails, 1164
- JawsMako::IMarkupAnnotation, 1165
 - createNormalAppearance, 1168
 - getAuthor, 1168
 - getCreationTime, 1168
 - getOpacity, 1169
 - getPopupReference, 1169
 - setAuthor, 1169
 - setCreationTime, 1169
 - setOpacity, 1170
 - setPopup, 1170
- JawsMako::IMediaHandler, 1171
 - sizeRequest, 1171
- JawsMako::INamedDestination, 1171
 - clone, 1173
 - create, 1173
 - getName, 1173
 - getTarget, 1173
- JawsMako::IOptionalContent, 1174
 - forceGroupState, 1176
 - groupsVisible, 1176, 1177
 - groupShouldBeIgnored, 1177
 - makeNodeOptional, 1178
 - setDefaultConfiguration, 1178
- JawsMako::IOptionalContentConfiguration, 1178
- JawsMako::IOptionalContentConfiguration::COrderEntry, 123

- JawsMako::IOptionalContentDetails, 1181
 - eVisibilityPolicy, 1182
 - eVPAIIOff, 1183
 - eVPAIION, 1183
 - eVPAAnyOff, 1183
 - eVPAAnyOn, 1183
 - getIsVisible, 1183
- JawsMako::IOptionalContentFixerTransform, 1183
 - create, 1185
- JawsMako::IOptionalContentGroup, 1185
- JawsMako::IOptionalContentGroupReference, 1187
- JawsMako::IOptionalContentGroupUsage, 1188
 - getUsers, 1190
 - recommendedVisibilityForCategories, 1190
 - setLanguage, 1191
 - setPageElement, 1191
 - setUsers, 1191
- JawsMako::IOptionalContentGroupUsageApplication, 1192
- JawsMako::IOptionalContentVisibilityExpression, 1193
 - evaluate, 1195
 - eVEOAnd, 1195
 - eVEOGroupState, 1195
 - eVEONot, 1195
 - eVEOOr, 1195
 - eVisibilityExpressionOperation, 1195
- JawsMako::IOutput, 1196
 - create, 1197
 - openWriter, 1198
 - setAllowedPermissionsFlags, 1199
 - writeAssembly, 1199, 1200
- JawsMako::IOutputIntent, 1200
 - getChecksum, 1202
 - getProfileFileSpecifications, 1202
- JawsMako::IOutputWriter, 1210
 - beginDocument, 1211
 - writePage, 1211
- JawsMako::IOverprintSimulationTransform, 1212
 - create, 1214
 - setSimulateBlackDeviceGrayTextOverprint, 1214
 - setSimulationColorSpace, 1214
- JawsMako::IOXPSInput, 1215
 - create, 1217
- JawsMako::IOXPSOutput, 1217
 - create, 1220
- JawsMako::IPage, 1221
 - clone, 1224
 - create, 1224
 - getContent, 1224
 - getJobTicket, 1224
 - getMediaBox, 1224
 - getObjectStore, 1225
 - getPageLabel, 1225
 - getPageRaster, 1225
 - getUserUnit, 1225
 - lookupFarReference, 1226
 - release, 1226
 - removeAnnotation, 1227
 - revert, 1227
 - setMediaBox, 1227
 - setPageLabel, 1228
- JawsMako::IPageCropperTransform, 1228
 - create, 1230
- JawsMako::IPageLabel, 1230
 - clone, 1233
 - create, 1233
 - eDecimal, 1233
 - eLabelStyle, 1232
 - eLetterLowercase, 1233
 - eLetterUppercase, 1233
 - eNone, 1233
 - eRomanLowercase, 1233
 - eRomanUppercase, 1233
 - getNumber, 1233
 - getPrefix, 1233
 - getStyle, 1234
 - setNumber, 1234
 - setPrefix, 1234
 - setStyle, 1235
- JawsMako::IPageLayout, 1235
 - create, 1236, 1237
 - setLineSpacingThreshold, 1237
 - setMultipleSpaceMode, 1237
 - setVirtualSpaceThreshold, 1238
- JawsMako::IPageLayoutData, 1238
 - create, 1239
- JawsMako::IPageLayoutNode, 1240
 - create, 1241
- JawsMako::IPageRaster, 1241
 - getAsDOMImage, 1243
- JawsMako::IPatternConverterTransform, 1244
 - create, 1245
- JawsMako::IPCL5Input, 1246
 - create, 1248
 - enableUnencapsulatedMode, 1248
 - setDefaultCopies, 1249
 - setDefaultDuplex, 1249
 - setDefaultDuplexBindingMode, 1249
 - setDefaultLandscape, 1249
 - setDefaultManualFeed, 1249
 - setDefaultPaperSize, 1250
 - setDefaultsFromPjl, 1250
 - setFlattenRops, 1250
 - setIgnorePrescribe, 1251
 - setMediaHandler, 1251
 - setPermanentResourceStore, 1251
 - setRopResolution, 1252
- JawsMako::IPCL5Output, 1253
 - create, 1255
 - setEmitPjl, 1255
 - setEnableTrueTypeNotDef, 1255
 - setImageCompression, 1255
 - setIncludeMarginsWhenSelectingPaper, 1256
 - setMediaSource, 1256
 - setOpenStream, 1256
 - setResolution, 1257

- setVersion, 1257
- JawsMako::IPCLXLAttributeHandler, 1257
 - ignoreAttribute, 1258
- JawsMako::IPCLXLInput, 1258
 - create, 1260
 - enableUnencapsulatedMode, 1261
 - setAttributeHandler, 1261
 - setDefaultCopies, 1261
 - setDefaultDuplex, 1262
 - setDefaultDuplexBindingMode, 1262
 - setDefaultLandscape, 1262
 - setDefaultManualFeed, 1262
 - setDefaultPaperSize, 1262
 - setDefaultsFromPjl, 1263
 - setFlattenRops, 1263
 - setMediaHandler, 1264
 - setRopResolution, 1264
- JawsMako::IPCLXLOutput, 1265
 - create, 1267
 - setEmitPjl, 1267
 - setOpenStream, 1267
- JawsMako::IPDFArray, 1268
 - append, 1270
 - containsCompositeObject, 1270
 - copy, 1271
 - create, 1271
 - createFromFBox, 1272
 - createFromNumbers, 1272
 - createFromNumbersArray, 1272
 - get, 1273
 - getBox, 1273
 - getInteger, 1273
 - getNumbers, 1274
 - getSize, 1274
 - getTypeAtIndex, 1274
 - insert, 1275
 - put, 1275
 - remove, 1275
- JawsMako::IPDFBoolean, 1276
 - create, 1277
 - getValue, 1278
- JawsMako::IPDFDictionary, 1278
 - begin, 1281
 - containsCompositeObject, 1281
 - copy, 1281
 - create, 1282
 - end, 1282
 - get, 1282, 1283
 - getKeyAtIndex, 1283
 - getSize, 1283
 - getValueAtIndex, 1283
 - getValueTypeAtIndex, 1284
 - put, 1284, 1285
 - putInteger, 1285
 - putReal, 1285
 - undefine, 1285, 1286
- JawsMako::IPDFFarReference, 1286
 - create, 1288
- getValue, 1289
- JawsMako::IPDFInput, 1289
 - create, 1292
 - getIncrementalSaves, 1292, 1293
 - getNumIncrementalSaves, 1293, 1294
 - openIncremental, 1294, 1295
 - scanPdfForFonts, 1295, 1296
 - scanPdfForInks, 1296, 1297
 - setAlternateBaseFont, 1298
 - setDefaultRenderingIntent, 1298
 - setFailOnFontFallback, 1298
 - setImproveGlyphSelectionForUnembeddedIdentityFonts, 1298
 - setPassword, 1299
 - setPkcs12, 1299
- JawsMako::IPDFInput::CPdfFontInfo, 126
 - font, 127
- JawsMako::IPDFInput::CPdfScannedInk, 129
 - pageNums, 129
- JawsMako::IPDFInteger, 1300
 - create, 1302
 - getValue, 1302
- JawsMako::IPDFNull, 1302
 - create, 1304
- JawsMako::IPDFObject, 1305
 - clone, 1307
 - containsReferences, 1307
 - createNumber, 1307
 - deepClone, 1307
 - getIsExecutable, 1307
 - getNumber, 1308
 - getString, 1309
 - getType, 1309
- JawsMako::IPDFObjectStore, 1309
 - clone, 1311
 - dirtyObject, 1311
 - getFarReferenceForReference, 1311
 - getRootObject, 1312
 - getRootObjectReference, 1312
 - hasReferencedObject, 1312
 - newReference, 1312
 - store, 1312
 - storeUsingNewReference, 1313
- JawsMako::IPDFOutput, 1313
 - create, 1322
 - eICDCT, 1320
 - eICRLE, 1320
 - eImageCompression, 1319
 - ePAIHDefault, 1320
 - ePAIHFailIfAlternatesFound, 1320
 - ePAIHFailIfDefaultForPrintingFound, 1320
 - ePDF1_3, 1321
 - ePDF1_4, 1321
 - ePDF1_5, 1321
 - ePDF1_6, 1321
 - ePDF1_7, 1322
 - ePDF2_0, 1322
 - ePDFA1b, 1322

- ePDFa2b, [1322](#)
- ePDFa2u, [1322](#)
- ePDFa3b, [1322](#)
- ePdfAlternateImageHandling, [1320](#)
- ePdfDeviceNHandling, [1320](#)
- ePdfExtendedGraphicsStateHandling, [1321](#)
- ePdfOptionalContentHandling, [1321](#)
- ePDFUA, [1322](#)
- ePDFVersion, [1321](#)
- ePDFX1a, [1322](#)
- ePDFX4, [1322](#)
- ePDNFailOnMismatchedDeviceNProcessSpace, [1320](#)
- ePDNFailOnNonUtf8ColorantName, [1320](#)
- ePDNHDefault, [1320](#)
- ePDNHFailIfBetterColorantInformationFoundAfterFirstUse, [1320](#)
- ePDNHFailIfDotGainFound, [1320](#)
- ePDNHFailOnInconsistentColorantInformation, [1320](#)
- ePDNHFailOnMissingColorantInformation, [1320](#)
- ePEGSHDefault, [1321](#)
- ePEGSHFailOnTransferFunction, [1321](#)
- ePOCHDefault, [1321](#)
- ePOCHFailIfAutoStateFound, [1321](#)
- ePOCHFailOnInconsistentConfigurationName, [1321](#)
- ePOCHFailOnInvalidOrder, [1321](#)
- overrideMaximumICCVersion, [1322](#)
- setAllowOptionalContentUpdateDuringWrite, [1322](#)
- setAllowRestrictedFonts, [1323](#)
- setAlternateImageErrorHandling, [1323](#)
- setAlwaysEmbedFonts, [1324](#)
- setAutoRotatePages, [1324](#)
- setBlockNotdefGlyphs, [1324](#)
- setColorImageMaxResolution, [1324](#)
- setCompressObjects, [1325](#)
- setCompressPages, [1325](#)
- setConvertAllColors, [1325](#)
- setConvertGray, [1326](#)
- setDefaultFormCJKLanguage, [1326](#)
- setDeviceNErrorHandling, [1326](#)
- setDownsampleMaskedImages, [1327](#)
- setEmbedBase14Fonts, [1327](#)
- setEmbedFonts, [1327](#)
- setEmbedIdentityCidFonts, [1327](#)
- setEmit30CmapSubtableForSymbolicTrueTypeFonts, [1328](#)
- setEnableIncrementalOutput, [1328](#)
- setEnableTrueTypeNotDef, [1329](#)
- setEncryption, [1329](#)
- setExtendedGraphicsStateErrorHandling, [1330](#)
- setForceEmitPageGroup, [1331](#)
- setForceMediaBoxOriginZero, [1331](#)
- setGrayImageMaxResolution, [1331](#)
- setJPEGChromaSubsampling, [1332](#)
- setJPEGQuality, [1332](#)
- setLinearize, [1333](#)
- setMonolImageMaxResolution, [1333](#)
- setNeverEmbedFonts, [1333](#)
- setNonUtf8InkNameFallbackEncoding, [1333](#)
- setOptionalContentErrorHandling, [1334](#)
- setOutputIntent, [1334](#)
- setOutputIntents, [1334](#)
- setPreferredColorImageCompression, [1335](#)
- setPreferredGrayImageCompression, [1336](#)
- setPreferredMonolImageCompression, [1336](#)
- setPreferredRenderedImageCompression, [1336](#)
- setProducer, [1337](#)
- setPublicKeyEncryption, [1337](#)
- setReencodeImages, [1338](#)
- setRenderResolution, [1338](#)
- setRetainClippedContent, [1338](#)
- setRetainEmbeddedFiles, [1339](#)
- setSubsetFonts, [1339](#)
- setTargetColorSpace, [1339](#)
- setTargetProfile, [1340](#)
- setUseMaskResolutionForMaskedImages, [1340](#)
- setValidateXmp, [1341](#)
- setVersion, [1341](#)
- JawsMako::IPDFPageExtractor, [1342](#)
 - create, [1343](#)
 - extract, [1344](#)
 - extractFrom, [1344](#)
 - extractTo, [1345](#)
 - getNumPages, [1345](#)
- JawsMako::IPDFPageInserter, [1346](#)
 - create, [1347](#)
 - getNumPages, [1348](#)
 - insert, [1348](#)
 - insertFrom, [1348](#)
 - insertTo, [1349](#)
 - save, [1349](#)
- JawsMako::IPDFReal, [1350](#)
 - create, [1351](#)
 - getValue, [1352](#)
- JawsMako::IPDFReference, [1352](#)
 - create, [1354](#)
 - getGeneration, [1355](#)
 - getObjectNum, [1355](#)
 - getValue, [1355](#)
- JawsMako::IPDFString, [1356](#)
 - create, [1358](#)
 - createText, [1359](#)
 - getTextValue, [1359](#)
 - getValue, [1359](#)
 - isUtf8Encoded, [1359](#)
- JawsMako::IPDFValidator, [1360](#)
 - validate, [1361](#)
- JawsMako::IPDFValidator::CContentError, [116](#)
- JawsMako::IPDFValidator::CGeneralError, [118](#)
- JawsMako::IPDFValidator::CPageErrors, [124](#)
- JawsMako::IPJLParser, [1362](#)
 - create, [1364](#)
 - eDBMLongEdge, [1364](#)
 - eDBMShortEdge, [1364](#)

- eDuplexBindingMode, 1364
- ePjIResult, 1364
- ePREndOfFile, 1364
- ePREnterHpgl2, 1364
- ePREnterPcl, 1364
- ePREnterPclXI, 1364
- ePREnterPdf, 1364
- ePREnterPostScript, 1364
- getAttributes, 1364
- parse, 1364
- JawsMako::IPJLParser::CPjIAttributeValue, 129
 - key, 130
 - modifier, 130
 - value, 130
- JawsMako::IPolyAnnotation, 1365
 - getPoints, 1368
 - setPoints, 1368
- JawsMako::IPopupAnnotation, 1369
 - create, 1371
 - getOpen, 1372
 - setOpen, 1372
- JawsMako::IPPMLInput, 1372
 - create, 1374
- JawsMako::IPRNInput, 1374
 - create, 1376
 - setIgnorePrescribe, 1376
- JawsMako::IProgressMonitor, 1377
 - create, 1378, 1379
 - getAbort, 1379
 - getProgressTick, 1379
- JawsMako::IProgressTick, 1380
 - create, 1381, 1382
 - setCallback, 1382
 - setTickMax, 1382
- JawsMako::IPSComentMonitor, 1383
 - comment, 1383
 - getComments, 1384
- JawsMako::IPSInjector, 1384
 - afterBeginPageSetup, 1385
 - afterBeginSetup, 1385
 - afterLastByte, 1385
 - beforeEndPageSetup, 1385
 - beforeEndSetup, 1386
 - beforeFirstByte, 1386
 - beforePsHeader, 1386
 - beforeShowpage, 1387
- JawsMako::IPSInput, 1387
 - create, 1389
 - enableUnencapsulatedMode, 1389
 - setProlog, 1389
- JawsMako::IPSOutput, 1390
 - create, 1393
 - setAllowReusableImageStreams, 1393
 - setColorImageMaxResolution, 1393
 - setConvertAllObjectsToTargetColorSpace, 1394
 - setGrayImageMaxResolution, 1394
 - setJPEGQuality, 1394
 - setMonoImageMaxResolution, 1395
 - setPreferredColorImageCompression, 1395
 - setPreferredGrayImageCompression, 1396
 - setPreferredMonoImageCompression, 1396
 - setStreamingOutput, 1396
 - setTargetColorSpace, 1397
 - setTargetProfile, 1397
- JawsMako::IRedactionAnnotation, 1415
 - create, 1418
 - getInteriorColor, 1418
 - getQuadPoints, 1418
 - setInteriorColor, 1419
 - setQuadPoints, 1419
- JawsMako::IRedactorTransform, 1419
 - create, 1421
- JawsMako::IRendererTransform, 1422
 - create, 1427
 - inkInfoToColorantInfo, 1427
 - setBlackPreservation, 1428
 - setDropSpots, 1428
 - setEnableFormSnappingForVectorReuse, 1429
 - setEnableVectorMode, 1429
 - setFormReconstructionCacheSize, 1429
 - setGenerateFlateCompressedPDFImages, 1430
 - setGenerateMasks, 1430
 - setMarkVectorFlattenedFontsForEmbedding, 1430
 - setMaximumRenderedResultPixels, 1431
 - setMergeAllSpots, 1431
 - setMergeSpots, 1431
 - setMinimumFlateCompressedImageSize, 1432
 - setMonochromeMode, 1432
 - setPreserveAllSpots, 1432
 - setPreserveSpots, 1433
 - setRasterFallbackResolution, 1433
 - setRasterFallbackThreshold, 1433
 - setRasterImageReuseCacheSize, 1434
 - setRenderTransparentNodesOnPageGroupMismatch, 1434
 - setSpotHalftone, 1434
 - setThresholdHalftone, 1435
 - setUseImageResolutionForRenderingWherePossible, 1435
 - setVectorAndTextResolution, 1435
 - setVectorAreaFormReuseCacheSize, 1436
 - setVectorAreaSourceReuseCacheSize, 1436
 - setVectorImageReuseCacheSize, 1436
- JawsMako::ISeparator, 1438
 - create, 1439
 - getNumSeparations, 1440
 - getSeparation, 1440
 - separate, 1441
 - setEnableVectorMode, 1441
 - setRasterFallbackThreshold, 1441
- JawsMako::IShapeAnnotation, 1446
 - create, 1449
 - getInteriorColor, 1450
 - getRectInset, 1450
 - setInteriorColor, 1450
- JawsMako::ISkiaRenderer, 1451

- create, 1452
- drawNode, 1452
- flushCaches, 1452
- JawsMako::ISoftMaskConverterTransform, 1453
- JawsMako::ISoundAnnotation, 1454
 - getSoundAsWav, 1457
- JawsMako::IStampAnnotation, 1458
 - create, 1461
 - getIntent, 1461
 - getName, 1461
 - setIntent, 1462
 - setName, 1462
- JawsMako::IStrokerTransform, 1462
 - create, 1465
 - setupForPDFStyleOutput, 1465
 - setupForXPSStyleOutput, 1466
- JawsMako::IStructure, 1466
 - tagNode, 1468
- JawsMako::IStructureElement, 1468
 - createMarkedContentReferencePair, 1471
 - getAttributes, 1471
 - getObjectStore, 1471
 - setAttributes, 1471
- JawsMako::IStructureElementChild, 1472
- JawsMako::IStructureElementReference, 1473
- JawsMako::IStructureElementReferenceChild, 1474
- JawsMako::IStructureMarkedContentReferenceChild, 1475
- JawsMako::IStructureObjectReferenceChild, 1477
- JawsMako::ISVGGenerator, 1478
 - create, 1480
 - generateSVG, 1480, 1481
 - getResource, 1481
 - getResources, 1481
 - setEnableImageDownsampling, 1482
 - setOptionalContent, 1482
 - setOptionalContentUsage, 1482
 - setPageResolutionCallback, 1482
 - setTargetResolutionCallback, 1482
 - setVisibleZeroWidthLines, 1483
- JawsMako::ISVGGenerator::CResourceEntry, 131
- JawsMako::ITextAnnotation, 1484
 - create, 1487
 - getIconName, 1487
 - getOpen, 1487
 - setIconName, 1488
 - setOpen, 1488
- JawsMako::ITextMarkupAnnotation, 1488
 - getQuadPoints, 1491
 - setQuadPoints, 1491
- JawsMako::ITextRun, 1492
 - getBoundsOnPage, 1493
 - getLocalBounds, 1493
- JawsMako::ITextSearch, 1494
 - create, 1495
- JawsMako::ITextSelect, 1496
 - create, 1497
- JawsMako::IThreads, 1497
- JawsMako::ITransform, 1499
 - setProgressMonitor, 1501
 - transform, 1501, 1502
 - transformPage, 1504
- JawsMako::ITransformChain, 1504
 - create, 1506
 - getTransforms, 1506
 - pushTransform, 1506
 - pushTransformFront, 1507
 - removeTransform, 1507
 - transform, 1507, 1508
 - transformPage, 1508
- JawsMako::IType3UnpackerTransform, 1508
 - create, 1510
- JawsMako::IUnicodeHelper, 1511
- JawsMako::IWidgetAnnotation, 1512
 - createBasicAppearances, 1517
 - createButton, 1517
 - createCheckBox, 1518
 - createChoiceField, 1519
 - createRadioButtons, 1519
 - createTextField, 1520
 - getActionsDictionary, 1521
 - getAdditionalActionsDictionary, 1521
 - getAppearanceCharacteristics, 1521
 - getDefaultAppearanceString, 1521
 - getDefaultValue, 1522
 - getFieldFlags, 1522
 - getFieldType, 1522
 - getFirstVisibleOption, 1523
 - getHighlightMode, 1523
 - getOptions, 1523
 - getPartialName, 1523
 - getQuadding, 1524
 - getSelectedOptions, 1524
 - getValue, 1524
 - setActionsDictionary, 1524
 - setAdditionalActionsDictionary, 1525
 - setAppearanceCharacteristics, 1525
 - setDefaultAppearanceString, 1525
 - setDefaultValue, 1525, 1526
 - setFieldFlags, 1526
 - setFirstVisibleOption, 1526
 - setHighlightMode, 1526
 - setOptions, 1527
 - setPartialName, 1527
 - setQuadding, 1527
 - setSelectedOptions, 1527
 - setValue, 1528
- JawsMako::IWidgetAppearanceCharacteristics, 1528
 - create, 1530
 - getAlternateCaption, 1530
 - getBackgroundColor, 1531
 - getBorderColor, 1531
 - getCaption, 1531
 - getRolloverCaption, 1531
 - getRotation, 1532
 - setAlternateCaption, 1532

- setBackgroundColor, [1532](#)
- setBorderColor, [1532](#)
- setCaption, [1533](#)
- setRolloverCaption, [1533](#)
- setRotation, [1533](#)
- JawsMako::IXAMLGenerator, [1534](#)
 - create, [1536](#)
 - generateXAML, [1537](#)
 - generateXAMLForAppearance, [1537](#), [1538](#)
 - generateXAMLForPageAndAnnotationAppearances, [1538](#)
 - generateXAMLForPageAnnotationAppearances, [1539](#)
 - getResource, [1539](#)
 - getResources, [1539](#)
 - setColorImageMaxResolution, [1540](#)
 - setGrayImageMaxResolution, [1540](#)
 - setJPEGQuality, [1540](#)
 - setMergeFonts, [1541](#)
 - setMonoImageMaxResolution, [1541](#)
 - setPreferredColorImageFormat, [1541](#)
 - setPreferredGrayImageFormat, [1542](#)
 - setPreferredMonoImageFormat, [1542](#)
 - setRenderResolution, [1542](#)
 - setSubsetFonts, [1542](#)
 - setTargetColorSpace, [1543](#)
 - setTargetProfile, [1543](#)
- JawsMako::IXAMLGenerator::CAnnotationXAML, [108](#)
- JawsMako::IXAMLGenerator::CResourceEntry, [131](#)
- JawsMako::IXPSInput, [1543](#)
 - create, [1545](#)
 - openStreaming, [1545](#)
- JawsMako::IXPSOutput, [1546](#)
 - create, [1549](#)
 - setColorImageMaxResolution, [1550](#)
 - setGrayImageMaxResolution, [1550](#)
 - setJPEGQuality, [1551](#)
 - setMergeFonts, [1551](#)
 - setMonoImageMaxResolution, [1551](#)
 - setPreferredColorImageFormat, [1551](#)
 - setPreferredGrayImageFormat, [1552](#)
 - setPreferredMonoImageFormat, [1552](#)
 - setRenameFonts, [1552](#)
 - setRenderResolution, [1553](#)
 - setSubsetFonts, [1553](#)
 - setTargetColorSpace, [1553](#)
 - setTargetProfile, [1553](#)
- JM_DUPLICATE_FIELD_PARTIAL_NAME
 - Error handling, [50](#)
- JM_ERR_ANNOTATION_NOT_FOUND
 - Error handling, [50](#)
- JM_ERR_APPEARANCE_NOT_FOUND
 - Error handling, [50](#)
- JM_ERR_ASSEMBLY_WRITE_FORBIDDEN
 - Error handling, [50](#)
- JM_ERR_ATTEMPTED_WRITE_ON_OPEN_INPUT
 - Error handling, [51](#)
- JM_ERR_BAD_CONFIGURATION
 - Error handling, [50](#)
- JM_ERR_BAD_UNICODE_CMAP
 - Error handling, [51](#)
- JM_ERR_DIRECTORY_DOESNT_EXIST
 - Error handling, [50](#)
- JM_ERR_DOCUMENT_NOT_FOUND
 - Error handling, [50](#)
- JM_ERR_DUPLICATE_WIDGET
 - Error handling, [50](#)
- JM_ERR_FONT_NOT_FOUND
 - Error handling, [50](#)
- JM_ERR_FORM_FIELD_NOT_FOUND
 - Error handling, [50](#)
- JM_ERR_GENERAL
 - Error handling, [50](#)
- JM_ERR_IJPDS
 - Error handling, [51](#)
- JM_ERR_INCORRECT_PDF_OBJECT_TYPE
 - Error handling, [51](#)
- JM_ERR_INFORMATION_NOT_AVAILABLE
 - Error handling, [50](#)
- JM_ERR_INVALID_OPTIONAL_CONTENT
 - Error handling, [51](#)
- JM_ERR_INVALID_PDF_OBJECT
 - Error handling, [51](#)
- JM_ERR_OPTIONAL_CONTENT_GROUP_NOT_FOUND
 - Error handling, [51](#)
- JM_ERR_PAGE_NOT_FOUND
 - Error handling, [50](#)
- JM_ERR_PCL
 - Error handling, [51](#)
- JM_ERR_PCLXL
 - Error handling, [51](#)
- JM_ERR_PDF_OBJECT_NOT_FOUND
 - Error handling, [51](#)
- JM_ERR_PJL
 - Error handling, [51](#)
- JM_ERR_RANGE_ERROR
 - Error handling, [50](#)
- JM_ERR_RESOURCE_NOT_FOUND
 - Error handling, [51](#)
- JM_ERR_REVERT_FAILED
 - Error handling, [50](#)
- JM_ERR_SYMBOLSET_NOT_FOUND
 - Error handling, [51](#)
- JM_ERR_TARGET_NOT_FOUND
 - Error handling, [50](#)
- JM_ERR_TOO_MANY_PDFOUT_WRITERS
 - Error handling, [50](#)
- JM_ERR_TTF_INSTRUCTIONS
 - Error handling, [51](#)
- JM_ERR_UNSUPPORTED
 - Error handling, [50](#)
- JM_ERR_WIDGET_NOT_FOUND
 - Error handling, [50](#)
- key
 - JawsMako::IPJLParser::CPjiAttributeValue, [130](#)

- Layout, 100
- layout
 - JawsMako::ILayout, 1125, 1126
- length
 - IRASStream, 1411
- lineTo
 - IDOMPathGeometryBuilder, 741
- loadFromFile
 - IDOMJobTkContent, 631
- loadFromInitString
 - IDOMJobTkContent, 632
- loadFromStream
 - IDOMJobTkContent, 632
- Logical structure, 75
- lookupFarReference
 - JawsMako::IDocument, 311
 - JawsMako::IPage, 1226
- makeNodeOptional
 - JawsMako::IOptionalContent, 1178
- mapping
 - IDOMFontOpenType::CCIDMap, 109
- match
 - JawsMako::ILayoutFontWeight, 1133
- matchesReference
 - JawsMako::IAnnotation, 229
- memutils.h, 1558
- Metadata, 78
 - eCoreProperties, 79
 - eDocumentInfo, 79
 - eMetadataTypeCnt, 79
 - ePageView, 79
 - ePDFInfo, 79
 - eType, 79
 - eViewerPreferences, 79
 - eXmpContainer_Alt, 80
 - eXmpContainer_Bag, 80
 - eXmpContainer_None, 80
 - eXmpContainer_Seq, 80
 - eXmpContainerType, 79
- modifier
 - JawsMako::IPJLParser::CPjlAttributeValue, 130
- moveTo
 - IDOMPathGeometryBuilder, 742
- newReference
 - JawsMako::IPDFObjectStore, 1312
- nextEnumerationItem
 - IDOMGlyphIDEnumerator, 523
 - IFontHeaderWriteSegmentBlockEnumerator, 990
 - IFontPCL5WriteSegmentBlockEnumerator, 993
- Nodes, 62
 - eDOMAnnotationAppearanceNode, 63
 - eDOMCanvasNode, 63
 - eDOMCharPathGroupNode, 63
 - eDOMContentRootNode, 63
 - eDOMDocumentNode, 63
 - eDOMDocumentSequenceNode, 63
 - eDOMFixedDocumentNode, 63
 - eDOMFixedPageNode, 63
 - eDOMFormInstanceNode, 63
 - eDOMFormNode, 63
 - eDOMFragmentNode, 63
 - eDOMGlyphNode, 63
 - eDOMGlyphsNode, 63
 - eDOMGroupNode, 63
 - eDOMJobTkContentNode, 63
 - eDOMJobTkGenericCharacterDataNode, 63
 - eDOMJobTkGenericNodeNode, 63
 - eDOMJobTkNodeNode, 63
 - eDOMJobTkValueNode, 63
 - eDOMNodeType, 63
 - eDOMNodeTypeCnt, 63
 - eDOMOutlineEntryNode, 63
 - eDOMOutlineNode, 63
 - eDOMPageNode, 63
 - eDOMPathNode, 63
 - eDOMRefNode, 63
 - eDOMTileNode, 63
 - eDOMTransparencyGroupNode, 63
 - eDOMVisualRootNode, 63
- normalizeStops
 - IDOMGradientBrush, 553
- notifyOnDestruct
 - IDOMNode, 694
- objclassid.h, 1558
- Object Hashing, 80
- offset
 - IDOMShape, 889
- open
 - IEDLStream, 975
 - JawsMako::IInput, 1083
- openIncremental
 - JawsMako::IPDFInput, 1294, 1295
- openStreaming
 - JawsMako::IXPSInput, 1545
- openWriter
 - JawsMako::IOutput, 1198
- operator=
 - CClassID, 113
 - CTransformMatrix< TItem >, 142
 - EDLQName, 219
- operator==
 - EDLQName, 219
- Optional Content Groups, 73
- optionalcontent.h, 1558
- ordering
 - IDOMFontOpenType::CCIDMap, 109
- Other DOM Objects, 80
- Others, 97
- Outlines, 76
 - eActionArray, 78
 - eActionGoTo3DView, 78
 - eActionGoToE, 77
 - eActionGoToR, 77
 - eActionHide, 77
 - eActionImportData, 77

- eActionJavaScript, [78](#)
- eActionLaunch, [77](#)
- eActionMovie, [77](#)
- eActionNamed, [77](#)
- eActionRendition, [78](#)
- eActionResetForm, [77](#)
- eActionSetOCGState, [78](#)
- eActionSound, [77](#)
- eActionSubmitForm, [77](#)
- eActionThread, [77](#)
- eActionTrans, [78](#)
- eExternal, [77](#)
- eInternal, [77](#)
- ePage, [77](#)
- ePageRect, [77](#)
- eTargetType, [77](#)
- eTextStyle, [78](#)
- eTextStyleBold, [78](#)
- eTextStyleBoldItalic, [78](#)
- eTextStyleItalic, [78](#)
- eTextStyleNone, [78](#)
- Output, [82](#)
- overrideMaximumICCVersion
 - JawsMako::IPDFOutput, [1322](#)
- pageExists
 - JawsMako::IDocument, [311](#)
- pageNums
 - JawsMako::IPDFInput::CPdfScannedInk, [129](#)
- Pages, [71](#)
- parse
 - JawsMako::IPJLParser, [1364](#)
- pcl5input.h, [1558](#)
- pcl5output.h, [1558](#)
- pclxlinput.h, [1558](#)
- pclxloutput.h, [1558](#)
- PDF Objects, [73](#)
 - ePDFObjectType, [75](#)
 - ePOTArray, [75](#)
 - ePOTBoolean, [75](#)
 - ePOTDictionary, [75](#)
 - ePOTFarReference, [75](#)
 - ePOTInteger, [75](#)
 - ePOTName, [75](#)
 - ePOTNull, [75](#)
 - ePOTOperator, [75](#)
 - ePOTReal, [75](#)
 - ePOTReference, [75](#)
 - ePOTStream, [75](#)
 - ePOTString, [75](#)
- PDF-specific, [71](#)
 - eAnnotationEditingAllowed, [72](#)
 - eContentAccessibilityExtractionAllowed, [72](#)
 - eCopyingAllowed, [72](#)
 - eDocumentAssemblyAllowed, [72](#)
 - eEditingAllowed, [72](#)
 - eEverythingAllowed, [72](#)
 - eFormFillingAllowed, [72](#)
 - eHighQualityPrintAllowed, [72](#)
 - ePermissionsFlags, [72](#)
 - ePrintAllowed, [72](#)
- pdfinput.h, [1558](#)
- pdfobjects.h, [1558](#)
- pdfoutput.h, [1558](#)
- pdfpage.h, [1558](#)
- pixelStride
 - JawsMako::JawsRenderer::CFrameBufferInfo, [118](#)
- Platform-dependent, [87](#)
 - edlExclusiveMakeTempDir, [88](#)
 - edlFopen, [89](#)
 - edlGetProcessId, [89](#)
 - edlGetTemporaryDirectory, [89](#)
 - edlMakeTempDir, [89](#)
 - edlMakeTempDirProvidingSubDirPath, [90](#)
 - edlMkdir, [90](#)
 - edlRmdir, [91](#)
 - edlSnprintf, [91](#)
 - edlStrtod, [91](#)
 - edlVsprintf, [92](#)
- platform.h, [1558](#)
- points
 - IDOMShadingPatternType4567Brush::CMeshEntry, [122](#)
- PointTpl< PointType >, [1555](#)
- postMul
 - CTransformMatrix< TItem >, [142](#)
- ppminput.h, [1558](#)
- preMul
 - CTransformMatrix< TItem >, [143](#)
- prninput.h, [1558](#)
- psinput.h, [1558](#)
- pushBack
 - IPushbackStream, [1398](#)
- pushFilter
 - IDOMFilteredImage, [445](#)
- pushTransform
 - JawsMako::ITransformChain, [1506](#)
- pushTransformFront
 - JawsMako::ITransformChain, [1507](#)
- put
 - IDOMResourceDictionary, [840](#)
 - JawsMako::IPDFArray, [1275](#)
 - JawsMako::IPDFDictionary, [1284](#), [1285](#)
- putInteger
 - JawsMako::IPDFDictionary, [1285](#)
- putReal
 - JawsMako::IPDFDictionary, [1285](#)
- PValue, [1555](#)
- quadCurveTo
 - IDOMPathGeometryBuilder, [742](#)
- read
 - IInputStream, [1103](#)
- readPdfObject
 - JawsMako::IDocument, [311](#)
- readPdfTrailerDictionary
 - JawsMako::IDocument, [312](#)

- readScanLine
 - ImageFrame, 1053
- recommendedVisibilityForCategories
 - JawsMako::IOptionalContentGroupUsage, 1190
- registerClass
 - IEDLClassFactory, 966
- registerNamedClass
 - IEDLClassFactory, 966
- registerNumbers
 - IDOMCatalog, 357
- registerObject
 - IDOMCatalog, 357
- registry
 - IDOMFontOpenType::CCIDMap, 109
- release
 - JawsMako::IPage, 1226
- remove
 - JawsMako::IPDFArray, 1275
- removeAnnotation
 - JawsMako::IPage, 1227
- removeChildField
 - JawsMako::IFormField, 1020
- removeChildWidget
 - JawsMako::IFormField, 1020, 1021
- removeDocument
 - JawsMako::IDocumentAssembly, 318
- removeField
 - JawsMako::IForm, 1009
- removeFont
 - JawsMako::IDistiller, 287
- removePage
 - JawsMako::IDocument, 312
- removeProperty
 - IDOMMetadata, 683
 - IDOMNode, 694
- removeTransform
 - JawsMako::ITransformChain, 1507
- removeWidget
 - JawsMako::IForm, 1010
- render
 - JawsMako::IJawsRenderer, 1111
- renderAntiAliased
 - JawsMako::IJawsRenderer, 1112
- Rendering, 98
- renderMonochrome
 - JawsMako::IJawsRenderer, 1113
- renderMonochromeToFrameBuffer
 - JawsMako::IJawsRenderer, 1113
- renderScreened
 - JawsMako::IJawsRenderer, 1114
- renderScreenedToFrameBuffers
 - JawsMako::IJawsRenderer, 1115
- renderSeparations
 - JawsMako::IJawsRenderer, 1116
- renderSeparationsToFrameBuffers
 - JawsMako::IJawsRenderer, 1117
- renderToFrameBuffer
 - JawsMako::IJawsRenderer, 1119
- renderToFrameBufferPadAndReverse
 - JawsMako::IJawsRenderer, 1120
- replaceChild
 - IDOMNode, 695
- revert
 - JawsMako::IPage, 1227
- rotate
 - CTransformMatrix< TItem >, 143
 - JawsMako::IAnnotation, 230
- run
 - IRunnable, 1438
- save
 - JawsMako::IPDFPageInserter, 1349
- scale
 - CTransformMatrix< TItem >, 143
- scanPdfForFonts
 - JawsMako::IPDFInput, 1295, 1296
- scanPdfForInks
 - JawsMako::IPDFInput, 1296, 1297
- separate
 - JawsMako::ISeparator, 1441
- Separator, 105
- separator.h, 1558
- set
 - CTransformMatrix< TItem >, 144
 - IDOMNodeFlags, 699
- setActionOrDest
 - JawsMako::ILinkAnnotation, 1159
- setActionsDictionary
 - JawsMako::IWidgetAnnotation, 1524
- setAdditionalActionsDictionary
 - JawsMako::IWidgetAnnotation, 1525
- setAllowedPermissionsFlags
 - JawsMako::IOutput, 1199
- setAllowMaskedResults
 - JawsMako::ImageMergerTransform, 1060
- setAllowOptionalContentUpdateDuringWrite
 - JawsMako::IPDFOutput, 1322
- setAllowRestrictedFonts
 - JawsMako::IPDFOutput, 1323
- setAllowReusableImageStreams
 - JawsMako::IPSOOutput, 1393
- setAlpha
 - IDOMColor, 378
- setAlternateBaseFont
 - JawsMako::IPDFInput, 1298
- setAlternateCaption
 - JawsMako::IWidgetAppearanceCharacteristics, 1532
- setAlternateImageErrorHandler
 - JawsMako::IPDFOutput, 1323
- setAlternateImages
 - IDOMImageBrush, 591
- setAlwaysEmbedFonts
 - JawsMako::IPDFOutput, 1324
- setAntiAlias
 - IDOMShadingPatternBrush, 850
- setAppearanceCharacteristics

- JawsMako::IWidgetAnnotation, 1525
- setAscii85EncodePages
 - JawsMako::IDistiller, 288
- setAttributeHandler
 - JawsMako::IPCLXInput, 1261
- setAttributes
 - JawsMako::IStructureElement, 1471
- setAuthor
 - JawsMako::IMarkupAnnotation, 1169
- setAutomationPropertiesHelpText
 - IDOMCanvas, 350
 - IDOMPathNode, 761
- setAutomationPropertiesName
 - IDOMCanvas, 351
 - IDOMPathNode, 761
- setAutoRotatePages
 - JawsMako::IDistiller, 288
 - JawsMako::IPDFOutput, 1324
- setBackground-color
 - IDOMShadingPatternBrush, 850
 - JawsMako::IWidgetAppearanceCharacteristics, 1532
- setBBox
 - IDOMShadingPatternBrush, 850
 - IDOMTilingPatternBrush, 923
- setBidiLevel
 - IDOMGlyphs, 541
- setBitsPerComponent
 - IDOMShadingPatternType4567Brush, 880
- setBitsPerCoordinate
 - IDOMShadingPatternType4567Brush, 880
- setBitsPerFlag
 - IDOMShadingPatternType4567Brush, 881
- setBlackPointCompensation
 - IDOMGlyphs, 541
 - IDOMPathNode, 762
 - IDOMTransparencyGroup, 935
- setBlackPreservation
 - JawsMako::IColorConverterTransform, 251
 - JawsMako::IJawsRenderer, 1121
 - JawsMako::IRendererTransform, 1428
- setBleedBox
 - IDOMFixedPage, 453
- setBlendMode
 - IDOMFormInstance, 504
 - IDOMGlyphs, 541
 - IDOMPathNode, 762
 - IDOMTransparencyGroup, 935
- setBlockNotdefGlyphs
 - JawsMako::IPDFOutput, 1324
- setBoolProperty
 - IDOMImageProperties, 610
- setBorder
 - JawsMako::IAnnotation, 230
- setBorderColor
 - JawsMako::IWidgetAppearanceCharacteristics, 1532
- setBottom
 - IDOMPageRectTarget, 716
- setBounds
 - IDOMForm, 497
- setBPS
 - ImageFrameWriter, 1056
- setBrush
 - IDOMMaskedBrush, 676
- setCallback
 - JawsMako::IProgressTick, 1382
- setCaption
 - JawsMako::IWidgetAppearanceCharacteristics, 1533
- setCaretStops
 - IDOMGlyphs, 542
- setCenter
 - IDOMRadialGradientBrush, 819
- setCharacterData
 - IDOMJobTkGenericCharacterData, 638
- setCharPathType
 - IDOMCharPathGroup, 364
- setCidMap
 - IDOMFontOpenType, 471
- setClip
 - IDOMGlyphs, 542
 - IDOMGroup, 564
 - IDOMPathNode, 762
- setClippedGroup
 - IDOMCharPathGroup, 365
- setClusters
 - IDOMGlyphs, 543
- setColor
 - IDOMGradientStop, 557
 - IDOMSolidColorBrush, 901
 - JawsMako::IAnnotation, 230
 - JawsMako::ILayoutTextRun, 1149
- setColored
 - IDOMGlyph, 520
- setColorImageCompression
 - JawsMako::IDistiller, 288
- setColorImageDownsampling
 - JawsMako::IDistiller, 290
- setColorImageMaxResolution
 - JawsMako::IPDFOutput, 1324
 - JawsMako::IPSOOutput, 1393
 - JawsMako::IXAMLGenerator, 1540
 - JawsMako::IXPSOutput, 1550
- setColorImageResolution
 - JawsMako::IDistiller, 290
- setColorInterpolationMode
 - IDOMGradientBrush, 554
- setColorJPEGQuality
 - JawsMako::IDistiller, 290
- setColorQFactor
 - JawsMako::IDistiller, 290
- setColorSpace
 - IDOMColor, 378, 379
 - IDOMShadingPatternBrush, 851
 - IDOMTransparencyGroup, 935

- IImageFrameWriter, 1057
- setColorTIFFCompression
 - JawsMako::ImageEncoderTransform, 1046
- setCommentMonitor
 - JawsMako::IDistiller, 291
- setComponentValue
 - IDOMColor, 379
- setCompressObjects
 - JawsMako::IDistiller, 291
 - JawsMako::IPDFOutput, 1325
- setCompressPages
 - JawsMako::IDistiller, 292
 - JawsMako::IPDFOutput, 1325
- setContent
 - IDOMJobTk, 621
- setContentBox
 - IDOMFixedPage, 454
- setContents
 - JawsMako::IAnnotation, 231
- setConvertAllColors
 - JawsMako::IPDFOutput, 1325
- setConvertAllObjectsToTargetColorSpace
 - JawsMako::IPSOOutput, 1394
- setConvertCMYKImagesToRGB
 - JawsMako::IDistiller, 292
- setConvertColorsInsideLuminositySoftMasks
 - JawsMako::IColorConverterTransform, 252
- setConvertGray
 - JawsMako::IPDFOutput, 1326
- setConvertThroughTransparencyGroupColorSpaces
 - JawsMako::IColorConverterTransform, 252
- setCreationTime
 - JawsMako::IMarkupAnnotation, 1169
- setCropBox
 - IDOMFixedPage, 454
- setCropToBBox
 - JawsMako::IDistiller, 292
- setDataSource
 - IDOMShadingPatternType4567Brush, 881
- setDay
 - IEDLTime, 986
- setDecode
 - IDOMShadingPatternType4567Brush, 881
- setDefaultAppearanceString
 - JawsMako::IFormField, 1021
 - JawsMako::IWidgetAnnotation, 1525
- setDefaultBlackPointCompensation
 - IColorManager, 271
- setDefaultBlackPreservation
 - IColorManager, 271
- setDefaultColor
 - JawsMako::ILayoutParagraph, 1142
- setDefaultConfiguration
 - JawsMako::IOptionalContent, 1178
- setDefaultCopies
 - JawsMako::IPCL5Input, 1249
 - JawsMako::IPCLXLInput, 1261
- setDefaultDuplex
 - JawsMako::IPCL5Input, 1249
 - JawsMako::IPCLXLInput, 1262
- setDefaultDuplexBindingMode
 - JawsMako::IPCL5Input, 1249
 - JawsMako::IPCLXLInput, 1262
- setDefaultFont
 - JawsMako::ILayoutParagraph, 1142
- setDefaultFontSize
 - JawsMako::ILayoutParagraph, 1143
- setDefaultFormCJKLanguage
 - JawsMako::IPDFOutput, 1326
- setDefaultLandscape
 - JawsMako::IPCL5Input, 1249
 - JawsMako::IPCLXLInput, 1262
- setDefaultManualFeed
 - JawsMako::IPCL5Input, 1249
 - JawsMako::IPCLXLInput, 1262
- setDefaultPanoseStyle
 - JawsMako::IDistiller, 292
- setDefaultPaperSize
 - JawsMako::IPCL5Input, 1250
 - JawsMako::IPCLXLInput, 1262
- setDefaultRenderingIntent
 - JawsMako::IPDFInput, 1298
- setDefaultsFromPjl
 - JawsMako::IPCL5Input, 1250
 - JawsMako::IPCLXLInput, 1263
- setDefaultValue
 - JawsMako::IFormField, 1021
 - JawsMako::IWidgetAnnotation, 1525, 1526
- setDescription
 - IDOMOutlineEntry, 710
- setDeviceCMYKIntercept
 - IColorManager, 272
- setDeviceFontName
 - IDOMGlyphs, 543
- setDeviceGrayIntercept
 - IColorManager, 272
- setDeviceNErrorHandling
 - JawsMako::IPDFOutput, 1326
- setDeviceNHandling
 - JawsMako::IColorConverterTransform, 252
- setDeviceRGBIntercept
 - IColorManager, 272
- setDocumentInformationString
 - JawsMako::IDistiller, 293
- setDomain
 - IDOMShadingPatternType1Brush, 856
 - IDOMShadingPatternType2Brush, 863
 - IDOMShadingPatternType3Brush, 871
- setDOMid
 - IDOMNode, 695
- setDownsampleMaskedImages
 - JawsMako::IImageDownsamplerTransform, 1041
 - JawsMako::IPDFOutput, 1327
- setDropSpots
 - JawsMako::IRendererTransform, 1428
- setDX

- CTransformMatrix< TItem >, 144
- setDY
 - CTransformMatrix< TItem >, 144
- setEdgeMode
 - IDOMCanvas, 351
 - IDOMPathNode, 762
- setEmbedBase14Fonts
 - JawsMako::IDistiller, 293
 - JawsMako::IPDFOutput, 1327
- setEmbedded
 - IDOMFontOpenType, 471
- setEmbedDiskBasedCidFonts
 - JawsMako::IDistiller, 293
- setEmbedFonts
 - JawsMako::IDistiller, 294
 - JawsMako::IPDFOutput, 1327
- setEmbedIdentityCidFonts
 - JawsMako::IPDFOutput, 1327
- setEmit30CmapSubtableForSymbolicTrueTypeFonts
 - JawsMako::IPDFOutput, 1328
- setEmitPjl
 - JawsMako::IPCL5Output, 1255
 - JawsMako::IPCLXLOutput, 1267
- setEnableFormSnappingForVectorReuse
 - JawsMako::IRendererTransform, 1429
- setEnableImageDownsampling
 - JawsMako::ISVGGenerator, 1482
- setEnableIncrementalOutput
 - JawsMako::IPDFOutput, 1328
- setEnableRasterCompression
 - JawsMako::IJawsRenderer, 1122
- setEnableTrueTypeNotDef
 - JawsMako::IJawsRenderer, 1122
 - JawsMako::IPCL5Output, 1255
 - JawsMako::IPDFOutput, 1329
- setEnableVectorMode
 - JawsMako::IRendererTransform, 1429
 - JawsMako::ISeparator, 1441
- setEncryption
 - JawsMako::IDistiller, 294
 - JawsMako::IPDFOutput, 1329
- setEndCircleCenter
 - IDOMShadingPatternType3Brush, 871
- setEndCircleRadius
 - IDOMShadingPatternType3Brush, 872
- setEndPoint
 - IDOMLinearGradientBrush, 669
 - IDOMShadingPatternType2Brush, 863
- setEpilog
 - JawsMako::IDistiller, 295
- setExpanded
 - IDOMOutlineEntry, 710
- setExtend
 - IDOMShadingPatternType2Brush, 864
 - IDOMShadingPatternType3Brush, 872
- setExtendedGraphicsStateErrorHandling
 - JawsMako::IPDFOutput, 1330
- setFactory
 - ISession, 1445
- setFailOnFontFallback
 - JawsMako::IPDFInput, 1298
- setFieldFlags
 - JawsMako::IFormField, 1022
 - JawsMako::IWidgetAnnotation, 1526
- setFile
 - IDOMActionLaunch, 324
- setFill
 - IDOMGlyphs, 543
 - IDOMPathNode, 763
- setFillOverprints
 - IDOMGlyphs, 543
 - IDOMPathNode, 763
- setFillRule
 - IDOMPathGeometry, 737
- setFirstVisibleOption
 - JawsMako::IWidgetAnnotation, 1526
- setFitType
 - IDOMPageRectTarget, 717
- setFlags
 - JawsMako::IAnnotation, 231
- setFlattenRops
 - JawsMako::IPCL5Input, 1250
 - JawsMako::IPCLXLInput, 1263
- setFont
 - IDOMGlyphs, 544
- setFontDevicePath
 - JawsMako::IDistiller, 295
- setFontIndex
 - IDOMGlyphs, 544
- setFontRenderingEmSize
 - IDOMGlyphs, 544
- setFontSource
 - IDOMFont, 460
- setForceEmitPageGroup
 - JawsMako::IPDFOutput, 1331
- setForceMediaBoxOriginZero
 - JawsMako::IPDFOutput, 1331
- setForm
 - IDOMFormInstance, 504
- setFormReconstructionCacheSize
 - JawsMako::IRendererTransform, 1429
- setFunction
 - IDOMShadingPatternBrush, 851
- setGenerateFlateCompressedPDFImages
 - JawsMako::IRendererTransform, 1430
- setGenerateMasks
 - JawsMako::IRendererTransform, 1430
- setGradientOrigin
 - IDOMRadialGradientBrush, 820
- setGradientStops
 - IDOMGradientBrush, 554
- setGrayImageCompression
 - JawsMako::IDistiller, 295
- setGrayImageDownsampling
 - JawsMako::IDistiller, 296
- setGrayImageMaxResolution

- JawsMako::IPDFOutput, 1331
- JawsMako::IPSOOutput, 1394
- JawsMako::IXAMLGenerator, 1540
- JawsMako::IXPSOutput, 1550
- setGrayImageResolution
 - JawsMako::IDistiller, 296
- setGrayJPEGQuality
 - JawsMako::IDistiller, 296
- setGrayQFactor
 - JawsMako::IDistiller, 296
- setGrayTIFFCompression
 - JawsMako::ImageEncoderTransform, 1047
- setHalftone
 - JawsMako::IDistiller, 297
- setHasCustomAdvance
 - IDOMGlyph, 520
- setHeight
 - IDOMFixedPage, 454
 - ImageFrameWriter, 1057
- setHighlightMode
 - JawsMako::ILinkAnnotation, 1160
 - JawsMako::IWidgetAnnotation, 1526
- setHiResBBox
 - JawsMako::IDistiller, 297
- setICCProfile
 - IDOMColorSpaceICCBased, 408
 - IDOMImageBrush, 591
- setIconName
 - JawsMako::ITextAnnotation, 1488
- setIgnoreMatchingDeviceIntercept
 - JawsMako::IJawsRenderer, 1122
- setIgnorePrescribe
 - JawsMako::IPCL5Input, 1251
 - JawsMako::IPRNInput, 1376
- setImageCompression
 - JawsMako::IPCL5Output, 1255
- setImageExtraChannelType
 - ImageFrameWriter, 1057
- setImageSource
 - IDOMImageBrush, 591
- setImproveGlyphSelectionForUnembeddedIdentityFonts
 - JawsMako::IPDFInput, 1298
- setIncludeMarginsWhenSelectingPaper
 - JawsMako::IPCL5Output, 1256
- setIndices
 - IDOMGlyphs, 544
- setInkList
 - JawsMako::ILinkAnnotation, 1080
- setIntent
 - JawsMako::IStampAnnotation, 1462
- setInteriorColor
 - JawsMako::ILineAnnotation, 1155
 - JawsMako::IRedactionAnnotation, 1419
 - JawsMako::IShapeAnnotation, 1450
- setIntProperty
 - IDOMImageProperties, 610
- setIsCharPath
 - IDOMGlyphs, 545
- setIsClosed
 - IDOMPathFigure, 726
- setIsFilled
 - IDOMPathFigure, 727
- setIsInvisible
 - IDOMGlyphs, 545
- setIsIsolated
 - IDOMTransparencyGroup, 937
- setIsKnockout
 - IDOMTransparencyGroup, 937
- setIsLargeArc
 - IDOMArcSegment, 332
- setIsMap
 - IDOMExternalTarget, 440
- setIsSideways
 - IDOMGlyphs, 545
- setIsStroked
 - IDOMPathSegment, 773
- setJobTicket
 - IDOMJobTkOwner, 654
 - JawsMako::IDistiller, 298
- setJobTkNodeType
 - IDOMJobTkNode, 649
- setJPEGChromaSubsampling
 - JawsMako::IPDFOutput, 1332
- setJPEGQuality
 - JawsMako::ImageEncoderTransform, 1047
 - JawsMako::IPDFOutput, 1332
 - JawsMako::IPSOOutput, 1394
 - JawsMako::IXAMLGenerator, 1540
 - JawsMako::IXPSOutput, 1551
- setKeepOverprintMode
 - JawsMako::IColorConverterTransform, 253
- setLanguage
 - IDOMCanvas, 351
 - IDOMFixedPage, 454
 - IDOMGlyphs, 545
 - IDOMOutline, 704
 - IDOMOutlineEntry, 710
 - IDOMPathNode, 763
 - JawsMako::IOptionalContentGroupUsage, 1191
- setLeading
 - JawsMako::ILayoutParagraph, 1143
- setLeft
 - IDOMPageRectTarget, 717
- setLetterSpacing
 - JawsMako::ILayoutTextRun, 1150
- setLevel
 - IDOMJobTkContent, 632
- setLinearize
 - JawsMako::IDistiller, 298
 - JawsMako::IPDFOutput, 1333
- setLineEndpoints
 - JawsMako::ILineAnnotation, 1155
- setLineSpacingThreshold
 - JawsMako::IPageLayout, 1237
- setMacParameters
 - IDOMActionLaunch, 325

- setMapDeviceGrayToCMYKBlack
 - IColorManager, 272
- setMarkedContentDetails
 - IDOMGroup, 564
- setMarkVectorFlattenedFontsForEmbedding
 - JawsMako::IRendererTransform, 1430
- setMatrix
 - IDOMForm, 498
 - IDOMShadingPatternType1Brush, 857
- setMaximumRenderedResultPixels
 - JawsMako::IRendererTransform, 1431
- setMediaBox
 - JawsMako::IPage, 1227
- setMediaHandler
 - JawsMako::IPCL5Input, 1251
 - JawsMako::IPCLXLInput, 1264
- setMediaSource
 - JawsMako::IPCL5Output, 1256
- setMergeAllSpots
 - JawsMako::IRendererTransform, 1431
- setMergeFonts
 - JawsMako::IXAMLGenerator, 1541
 - JawsMako::IXPSOutput, 1551
- setMergeSpots
 - JawsMako::IRendererTransform, 1431
- setMeshEntries
 - IDOMShadingPatternType4567Brush, 881
- setMinimumFlateCompressedImageSize
 - JawsMako::IRendererTransform, 1432
- setModificationTime
 - JawsMako::IAnnotation, 231
- setModified
 - IDOMJobTkContent, 633
- setMonochromeMode
 - JawsMako::IRendererTransform, 1432
- setMonoImageCompression
 - JawsMako::IDistiller, 298
- setMonoImageDownsampling
 - JawsMako::IDistiller, 298
- setMonoImageMaxResolution
 - JawsMako::IPDFOutput, 1333
 - JawsMako::IPSOOutput, 1395
 - JawsMako::IXAMLGenerator, 1541
 - JawsMako::IXPSOutput, 1551
- setMonoImageResolution
 - JawsMako::IDistiller, 299
- setMonoTIFFCompression
 - JawsMako::IImageEncoderTransform, 1047
- setMonth
 - IEDLTime, 986
- setMultipleSpaceMode
 - JawsMako::IPageLayout, 1237
- setName
 - EDLQName, 220
 - JawsMako::IStampAnnotation, 1462
- setNamedDestinations
 - JawsMako::IDocument, 312
- setNamespace
 - EDLQName, 220
 - IEDLNamespace, 969
- setNavigateLink
 - IDOMCanvas, 352
 - IDOMGlyphs, 546
 - IDOMPathNode, 764
- setNeedAppearances
 - JawsMako::IForm, 1010
- setNeverEmbedFonts
 - JawsMako::IPDFOutput, 1333
- setNewWindow
 - IDOMActionLaunch, 325
- setNextSibling
 - IDOMNode, 695
- setNonUtf8InkNameFallbackEncoding
 - JawsMako::IPDFOutput, 1333
- setNumber
 - JawsMako::IPageLabel, 1234
- setObjectProperty
 - IDOMImageProperties, 610
- setOffset
 - IDOMGradientStop, 558
- setOpacity
 - IDOMBrush, 338
 - IDOMFormInstance, 505
 - IDOMGlyphs, 546
 - IDOMPathNode, 764
 - IDOMTransparencyGroup, 937
 - JawsMako::IMarkupAnnotation, 1170
- setOpacityMask
 - IDOMFormInstance, 505
 - IDOMGlyphs, 546
 - IDOMPathNode, 764
 - IDOMTransparencyGroup, 937
- setOpen
 - JawsMako::IPopupAnnotation, 1372
 - JawsMako::ITextAnnotation, 1488
- setOpenStream
 - JawsMako::IPCL5Output, 1256
 - JawsMako::IPCLXLOutput, 1267
- setOPI
 - JawsMako::IDistiller, 299
- setOptionalContent
 - JawsMako::ISVGGenerator, 1482
- setOptionalContentDetails
 - IDOMGroup, 564
 - IDOMImageBrush, 592
- setOptionalContentErrorHandling
 - JawsMako::IPDFOutput, 1334
- setOptionalContentUsage
 - JawsMako::ISVGGenerator, 1482
- setOptions
 - JawsMako::IFormField, 1022
 - JawsMako::IWidgetAnnotation, 1527
- setOriginX
 - IDOMGlyphs, 546
- setOriginY
 - IDOMGlyphs, 547

- setOutline
 - JawsMako::IDocument, 312
- setOutputIntent
 - JawsMako::IPDFOutput, 1334
- setOutputIntents
 - JawsMako::IPDFOutput, 1334
- setOverprint
 - JawsMako::IDistiller, 299
- setOverprintMode
 - IDOMGlyphs, 547
 - IDOMPathNode, 764
 - JawsMako::IDistiller, 300
- setOwner
 - IDOMJobTk, 621
- setPA
 - JawsMako::ILinkAnnotation, 1160
- setPageElement
 - JawsMako::IOptionalContentGroupUsage, 1191
- setPageFilterHigh
 - JawsMako::IDistiller, 300
- setPageFilterLow
 - JawsMako::IDistiller, 300
- setPageFilterRange
 - JawsMako::IDistiller, 301
- setPageGroup
 - IDOMFixedPage, 455
- setPageId
 - IDOMPageRectTarget, 717
- setPageLabel
 - JawsMako::IPage, 1228
- setPageResolutionCallback
 - JawsMako::ISVGGenerator, 1482
- setPaintType
 - IDOMTilingPatternBrush, 924
- setPanose
 - JawsMako::IDistiller, 301
- setParameter
 - JawsMako::IDistiller, 301
 - JawsMako::IInput, 1083
- setParentNode
 - IDOMNode, 695
- setPartialName
 - JawsMako::IFormField, 1022
 - JawsMako::IWidgetAnnotation, 1527
- setPassword
 - JawsMako::IPDFInput, 1299
- setPathData
 - IDOMPathNode, 765
- setPatternColor
 - IDOMTilingPatternBrush, 924
- setPdfPropertiesDictionary
 - IDOMForm, 498
 - IDOMImageBrush, 592
- setPdfVersion
 - JawsMako::IDistiller, 302
- setPermanentResourceStore
 - JawsMako::IPCL5Input, 1251
- setPkcs12
 - JawsMako::IPDFInput, 1299
- setPoint
 - IDOMArcSegment, 332
- setPoints
 - JawsMako::IPolyAnnotation, 1368
- setPopup
 - JawsMako::IMarkupAnnotation, 1170
- setPos
 - IRASStream, 1411
- setPosE
 - IRASStream, 1412
- setPreferredColorFormat
 - JawsMako::IImageEncoderTransform, 1047
- setPreferredColorImageCompression
 - JawsMako::IPDFOutput, 1335
 - JawsMako::IPSOOutput, 1395
- setPreferredColorImageFormat
 - JawsMako::IXAMLGenerator, 1541
 - JawsMako::IXPSOutput, 1551
- setPreferredFormat
 - JawsMako::IImageEncoderTransform, 1048
- setPreferredGrayFormat
 - JawsMako::IImageEncoderTransform, 1048
- setPreferredGrayImageCompression
 - JawsMako::IPDFOutput, 1336
 - JawsMako::IPSOOutput, 1396
- setPreferredGrayImageFormat
 - JawsMako::IXAMLGenerator, 1542
 - JawsMako::IXPSOutput, 1552
- setPreferredMonoFormat
 - JawsMako::IImageEncoderTransform, 1048
- setPreferredMonoImageCompression
 - JawsMako::IPDFOutput, 1336
 - JawsMako::IPSOOutput, 1396
- setPreferredMonoImageFormat
 - JawsMako::IXAMLGenerator, 1542
 - JawsMako::IXPSOutput, 1552
- setPreferredRenderedImageCompression
 - JawsMako::IPDFOutput, 1336
- setPrefix
 - IEDLNamespace, 970
 - JawsMako::IPageLabel, 1234
- setPreserveAllSpots
 - JawsMako::IRendererTransform, 1432
- setPreserveSpots
 - JawsMako::IRendererTransform, 1433
- setPreviousSibling
 - IDOMNode, 696
- setProducer
 - JawsMako::IPDFOutput, 1337
- setProgressMonitor
 - JawsMako::ITransform, 1501
- setProlog
 - JawsMako::IDistiller, 302
 - JawsMako::IPSOInput, 1389
- setProperty
 - IDOMImageProperties, 611
 - IDOMMetadata, 683

- IDOMNode, 696
- setPublicKeyEncryption
 - JawsMako::IPDFOutput, 1337
- setQName
 - IDOMJobTkNode, 650
- setQuadding
 - JawsMako::IFormField, 1022
 - JawsMako::IWidgetAnnotation, 1527
- setQuadPoints
 - JawsMako::ILinkAnnotation, 1160
 - JawsMako::IRedactionAnnotation, 1419
 - JawsMako::ITextMarkupAnnotation, 1491
- setRadiusX
 - IDOMArcSegment, 332
 - IDOMRadialGradientBrush, 820
- setRadiusY
 - IDOMArcSegment, 333
 - IDOMRadialGradientBrush, 820
- setRasterFallbackResolution
 - JawsMako::IRendererTransform, 1433
- setRasterFallbackThreshold
 - JawsMako::IRendererTransform, 1433
 - JawsMako::ISeparator, 1441
- setRasterImageReuseCacheSize
 - JawsMako::IRendererTransform, 1434
- setRect
 - JawsMako::IAnnotation, 231
- setReencodelImages
 - JawsMako::IPDFOutput, 1338
- setRenameFonts
 - JawsMako::IXPSOutput, 1552
- setRenderingIntent
 - IDOMGlyphs, 547
 - IDOMPathNode, 765
 - IDOMTransparencyGroup, 938
- setRenderResolution
 - JawsMako::IPDFOutput, 1338
 - JawsMako::IXAMLGenerator, 1542
 - JawsMako::IXPSOutput, 1553
- setRenderTransform
 - IDOMFormInstance, 505
 - IDOMGlyphs, 548
 - IDOMGroup, 565
 - IDOMMatrix, 678
 - IDOMPathGeometry, 737
 - IDOMPathNode, 765
 - IDOMTransformableBrush, 927
- setRenderTransparentNodesOnPageGroupMismatch
 - JawsMako::IRendererTransform, 1434
- setResolution
 - JawsMako::IDistiller, 303
 - JawsMako::IPCL5Output, 1257
- setResourceDevicePath
 - JawsMako::IDistiller, 303
- setResourceDictionary
 - IDOMCanvas, 352
 - IDOMFixedPage, 455
- setRetainClippedContent
 - JawsMako::IPDFOutput, 1338
- setRetainEmbeddedFiles
 - JawsMako::IPDFOutput, 1339
- setRight
 - IDOMPageRectTarget, 717
- setRolloverCaption
 - JawsMako::IWidgetAppearanceCharacteristics, 1533
- setRopResolution
 - JawsMako::IPCL5Input, 1252
 - JawsMako::IPCLXLInput, 1264
- setRotation
 - JawsMako::IWidgetAppearanceCharacteristics, 1533
- setRotationAngle
 - IDOMArcSegment, 333
- setSelectedOptions
 - JawsMako::IWidgetAnnotation, 1527
- setSequentialMode
 - JawsMako::IInput, 1084
- setShadingType
 - IDOMShadingPatternType4567Brush, 883
- setShouldZeroWidthLinesBeVisible
 - IDOMPathNode, 766
- setSimulateBlackDeviceGrayTextOverprint
 - JawsMako::IOverprintSimulationTransform, 1214
- setSimulationColorSpace
 - JawsMako::IOverprintSimulationTransform, 1214
- setSnapsToDevicePixels
 - IDOMPathNode, 766
- setSpacing
 - JawsMako::ILayoutParagraph, 1143
- setSpotHalftone
 - JawsMako::IRendererTransform, 1434
- setSpreadMethod
 - IDOMGradientBrush, 554
- setStartCircleCenter
 - IDOMShadingPatternType3Brush, 872
- setStartCircleRadius
 - IDOMShadingPatternType3Brush, 872
- setStartPoint
 - IDOMLinearGradientBrush, 669
 - IDOMPathFigure, 727
 - IDOMShadingPatternType2Brush, 864
- setStartupDirectory
 - ISession, 1445
- setState
 - JawsMako::IAnnotation, 232
- setStream
 - IDOMCachedImage, 342
 - IDOMCompositedImage, 430
 - IDOMFilteredImage, 445
 - IDOMRecombineAlphaImage, 830
 - IDOMRecombineImage, 835
 - IDOMResource, 837
 - IDOMTrivialImage, 942
- setStreamingOutput
 - JawsMako::IPSOOutput, 1396

- setStroke
 - IDOMPathNode, 766
- setStrokeAdjust
 - IDOMPathNode, 766
- setStrokeDashLineCap
 - IDOMPathNode, 767
- setStrokeDashOffset
 - IDOMPathNode, 767
- setStrokeDashPattern
 - IDOMPathNode, 767
- setStrokeEndLineCap
 - IDOMPathNode, 768
- setStrokeLineJoin
 - IDOMPathNode, 768
- setStrokeMiterLimit
 - IDOMPathNode, 768
- setStrokeMiterLimitTreatment
 - IDOMPathNode, 769
- setStrokeOverprints
 - IDOMPathNode, 769
- setStrokePath
 - IDOMCharPathGroup, 365
- setStrokeStartLineCap
 - IDOMPathNode, 769
- setStrokeThickness
 - IDOMPathNode, 769
- setStructureElement
 - IDOMOutlineEntry, 711
- setStructureElementReference
 - IDOMForm, 498
- setStyle
 - JawsMako::IPageLabel, 1235
- setStyleSimulations
 - IDOMGlyphs, 548
- setSubsetFonts
 - JawsMako::IDistiller, 303
 - JawsMako::IPDFOutput, 1339
 - JawsMako::IXAMLGenerator, 1542
 - JawsMako::IXPSOutput, 1553
- setSweepDirection
 - IDOMArcSegment, 333
- setSymbolSetIDCode
 - IDOMFontPCL5, 476
- setTarget
 - IDOMOutlineEntry, 711
- setTargetColorSpace
 - JawsMako::IPDFOutput, 1339
 - JawsMako::IPSOOutput, 1397
 - JawsMako::IXAMLGenerator, 1543
 - JawsMako::IXPSOutput, 1553
- setTargetId
 - IDOMInternalTarget, 616
- setTargetPage
 - IDOMPageTarget, 721
- setTargetProfile
 - JawsMako::IColorConverterTransform, 254
 - JawsMako::IPDFOutput, 1340
 - JawsMako::IPSOOutput, 1397
 - JawsMako::IXAMLGenerator, 1543
 - JawsMako::IXPSOutput, 1553
- setTargetResolutionCallback
 - JawsMako::ISVGGenerator, 1482
- setTargetSpace
 - JawsMako::IColorConverterTransform, 254
- setTargetUri
 - IDOMExternalTarget, 440
- setTemporaryDirectory
 - ISession, 1446
- setTextColor
 - IDOMOutlineEntry, 711
- setTextStyle
 - IDOMOutlineEntry, 712
- setThresholdHalftone
 - JawsMako::IRendererTransform, 1435
- setThumbnail
 - IDOMFixedPage, 455
 - JawsMako::IDocumentAssembly, 318
- setThumbnails
 - JawsMako::IDistiller, 304
- setTickMax
 - JawsMako::IProgressTick, 1382
- setTIFFCompression
 - JawsMako::IImageEncoderTransform, 1048
- setTileMode
 - IDOMImageBrush, 592
 - IDOMVisualBrush, 955
- setTilingStep
 - IDOMTilingPatternBrush, 924
- setTilingType
 - IDOMTilingPatternBrush, 924
- setTime
 - IEDLTime, 986
- setTop
 - IDOMPageRectTarget, 718
- setTransfers
 - JawsMako::IDistiller, 304
- setTrimBox
 - IDOMFixedPage, 456
- setTrimToBBox
 - JawsMako::IDistiller, 304
- setUnicodeString
 - IDOMGlyphs, 548
- setUnixParameters
 - IDOMActionLaunch, 325
- setupForPDFStyleOutput
 - JawsMako::IStrokerTransform, 1465
- setupForXPSStyleOutput
 - JawsMako::IStrokerTransform, 1466
- setUri
 - IDOMResource, 837
- setUseDeviceDependentColor
 - JawsMako::IDistiller, 304
- setUseImageResolutionForRenderingWherePossible
 - JawsMako::IRendererTransform, 1435
- setUseMaskResolutionForMaskedImages
 - JawsMako::IImageDownsamplerTransform, 1041

- JawsMako::IPDFOutput, 1340
- setUsers
 - JawsMako::IOptionalContentGroupUsage, 1191
- setValidateXmp
 - JawsMako::IPDFOutput, 1341
- setValue
 - IDOMJobTkValue, 658
 - JawsMako::IFormField, 1023
 - JawsMako::IWidgetAnnotation, 1528
- setVectorAndTextResolution
 - JawsMako::IRendererTransform, 1435
- setVectorAreaFormReuseCacheSize
 - JawsMako::IRendererTransform, 1436
- setVectorAreaSourceReuseCacheSize
 - JawsMako::IRendererTransform, 1436
- setVectorImageReuseCacheSize
 - JawsMako::IRendererTransform, 1436
- setVersion
 - IDOMJobTkContent, 633
 - JawsMako::IPCL5Output, 1257
 - JawsMako::IPDFOutput, 1341
- setVerticesPerRow
 - IDOMShadingPatternType4567Brush, 883
- setViewBox
 - IDOMImageBrush, 592
 - IDOMVisualBrush, 956
- setViewBoxUnits
 - IDOMImageBrush, 593
 - IDOMVisualBrush, 956
- setViewPort
 - IDOMImageBrush, 593
 - IDOMVisualBrush, 956
- setViewPortUnits
 - IDOMImageBrush, 593
 - IDOMVisualBrush, 957
- setVirtualSpaceThreshold
 - JawsMako::IPageLayout, 1238
- setVisibleZeroWidthLines
 - JawsMako::ISVGGenerator, 1483
- setVisual
 - IDOMTilingPatternBrush, 925
 - IDOMVisualBrush, 957
- setWidth
 - IDOMFixedPage, 456
 - IImageFrameWriter, 1057
- setWinParameters
 - IDOMActionLaunch, 325
- setXfaPacketData
 - JawsMako::IForm, 1010
- setXResolution
 - IImageFrameWriter, 1058
- setXX
 - CTransformMatrix< TItem >, 145
- setXY
 - CTransformMatrix< TItem >, 145
- setYear
 - IEDLTime, 987
- setYResolution
 - IImageFrameWriter, 1058
- setYX
 - CTransformMatrix< TItem >, 145
- setYY
 - CTransformMatrix< TItem >, 145
- setZoom
 - IDOMPageRectTarget, 718
- shouldMerge
 - JawsMako::IFontProcessorDeferredTransform, 997
 - JawsMako::IFontProcessorTransform, 1000
- shouldSubset
 - JawsMako::IFontProcessorDeferredTransform, 997
 - JawsMako::IFontProcessorTransform, 1000
- SignatureID, 1556
- similar
 - CGlyphsCluster, 119
 - CIndicesGlyph, 121
 - IDOMColorSpace, 385
- sizeRequest
 - JawsMako::IMediaHandler, 1171
- skip
 - IInputStream, 1104
- skipScanLines
 - IImageFrame, 1054
- smartptr.h, 1558
- split
 - IDOMGlyphs, 548
 - IDOMPathNode, 770
- stateInsideBrush
 - JawsMako::CTransformState, 149
- stateInsideNode
 - JawsMako::CTransformState, 149
- store
 - JawsMako::IPDFObjectStore, 1312
- storeUsingNewReference
 - JawsMako::IPDFObjectStore, 1313
- Streams, 81
- stripInstructions
 - IDOMFontOpenType, 471
- structure.h, 1558
- supplement
 - IDOMFontOpenType::CCIDMap, 109
- tagNode
 - JawsMako::IStructure, 1468
- testGamut
 - IColorManager, 273
 - IDOMColor, 380
- Text, 104
- text.h, 1558
- toPDFDate
 - IEDLTime, 987
- toW3CDTF
 - IEDLTime, 987
- transform
 - CTransformMatrix< TItem >, 146
 - JawsMako::ITransform, 1501, 1502
 - JawsMako::ITransformChain, 1507, 1508
- transformAnnotation

- JawsMako::ICustomTransform::Implementation, 1063
- transformAnnotationAppearance
 - JawsMako::ICustomTransform::Implementation, 1063
- transformBrush
 - JawsMako::ICustomTransform::Implementation, 1064
- transformCanvas
 - JawsMako::ICustomTransform::Implementation, 1064
- transformCharPathGroup
 - JawsMako::ICustomTransform::Implementation, 1065
- transformColor
 - JawsMako::ICustomTransform::Implementation, 1065
- transformColorSpace
 - JawsMako::ICustomTransform::Implementation, 1066
- transformFixedPage
 - JawsMako::ICustomTransform::Implementation, 1066
- transformFont
 - JawsMako::ICustomTransform::Implementation, 1067
- transformForm
 - JawsMako::ICustomTransform::Implementation, 1067
- transformFormInstance
 - JawsMako::ICustomTransform::Implementation, 1068
- transformGlyphs
 - JawsMako::ICustomTransform::Implementation, 1068
- transformGradientBrush
 - JawsMako::ICustomTransform::Implementation, 1069
- transformGroup
 - JawsMako::ICustomTransform::Implementation, 1069
- transformImage
 - JawsMako::ICustomTransform::Implementation, 1070
- transformImageBrush
 - JawsMako::ICustomTransform::Implementation, 1070
- transformLinearGradientBrush
 - JawsMako::ICustomTransform::Implementation, 1071
- transformMaskedBrush
 - JawsMako::ICustomTransform::Implementation, 1071
- transformNode
 - JawsMako::ICustomTransform::Implementation, 1071
- transformNullBrush
 - JawsMako::ICustomTransform::Implementation, 1072
- transformPage
 - JawsMako::ITransform, 1504
 - JawsMako::ITransformChain, 1508
- transformPath
 - JawsMako::ICustomTransform::Implementation, 1072
- transformPriv
 - JawsMako::CTransformState, 150
- transformRadialGradientBrush
 - JawsMako::ICustomTransform::Implementation, 1073
- transformRect
 - CTransformMatrix< TItem >, 146
- Transforms, 92
 - create, 95, 96
 - eBrushUsage, 94
 - eBUAlternateImage, 95
 - eBUFill, 95
 - eBUGeneral, 95
 - eBUNone, 95
 - eBUOpacityMask, 95
 - eBUStroke, 95
- transforms.h, 1558
- transformShadingPatternBrush
 - JawsMako::ICustomTransform::Implementation, 1073
- transformSoftMaskBrush
 - JawsMako::ICustomTransform::Implementation, 1074
- transformSolidColorBrush
 - JawsMako::ICustomTransform::Implementation, 1074
- transformTilingPatternBrush
 - JawsMako::ICustomTransform::Implementation, 1075
- transformTransparencyGroup
 - JawsMako::ICustomTransform::Implementation, 1075
- transformVisualBrush
 - JawsMako::ICustomTransform::Implementation, 1076
- transformVisualRoot
 - JawsMako::ICustomTransform::Implementation, 1076
- translate
 - CTransformMatrix< TItem >, 147
- Transparency blend modes, 98
 - eBlendMode, 99
 - eBlendModeColor, 99
 - eBlendModeColorBurn, 99
 - eBlendModeColorDodge, 99
 - eBlendModeDarken, 99
 - eBlendModeDifference, 99
 - eBlendModeExclusion, 99
 - eBlendModeHardLight, 99
 - eBlendModeHue, 99
 - eBlendModeLighten, 99

- eBlendModeLuminosity, [99](#)
- eBlendModeMultiply, [99](#)
- eBlendModeNormal, [99](#)
- eBlendModeOverlay, [99](#)
- eBlendModeSaturation, [99](#)
- eBlendModeScreen, [99](#)
- eBlendModeSoftLight, [99](#)
- eBlendModeUnspecified, [99](#)
- Types, [105](#)
- types.h, [1558](#)
- undefine
 - JawsMako::IPDFDictionary, [1285](#), [1286](#)
- unite
 - IDOMShape, [890](#)
- unregisterNotify
 - IDOMNode, [696](#)
- unregisterObject
 - IDOMCatalog, [358](#)
- updateHash
 - JawsMako::IHashable, [1031](#)
- useDeviceLinkForConversionsBetween
 - IColorManager, [273](#), [274](#)
- Utility, [87](#)
- Validate, [104](#)
- validate
 - JawsMako::IPDFValidator, [1361](#)
- validateInstructions
 - IDOMFontOpenType, [472](#)
 - IDOMGlyphs, [549](#)
- value
 - JawsMako::IPJLParse::CPjlAttributeValue, [130](#)
- Version 7.4.0 API Documentation, [1](#)
- walkTree
 - IDOMNode, [696](#)
- widgetInSubtree
 - JawsMako::IFormField, [1023](#), [1024](#)
- widgetInTree
 - JawsMako::IForm, [1011](#)
- width
 - JawsMako::CAnnotationBorder, [107](#)
- write
 - IOutputStream, [1209](#)
- writeAssembly
 - JawsMako::IOutput, [1199](#), [1200](#)
- writeEnumerationItemBlock
 - IFontPCL5WriteSegmentBlockEnumerator, [993](#)
- writeFormatted
 - IOutputStream, [1209](#)
- writeFormattedE
 - IOutputStream, [1209](#)
- writePage
 - JawsMako::IOutputWriter, [1211](#)
- writeScanLine
 - IImageFrameWriter, [1058](#)
- writeSegmentBlock
 - IFontHeaderWriteSegmentBlockEnumerator, [990](#)
- writeStream
 - IOutputStream, [1210](#)
- xpsoutput.h, [1558](#)
- xx
 - CTransformMatrix< TItem >, [147](#)
- xy
 - CTransformMatrix< TItem >, [147](#)
- yx
 - CTransformMatrix< TItem >, [147](#)
- yy
 - CTransformMatrix< TItem >, [147](#)



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