<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.9</td>
<td>IPJLParser::CPjLAttributeValue Class Reference</td>
<td>53</td>
</tr>
<tr>
<td>8.9.1</td>
<td>Detailed Description</td>
<td>53</td>
</tr>
<tr>
<td>8.9.2</td>
<td>Member Data Documentation</td>
<td>53</td>
</tr>
<tr>
<td>8.9.2.1</td>
<td>key</td>
<td>54</td>
</tr>
<tr>
<td>8.9.2.2</td>
<td>modifier</td>
<td>54</td>
</tr>
<tr>
<td>8.9.2.3</td>
<td>value</td>
<td>54</td>
</tr>
<tr>
<td>8.10</td>
<td>CQuadPoint Class Reference</td>
<td>54</td>
</tr>
<tr>
<td>8.10.1</td>
<td>Detailed Description</td>
<td>54</td>
</tr>
<tr>
<td>8.11</td>
<td>CRectlnset Class Reference</td>
<td>54</td>
</tr>
<tr>
<td>8.11.1</td>
<td>Detailed Description</td>
<td>55</td>
</tr>
<tr>
<td>8.12</td>
<td>IJawsRenderer::CSpotHalftone Class Reference</td>
<td>55</td>
</tr>
<tr>
<td>8.12.1</td>
<td>Detailed Description</td>
<td>55</td>
</tr>
<tr>
<td>8.13</td>
<td>CTemporaryStoreParameters Class Reference</td>
<td>56</td>
</tr>
<tr>
<td>8.13.1</td>
<td>Detailed Description</td>
<td>56</td>
</tr>
<tr>
<td>8.13.2</td>
<td>Constructor &amp; Destructor Documentation</td>
<td>56</td>
</tr>
<tr>
<td>8.13.2.1</td>
<td>CTemporaryStoreParameters()</td>
<td>56</td>
</tr>
<tr>
<td>8.14</td>
<td>IJawsRenderer::CThresholdArrayHalftone Class Reference</td>
<td>57</td>
</tr>
<tr>
<td>8.14.1</td>
<td>Detailed Description</td>
<td>57</td>
</tr>
<tr>
<td>8.15</td>
<td>IJawsRenderer::CThresholdHalftone Class Reference</td>
<td>58</td>
</tr>
<tr>
<td>8.15.1</td>
<td>Detailed Description</td>
<td>58</td>
</tr>
<tr>
<td>8.16</td>
<td>CTransformMatrix&lt;TItem&gt; Class Template Reference</td>
<td>58</td>
</tr>
<tr>
<td>8.16.1</td>
<td>Detailed Description</td>
<td>60</td>
</tr>
<tr>
<td>8.16.2</td>
<td>Member Enumeration Documentation</td>
<td>60</td>
</tr>
<tr>
<td>8.16.2.1</td>
<td>eOperationTypes</td>
<td>60</td>
</tr>
<tr>
<td>8.16.3</td>
<td>Constructor &amp; Destructor Documentation</td>
<td>60</td>
</tr>
<tr>
<td>8.16.3.1</td>
<td>CTransformMatrix() [1/3]</td>
<td>60</td>
</tr>
<tr>
<td>8.16.3.2</td>
<td>CTransformMatrix() [2/3]</td>
<td>61</td>
</tr>
<tr>
<td>8.16.3.3</td>
<td>CTransformMatrix() [3/3]</td>
<td>61</td>
</tr>
<tr>
<td>8.16.4</td>
<td>Member Function Documentation</td>
<td>62</td>
</tr>
<tr>
<td>8.16.4.1</td>
<td>classify()</td>
<td>62</td>
</tr>
<tr>
<td>Section</td>
<td>Class Reference</td>
<td>Detailed Description</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>8.17.3.1</td>
<td>inUncoloredTilingBrush</td>
<td>71</td>
</tr>
<tr>
<td>8.17.3.2</td>
<td>transformPriv</td>
<td>72</td>
</tr>
<tr>
<td>8.18</td>
<td>JawsMako::IAnnotationUtils::CXMLResource</td>
<td>72</td>
</tr>
<tr>
<td>8.18.1</td>
<td>Detailed Description</td>
<td>72</td>
</tr>
<tr>
<td>8.19</td>
<td>IDOMTransparencyGroup::Data</td>
<td>72</td>
</tr>
<tr>
<td>8.19.1</td>
<td>Detailed Description</td>
<td>73</td>
</tr>
<tr>
<td>8.20</td>
<td>IDOMImage::Data</td>
<td>73</td>
</tr>
<tr>
<td>8.20.1</td>
<td>Detailed Description</td>
<td>73</td>
</tr>
<tr>
<td>8.21</td>
<td>IDOMRawImage::Data</td>
<td>74</td>
</tr>
<tr>
<td>8.21.1</td>
<td>Detailed Description</td>
<td>74</td>
</tr>
<tr>
<td>8.22</td>
<td>IDOMPDFImage::Data</td>
<td>74</td>
</tr>
<tr>
<td>8.22.1</td>
<td>Detailed Description</td>
<td>75</td>
</tr>
<tr>
<td>8.23</td>
<td>IDOMPCImage::Data</td>
<td>75</td>
</tr>
<tr>
<td>8.23.1</td>
<td>Detailed Description</td>
<td>76</td>
</tr>
<tr>
<td>8.24</td>
<td>IDOMRecombineImage::Data</td>
<td>76</td>
</tr>
<tr>
<td>8.24.1</td>
<td>Detailed Description</td>
<td>76</td>
</tr>
<tr>
<td>8.25</td>
<td>IDOMRecombineAlphaImage::Data</td>
<td>77</td>
</tr>
<tr>
<td>8.25.1</td>
<td>Detailed Description</td>
<td>77</td>
</tr>
<tr>
<td>8.26</td>
<td>IEDLTime::Data</td>
<td>77</td>
</tr>
<tr>
<td>8.26.1</td>
<td>Detailed Description</td>
<td>78</td>
</tr>
<tr>
<td>8.27</td>
<td>IDOMCompositeImage::Data</td>
<td>78</td>
</tr>
<tr>
<td>8.27.1</td>
<td>Detailed Description</td>
<td>78</td>
</tr>
<tr>
<td>8.28</td>
<td>IDOMImageChannelSelectorFilter::Data</td>
<td>79</td>
</tr>
<tr>
<td>8.28.1</td>
<td>Detailed Description</td>
<td>79</td>
</tr>
<tr>
<td>8.29</td>
<td>IDOMImageColorSpaceSubstitutionFilter::Data</td>
<td>79</td>
</tr>
<tr>
<td>8.29.1</td>
<td>Detailed Description</td>
<td>80</td>
</tr>
<tr>
<td>8.30</td>
<td>IDOMGradientStop::Data</td>
<td>80</td>
</tr>
<tr>
<td>8.30.1</td>
<td>Detailed Description</td>
<td>80</td>
</tr>
<tr>
<td>8.31</td>
<td>IDOMImageColorConverterFilter::Data</td>
<td>81</td>
</tr>
<tr>
<td>8.31.1</td>
<td>Detailed Description</td>
<td>81</td>
</tr>
</tbody>
</table>
8.32 IDOMImageBleederFilter::Data Class Reference .............................................. 81
  8.32.1 Detailed Description .............................................................................. 82

8.33 IDOMImageDownsamplerFilter::Data Class Reference ......................................... 82
  8.33.1 Detailed Description .............................................................................. 82

8.34 IDOMImageMaskExpanderFilter::Data Class Reference ........................................ 83
  8.34.1 Detailed Description .............................................................................. 83

8.35 IDOMSolidColorBrush::Data Class Reference ................................................... 83
  8.35.1 Detailed Description .............................................................................. 84

8.36 IDOMImageDeindexerFilter::Data Class Reference ................................................ 84
  8.36.1 Detailed Description .............................................................................. 84

8.37 IDOMImageDeviceNToBaseFilter::Data Class Reference ......................................... 85
  8.37.1 Detailed Description .............................................................................. 85

8.38 IDOMLinearGradientBrush::Data Class Reference .............................................. 85
  8.38.1 Detailed Description .............................................................................. 86

8.39 IDOMImageInverterFilter::Data Class Reference ................................................ 86
  8.39.1 Detailed Description .............................................................................. 86

8.40 IDOMDePremultiplierFilter::Data Class Reference .............................................. 87
  8.40.1 Detailed Description .............................................................................. 87

8.41 IDOMRadialGradientBrush::Data Class Reference .............................................. 87
  8.41.1 Detailed Description .............................................................................. 88

8.42 IDOMImageMatteRemoverFilter::Data Class Reference .......................................... 88
  8.42.1 Detailed Description .............................................................................. 88

8.43 IDOMImageBitScalerFilter::Data Class Reference ................................................ 89
  8.43.1 Detailed Description .............................................................................. 89

8.44 IDOMImageBrush::Data Class Reference ........................................................... 89
  8.44.1 Detailed Description .............................................................................. 90

8.45 IDOMImageColorKeyFilter::Data Class Reference ............................................... 90
  8.45.1 Detailed Description .............................................................................. 90

8.46 IDOMImageDecodeFilter::Data Class Reference .................................................. 91
  8.46.1 Detailed Description .............................................................................. 91
8.113.3.1 addAppearance() .................................................. 145
8.113.3.2 clone() .............................................................. 146
8.113.3.3 getAppearance() .................................................. 146
8.113.3.4 getAppearances() ................................................. 146
8.113.3.5 getBorder() ....................................................... 147
8.113.3.6 getColor() .......................................................... 147
8.113.3.7 getContents() .................................................... 147
8.113.3.8 getFlags() .......................................................... 147
8.113.3.9 getModificationTime() ......................................... 148
8.113.3.10 getRect() .......................................................... 148
8.113.3.11 getReference() .................................................. 148
8.113.3.12 getState() .......................................................... 148
8.113.3.13 getType() .......................................................... 149
8.113.3.14 matchesReference() ............................................ 149
8.113.3.15 rotate() ........................................................... 149
8.113.3.16 setBorder() ....................................................... 150
8.113.3.17 setColor() .......................................................... 150
8.113.3.18 setContents() .................................................... 150
8.113.3.19 setFlags() .......................................................... 150
8.113.3.20 setModificationTime() ....................................... 151
8.113.3.21 setRect() .......................................................... 151
8.113.3.22 setState() .......................................................... 151

8.114 JawsMako::IAnnotationAppearance Class Reference .................. 152

8.114.1 Detailed Description ............................................... 152

8.114.2 Member Function Documentation ................................... 153

8.114.2.1 clone() ........................................................... 153
8.114.2.2 create() ........................................................... 153
8.114.2.3 getForm() .......................................................... 153
8.114.2.4 getScaledAppearance() ....................................... 154
8.114.2.5 getState() .......................................................... 154
8.114.2.6 getUsage() .................................................. 154

8.115 JawsMako::IAnnotationReference Class Reference ........................................... 155
  8.115.1 Detailed Description .................................................. 155
  8.115.2 Member Function Documentation ........................................ 155
    8.115.2.1 equals() .................................................. 155

8.116 JawsMako::IAnnotationUtils Class Reference .................................................. 156
  8.116.1 Detailed Description .................................................. 156
  8.116.2 Member Function Documentation ........................................ 156
    8.116.2.1 createAnnotationReferenceFromTag() ......................... 156
    8.116.2.2 generateXMLForDocument() ................................... 157

8.117 JawsMako::ICaretAnnotation Class Reference .................................................. 157
  8.117.1 Detailed Description .................................................. 158
  8.117.2 Member Function Documentation ........................................ 158
    8.117.2.1 getRectInset() .................................................. 158

8.118 JawsMako::ICFFCIDSplitterTransform Class Reference .................................... 159
  8.118.1 Detailed Description .................................................. 159
  8.118.2 Member Function Documentation ........................................ 159
    8.118.2.1 create() .................................................. 160

8.119 JawsMako::IColorConverterTransform Class Reference ................................... 161
  8.119.1 Detailed Description .................................................. 162
  8.119.2 Member Function Documentation ........................................ 163
    8.119.2.1 create() .................................................. 163
    8.119.2.2 setConvertColorsInsideLuminositySoftMasks() .................. 163
    8.119.2.3 setTargetProfile() ........................................ 163
    8.119.2.4 setTargetSpace() ........................................ 164

8.120 IColorManager Class Reference .................................................. 164
  8.120.1 Detailed Description .................................................. 166
  8.120.2 Member Function Documentation ........................................ 166
    8.120.2.1 convertColors() [1/2] ........................................ 167
    8.120.2.2 convertColors() [2/2] ........................................ 167
8.122.1 Detailed Description ................................................................. 178
8.123 JawsMako::IDocument Class Reference .............................................. 179
  8.123.1 Detailed Description ............................................................... 180
  8.123.2 Member Function Documentation ............................................... 181
    8.123.2.1 addNamedDestination() ..................................................... 181
    8.123.2.2 appendPage() ............................................................... 181
    8.123.2.3 clone() ........................................................................ 181
    8.123.2.4 create() ....................................................................... 182
    8.123.2.5 getForm() ...................................................................... 182
    8.123.2.6 getJobTicket() ................................................................. 182
    8.123.2.7 getNumPages() ................................................................. 182
    8.123.2.8 getOutline() .................................................................. 183
    8.123.2.9 getPage() ...................................................................... 183
    8.123.2.10 getThreads() ................................................................ 183
    8.123.2.11 insertPage() .................................................................. 183
    8.123.2.12 removePage() .................................................................. 184
    8.123.2.13 setNamedDestinations() .................................................... 184
    8.123.2.14 setOutline() .................................................................. 184
8.124 JawsMako::IDocumentAssembly Class Reference ................................. 185
  8.124.1 Detailed Description ................................................................. 186
  8.124.2 Member Function Documentation ............................................... 186
    8.124.2.1 appendDocument() ............................................................ 186
    8.124.2.2 clone() ........................................................................ 187
    8.124.2.3 create() ....................................................................... 187
    8.124.2.4 getDocument() ................................................................. 187
    8.124.2.5 getJobMetadata() ............................................................. 188
    8.124.2.6 getJobTicket() ................................................................. 188
    8.124.2.7 getNumDocuments() .......................................................... 188
    8.124.2.8 getSecurityInfo() ............................................................. 188
    8.124.2.9 getThumbnail() ............................................................... 189
| 8.127.3.1 | classID() | 199 |
| 8.127.3.2 | convertToSimpleSegment() | 199 |
| 8.127.3.3 | getIsLargeArc() | 199 |
| 8.127.3.4 | getPoint() | 200 |
| 8.127.3.5 | getRadiusX() | 200 |
| 8.127.3.6 | getRadiusY() | 200 |
| 8.127.3.7 | getRotationAngle() | 201 |
| 8.127.3.8 | getSweepDirection() | 201 |
| 8.127.3.9 | setIsLargeArc() | 201 |
| 8.127.3.10 | setPoint() | 202 |
| 8.127.3.11 | setRadiusX() | 202 |
| 8.127.3.12 | setRadiusY() | 202 |
| 8.127.3.13 | setRotationAngle() | 203 |
| 8.127.3.14 | setSweepDirection() | 203 |

**8.128 IDOMAudioFile Class Reference**

| 8.128.1 | Detailed Description | 204 |
| 8.128.2 | Member Function Documentation | 204 |
| 8.128.2.1 | classID() | 205 |

**8.129 IDOMBrush Class Reference**

| 8.129.1 | Detailed Description | 206 |
| 8.129.2 | Member Function Documentation | 206 |
| 8.129.2.1 | getBrushType() | 206 |
| 8.129.2.2 | getOpacity() | 206 |
| 8.129.2.3 | setOpacity() | 206 |

**8.130 IDOMCanvas Class Reference**

| 8.130.1 | Detailed Description | 208 |
| 8.130.2 | Member Function Documentation | 209 |
| 8.130.2.1 | classID() | 209 |
| 8.130.2.2 | getAutomationPropertiesHelpText() | 209 |
| 8.130.2.3 | getAutomationPropertiesName() | 209 |
8.130.2.4 getEdgeMode() ........................................... 210
8.130.2.5 getLanguage() ........................................... 210
8.130.2.6 getNavigateLink() ...................................... 211
8.130.2.7 getResourceDictionary() ................................. 211
8.130.2.8 setAutomationPropertiesHelpText() .......................... 211
8.130.2.9 setAutomationPropertiesName() ............................. 212
8.130.2.10 setEdgeMode() ........................................... 212
8.130.2.11 setLanguage() ........................................... 213
8.130.2.12 setNavigateLink() ...................................... 213
8.130.2.13 setResourceDictionary() ................................. 213

8.131 IDOMCatalog Class Reference .................................................. 214
8.131.1 Detailed Description ................................................. 215
8.131.2 Member Function Documentation ................................. 215
 8.131.2.1 classID() ........................................... 215
 8.131.2.2 createNewDOMid() {1/2} ................................... 215
 8.131.2.3 createNewDOMid() {2/2} ................................... 216
 8.131.2.4 getCount() ........................................... 216
 8.131.2.5 getIdByIndex() ........................................... 216
 8.131.2.6 getIdByNumbers() ....................................... 217
 8.131.2.7 getIdByURI() .......................................... 217
 8.131.2.8 getObject() .......................................... 218
 8.131.2.9 getURI() ........................................... 218
 8.131.2.10 registerNumbers() .................................... 218
 8.131.2.11 registerObject() ....................................... 220
 8.131.2.12 unregisterObject() .................................. 220

8.132 IDOMCharPathGroup Class Reference ..................................... 221
8.132.1 Detailed Description ................................................. 222
8.132.2 Member Function Documentation ................................. 222
 8.132.2.1 classID() ........................................... 222
 8.132.2.2 getBlendMode() ......................................... 223
8.132.2.3 getCharPathType() ........................................ 223
8.132.2.4 getClippedGroup() ........................................ 223
8.132.2.5 getFill() ........................................ 223
8.132.2.6 getStrokePath() ........................................ 224
8.132.2.7 setBlendMode() ........................................ 224
8.132.2.8 setCharPathType() ........................................ 224
8.132.2.9 setClippedGroup() ........................................ 226
8.132.2.10 setFill() ........................................ 226
8.132.2.11 setStrokePath() ........................................ 226

8.133 IDOMColor Class Reference .................................... 227

8.133.1 Detailed Description ........................................ 228
8.133.2 Member Function Documentation ............................ 228
8.133.2.1 classID() ........................................ 228
8.133.2.2 create() ........................................ 229
8.133.2.3 createFromArray() ........................................ 229
8.133.2.4 createFromVect() ........................................ 230
8.133.2.5 getAlpha() ........................................ 230
8.133.2.6 getColorSpace() ........................................ 230
8.133.2.7 getComponentValue() .................................... 231
8.133.2.8 setAlpha() ........................................ 231
8.133.2.9 setColorSpace() ........................................ 232
8.133.2.10 setColorSpace() [1/2] ................................ 232
8.133.2.11 setComponentValue() .................................. 233

8.134 IDOMColorSpace Class Reference .............................. 233

8.134.1 Detailed Description ........................................ 235
8.134.2 Member Enumeration Documentation ........................ 235
8.134.2.1 eColorSpaceType ........................................ 235
8.134.3 Member Function Documentation ............................ 236
8.134.3.1 equals() ........................................ 236
8.134.3.2 getColorSpaceType() .................................... 236
8.134.3.3 getComponentRange() .................................................. 236
8.134.3.4 getComponentsHaveSameRange() ................................. 237
8.134.3.5 getDefaultRenderingIntent() ........................................ 237
8.134.3.6 getNumComponents() ................................................... 237
8.134.3.7 similar() ................................................................. 237

8.135 IDOMColorSpaceDeviceCMY Class Reference .......................... 238
  8.135.1 Detailed Description .................................................. 239
  8.135.2 Member Function Documentation ................................... 239
    8.135.2.1 classID() ......................................................... 239
    8.135.2.2 create() ........................................................ 239

8.136 IDOMColorSpaceDeviceCMYK Class Reference .......................... 240
  8.136.1 Detailed Description .................................................. 240
  8.136.2 Member Function Documentation ................................... 240
    8.136.2.1 classID() ......................................................... 241
    8.136.2.2 create() ........................................................ 241

8.137 IDOMColorSpaceDeviceGray Class Reference .......................... 241
  8.137.1 Detailed Description .................................................. 242
  8.137.2 Member Function Documentation ................................... 242
    8.137.2.1 classID() ......................................................... 242
    8.137.2.2 create() ........................................................ 243

8.138 IDOMColorSpaceDeviceN Class Reference .............................. 243
  8.138.1 Detailed Description .................................................. 244
  8.138.2 Member Function Documentation ................................... 245
    8.138.2.1 classID() ......................................................... 245
    8.138.2.2 create() ........................................................ 245
    8.138.2.3 create() ......................................................... 245
    8.138.2.4 create() ......................................................... 246
    8.138.2.5 getAlternateColorSpace() .................................... 247
    8.138.2.6 getColorant() .................................................... 247
    8.138.2.7 getIsNChannel() ................................................ 247
8.138.2.8 getPrintingOrder() .......................................................... 247
8.138.2.9 getProcessColorSpace() ................................................... 248
8.138.2.10 getProcessComponentNames() ........................................ 248
8.138.2.11 getTintTransform() ........................................................... 248

8.139 IDOMColorSpaceDeviceRGB Class Reference ........................................... 249
8.139.1 Detailed Description ................................................................... 249
8.139.2 Member Function Documentation .................................................. 249
8.139.2.1 classID() ................................................................................. 250
8.139.2.2 create() ................................................................................. 250

8.140 IDOMColorSpaceICCBased Class Reference ............................................. 250
8.140.1 Detailed Description ................................................................... 252
8.140.2 Member Function Documentation .................................................. 252
8.140.2.1 classID() ................................................................................. 252
8.140.2.2 create() ................................................................................. 252
8.140.2.3 getAlternateColorSpace() ....................................................... 252
8.140.2.4 getICCProfile() ....................................................................... 253
8.140.2.5 setICCProfile() ....................................................................... 253

8.141 IDOMColorSpaceIndexed Class Reference .............................................. 253
8.141.1 Detailed Description ................................................................... 255
8.141.2 Member Function Documentation .................................................. 255
8.141.2.1 classID() ................................................................................. 255
8.141.2.2 create() ................................................................................. 255
8.141.2.3 create() [1/2] .................................................................... 255
8.141.2.4 create() [2/2] .................................................................... 256
8.141.2.5 getMappingFunction() ............................................................ 256
8.141.2.6 getUnderlyingColorSpace() ................................................... 256

8.142 IDOMColorSpaceLAB Class Reference .................................................. 257
8.142.1 Detailed Description ................................................................... 258
8.142.2 Member Function Documentation .................................................. 258
8.142.2.1 classID() ................................................................................. 258
8.142.2.2 create() ................................................................................. 258
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.142.2.3</td>
<td>getBlackPoint()</td>
<td>259</td>
</tr>
<tr>
<td>8.142.2.4</td>
<td>getRangeAB()</td>
<td>259</td>
</tr>
<tr>
<td>8.142.2.5</td>
<td>getWhitePoint()</td>
<td>259</td>
</tr>
<tr>
<td>8.143</td>
<td>IDOMColorSpacescRGB Class Reference</td>
<td>260</td>
</tr>
<tr>
<td>8.143.1</td>
<td>Detailed Description</td>
<td>261</td>
</tr>
<tr>
<td>8.143.2</td>
<td>Member Function Documentation</td>
<td>261</td>
</tr>
<tr>
<td>8.143.2.1</td>
<td>classID()</td>
<td>261</td>
</tr>
<tr>
<td>8.143.2.2</td>
<td>create()</td>
<td>261</td>
</tr>
<tr>
<td>8.144</td>
<td>IDOMColorSpacesGray Class Reference</td>
<td>262</td>
</tr>
<tr>
<td>8.144.1</td>
<td>Detailed Description</td>
<td>262</td>
</tr>
<tr>
<td>8.144.2</td>
<td>Member Function Documentation</td>
<td>262</td>
</tr>
<tr>
<td>8.144.2.1</td>
<td>classID()</td>
<td>263</td>
</tr>
<tr>
<td>8.144.2.2</td>
<td>create()</td>
<td>263</td>
</tr>
<tr>
<td>8.145</td>
<td>IDOMColorSpacesRGB Class Reference</td>
<td>263</td>
</tr>
<tr>
<td>8.145.1</td>
<td>Detailed Description</td>
<td>264</td>
</tr>
<tr>
<td>8.145.2</td>
<td>Member Function Documentation</td>
<td>264</td>
</tr>
<tr>
<td>8.145.2.1</td>
<td>classID()</td>
<td>264</td>
</tr>
<tr>
<td>8.145.2.2</td>
<td>create()</td>
<td>265</td>
</tr>
<tr>
<td>8.146</td>
<td>IDOMCompositeImage Class Reference</td>
<td>265</td>
</tr>
<tr>
<td>8.146.1</td>
<td>Detailed Description</td>
<td>266</td>
</tr>
<tr>
<td>8.146.2</td>
<td>Member Function Documentation</td>
<td>266</td>
</tr>
<tr>
<td>8.146.2.1</td>
<td>classID()</td>
<td>266</td>
</tr>
<tr>
<td>8.146.2.2</td>
<td>create()</td>
<td>266</td>
</tr>
<tr>
<td>8.146.2.3</td>
<td>getStream()</td>
<td>267</td>
</tr>
<tr>
<td>8.146.2.4</td>
<td>setStream()</td>
<td>267</td>
</tr>
<tr>
<td>8.147</td>
<td>IDOMDePremultiplyFilter Class Reference</td>
<td>268</td>
</tr>
<tr>
<td>8.147.1</td>
<td>Detailed Description</td>
<td>268</td>
</tr>
<tr>
<td>8.148</td>
<td>IDOMDeviceNColorant Class Reference</td>
<td>268</td>
</tr>
<tr>
<td>8.148.1</td>
<td>Detailed Description</td>
<td>269</td>
</tr>
<tr>
<td>8.148.2</td>
<td>Member Function Documentation</td>
<td>269</td>
</tr>
</tbody>
</table>
### 8.152 IDOMFixedPage Class Reference

#### 8.152.1 Detailed Description

#### 8.152.2 Member Function Documentation

- `classID()`: 285
- `create()`: 285
- `getAnnotationManager()`: 286
- `getBleedBox()`: 286
- `getContentBox()`: 286
- `getCropBox()`: 287
- `getHeight()`: 287
- `getLanguage()`: 287
- `getLinkManager()`: 288
- `getPageGroup()`: 288
- `getResourceDictionary()`: 288
- `getThumbnail()`: 289
- `getTrimBox()`: 289
- `getWidth()`: 289
- `isDegenerated()`: 290
- `setBleedBox()`: 290
- `setContentBox()`: 290
- `setCropBox()`: 291
- `setHeight()`: 291
- `setLanguage()`: 291
- `setPageGroup()`: 292
- `setResourceDictionary()`: 292
8.158.1 Detailed Description ........................................ 315
8.158.2 Member Enumeration Documentation .......................... 315
  8.158.2.1 eFontSourceType ...................................... 315
8.158.3 Member Function Documentation ............................... 315
  8.158.3.1 classID() ........................................... 315
  8.158.3.2 determineUri() [1/2] .................................. 316
  8.158.3.3 determineUri() [2/2] .................................. 316
  8.158.3.4 getFontSourceType() .................................. 316

8.159 IDOMFontSourceFromStream Class Reference ...................... 317
8.159.1 Detailed Description ........................................ 318
8.159.2 Member Function Documentation ............................... 318
  8.159.2.1 classID() ........................................... 318
  8.159.2.2 determineUri() [1/2] .................................. 318
  8.159.2.3 determineUri() [2/2] .................................. 318
  8.159.2.4 getStream() .......................................... 319
  8.159.2.5 getStreamLength() .................................... 319

8.160 IDOMFontSourceObfuscationConverter Class Reference .......... 320
8.160.1 Detailed Description ........................................ 321
8.160.2 Member Enumeration Documentation .......................... 321
  8.160.2.1 eOperation .......................................... 321
8.160.3 Member Function Documentation ............................... 321
  8.160.3.1 classID() ........................................... 321
  8.160.3.2 determineUri() [1/2] .................................. 322
  8.160.3.3 determineUri() [2/2] .................................. 322
  8.160.3.4 getInputFontSource() .................................. 322
  8.160.3.5 getStream() .......................................... 323

8.161 IDOMFontSourceStreamFilter Class Reference .................... 323
8.161.1 Detailed Description ........................................ 325
8.161.2 Member Enumeration Documentation .......................... 325
  8.161.2.1 eFontStreamFilterType ................................. 325
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.161.3</td>
<td>Member Function Documentation</td>
<td>325</td>
</tr>
<tr>
<td>8.161.3.1</td>
<td>determineUri() [1/2]</td>
<td>325</td>
</tr>
<tr>
<td>8.161.3.2</td>
<td>determineUri() [2/2]</td>
<td>325</td>
</tr>
<tr>
<td>8.161.3.3</td>
<td>getFontStreamFilterType()</td>
<td>326</td>
</tr>
<tr>
<td>8.161.3.4</td>
<td>getInputFontSource()</td>
<td>326</td>
</tr>
<tr>
<td>8.161.3.5</td>
<td>getStream()</td>
<td>327</td>
</tr>
<tr>
<td>8.162</td>
<td>IDOMForm Class Reference</td>
<td>328</td>
</tr>
<tr>
<td>8.162.1</td>
<td>Detailed Description</td>
<td>329</td>
</tr>
<tr>
<td>8.162.2</td>
<td>Member Function Documentation</td>
<td>329</td>
</tr>
<tr>
<td>8.162.2.1</td>
<td>classID()</td>
<td>329</td>
</tr>
<tr>
<td>8.162.2.2</td>
<td>create()</td>
<td>329</td>
</tr>
<tr>
<td>8.162.2.3</td>
<td>getFormId()</td>
<td>330</td>
</tr>
<tr>
<td>8.162.2.4</td>
<td>getMatrix()</td>
<td>330</td>
</tr>
<tr>
<td>8.162.2.5</td>
<td>getStructureElementReference()</td>
<td>331</td>
</tr>
<tr>
<td>8.162.2.6</td>
<td>setBounds()</td>
<td>331</td>
</tr>
<tr>
<td>8.162.2.7</td>
<td>setMatrix()</td>
<td>331</td>
</tr>
<tr>
<td>8.162.2.8</td>
<td>setStructureElementReference()</td>
<td>332</td>
</tr>
<tr>
<td>8.163</td>
<td>IDOMFormInstance Class Reference</td>
<td>332</td>
</tr>
<tr>
<td>8.163.1</td>
<td>Detailed Description</td>
<td>333</td>
</tr>
<tr>
<td>8.163.2</td>
<td>Member Function Documentation</td>
<td>333</td>
</tr>
<tr>
<td>8.163.2.1</td>
<td>classID()</td>
<td>333</td>
</tr>
<tr>
<td>8.163.2.2</td>
<td>getBlendMode()</td>
<td>333</td>
</tr>
<tr>
<td>8.163.2.3</td>
<td>getForm()</td>
<td>334</td>
</tr>
<tr>
<td>8.163.2.4</td>
<td>getOpacity()</td>
<td>334</td>
</tr>
<tr>
<td>8.163.2.5</td>
<td>getOpacityMask()</td>
<td>334</td>
</tr>
<tr>
<td>8.163.2.6</td>
<td>getRenderTransform()</td>
<td>335</td>
</tr>
<tr>
<td>8.163.2.7</td>
<td>setBlendMode()</td>
<td>335</td>
</tr>
<tr>
<td>8.163.2.8</td>
<td>setForm()</td>
<td>335</td>
</tr>
<tr>
<td>8.163.2.9</td>
<td>setOpacity()</td>
<td>337</td>
</tr>
<tr>
<td>8.163.2.10</td>
<td>setOpacityMask()</td>
<td>337</td>
</tr>
</tbody>
</table>
8.163.2.1 setRenderTransform() .................................................. 337

8.164 IDOMFunction Class Reference ............................................. 339

8.164.1 Detailed Description ....................................................... 340

8.164.2 Member Enumeration Documentation .................................. 340

8.164.2.1 eFunctionType ......................................................... 340

8.164.3 Member Function Documentation ....................................... 340

8.164.3.1 evaluate() [1/2] ...................................................... 340

8.164.3.2 evaluate() [2/2] ...................................................... 341

8.164.3.3 getFunctionType() .................................................... 341

8.164.3.4 getInputDomain() ..................................................... 341

8.164.3.5 getNumInputValues() ............................................... 342

8.164.3.6 getNumOutputValues() ............................................. 342

8.164.3.7 getOutputRange() .................................................... 342

8.165 IDOMGlyph Class Reference ................................................. 343

8.165.1 Detailed Description ...................................................... 344

8.165.2 Member Typedef Documentation ..................................... 345

8.165.2.1 GlyphID .............................................................. 345

8.165.3 Member Enumeration Documentation ................................ 345

8.165.3.1 eGlyphIDSpecial .................................................... 345

8.165.4 Member Function Documentation ..................................... 345

8.165.4.1 classID() ............................................................ 346

8.165.4.2 getAdvanceX() ...................................................... 346

8.165.4.3 getAdvanceY() ...................................................... 346

8.165.4.4 getBounds() ........................................................ 347

8.165.4.5 getColored() ........................................................ 347

8.165.4.6 getGlyphID() ........................................................ 347

8.165.4.7 getGlyphName() [1/2] .............................................. 347

8.165.4.8 getGlyphName() [2/2] .............................................. 348

8.165.4.9 getGlyphUnicode() ................................................. 348

8.165.4.10 getHasCustomAdvance() .......................................... 348
8.175 Member Function Documentation .................................................. 398
  8.175.1 createImageDecoder() .............................................................. 398
  8.175.2 createImageEncoder() .............................................................. 399
  8.175.3 getFirstImageFrame() .............................................................. 399
  8.175.4 getImageProperties() .............................................................. 399
  8.175.5 getImageType() ....................................................................... 400
8.176 IDOMImageBitScalerFilter Class Reference ............................................. 400
  8.176.1 Detailed Description ................................................................. 401
  8.176.2 Member Function Documentation ................................................. 401
    8.176.2.1 classID() ....................................................................... 401
    8.176.2.2 create() ....................................................................... 401
8.177 IDOMImageBleederFilter Class Reference ............................................... 401
  8.177.1 Detailed Description ................................................................. 402
  8.177.2 Member Function Documentation ................................................. 402
    8.177.2.1 classID() ....................................................................... 402
    8.177.2.2 create() ....................................................................... 402
8.178 IDOMImageBrush Class Reference ......................................................... 403
  8.178.1 Detailed Description ................................................................. 404
  8.178.2 Member Function Documentation ................................................. 405
    8.178.2.1 classID() ....................................................................... 405
    8.178.2.2 create() ....................................................................... 405
    8.178.2.3 getEquivalentTilingBrush() ............................................... 405
    8.178.2.4 getEquivalentVisualBrush() ............................................... 407
    8.178.2.5 getICCProfile() .................................................................. 407
    8.178.2.6 getImageSource() ............................................................... 408
    8.178.2.7 getTileMode() .................................................................. 408
    8.178.2.8 getViewBox() .................................................................. 408
    8.178.2.9 getViewBoxUnits() ............................................................. 409
    8.178.2.10 getViewPort() ................................................................. 409
    8.178.2.11 getViewPortUnits() .......................................................... 409
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.192.2</td>
<td>addNamespace()</td>
</tr>
<tr>
<td>8.192.3</td>
<td>addNamespace()</td>
</tr>
<tr>
<td>8.192.4</td>
<td>addParameterInit()</td>
</tr>
<tr>
<td>8.192.5</td>
<td>addToInitString()</td>
</tr>
<tr>
<td>8.192.6</td>
<td>classID()</td>
</tr>
<tr>
<td>8.192.7</td>
<td>convertToQName()</td>
</tr>
<tr>
<td>8.192.8</td>
<td>findChild()</td>
</tr>
<tr>
<td>8.192.9</td>
<td>findChild()</td>
</tr>
<tr>
<td>8.192.10</td>
<td>findNamespaceByName()</td>
</tr>
<tr>
<td>8.192.11</td>
<td>findNamespaceByPrefix()</td>
</tr>
<tr>
<td>8.192.12</td>
<td>getLevel()</td>
</tr>
<tr>
<td>8.192.13</td>
<td>getModified()</td>
</tr>
<tr>
<td>8.192.14</td>
<td>getNamespaceCollectionEnum()</td>
</tr>
<tr>
<td>8.192.15</td>
<td>getNamespacesCount()</td>
</tr>
<tr>
<td>8.192.16</td>
<td>getRootNode()</td>
</tr>
<tr>
<td>8.192.17</td>
<td>getVersion()</td>
</tr>
<tr>
<td>8.192.18</td>
<td>isValid()</td>
</tr>
<tr>
<td>8.192.19</td>
<td>loadFromFile()</td>
</tr>
<tr>
<td>8.192.20</td>
<td>loadFromInitString()</td>
</tr>
<tr>
<td>8.192.21</td>
<td>loadFromStream()</td>
</tr>
<tr>
<td>8.192.22</td>
<td>setLevel()</td>
</tr>
<tr>
<td>8.192.23</td>
<td>setModified()</td>
</tr>
<tr>
<td>8.192.24</td>
<td>setVersion()</td>
</tr>
</tbody>
</table>

8.193 IDOMJobTkGenericCharacterData Class Reference
8.194 IDOMJobTkGenericNode Class Reference

8.194.1 Detailed Description

8.194.2 Member Function Documentation

8.194.2.1 addAttribute()
8.194.2.2 classID()
8.194.2.3 getAttributeAtIndex()
8.194.2.4 getNumAttributes()
8.194.2.5 getQName()

8.195 IDOMJobTkNode Class Reference

8.195.1 Detailed Description

8.195.2 Member Function Documentation

8.195.2.1 classID()
8.195.2.2 findChild()[1/2]
8.195.2.3 findChild()[2/2]
8.195.2.4 getChildValue()
8.195.2.5 getJobTkContent()
8.195.2.6 getJobTkNodeType()
8.195.2.7 getQName()
8.195.2.8 getQNameAsString()
8.195.2.9 setJobTkNodeType()
8.195.2.10 setQName()

8.196 IDOMJobTkOwner Class Reference

8.196.1 Detailed Description

8.196.2 Member Function Documentation

8.196.2.1 classID()
8.196.2.2 getJobTicket()
8.196.2.3 setJobTicket()

8.197 IDOMJobTkValue Class Reference

8.197.1 Detailed Description

8.197.2 Member Function Documentation
8.204.2.11 getDOMid() ............................................. 492
8.204.2.12 getFirstChild() ..................................... 492
8.204.2.13 getFlags() ............................................ 493
8.204.2.14 getLastChild() ....................................... 493
8.204.2.15 getNextChild() ...................................... 493
8.204.2.16 getNextSibling() .................................... 493
8.204.2.17 getNodeType() ........................................ 494
8.204.2.18 getParentNode() ..................................... 494
8.204.2.19 getPreviousChild() ................................. 494
8.204.2.20 getPreviousSibling() ............................... 494
8.204.2.21 getProperty() ........................................ 495
8.204.2.22 getPropertyCollectionEnum() ..................... 495
8.204.2.23 hasChildNodes() .................................... 495
8.204.2.24 insertChild() ........................................ 495
8.204.2.25 isAncestor() ......................................... 496
8.204.2.26 isComplete() ......................................... 496
8.204.2.27 removeProperty() ................................... 496
8.204.2.28 replaceChild() ....................................... 497
8.204.2.29 setDOMid() ........................................... 497
8.204.2.30 setNextSibling() .................................... 497
8.204.2.31 setParentNode() ..................................... 498
8.204.2.32 setPreviousSibling() ............................... 498
8.204.2.33 setProperty() ........................................ 498
8.204.2.34 walkTree() ........................................... 499

8.205 IDOMNodeFlags Class Reference ........................................ 499
8.205.1 Detailed Description ...................................... 500
8.205.2 Member Enumeration Documentation .................... 500
8.205.2.1 DOMNodeFlags ........................................ 500
8.205.3 Member Function Documentation ......................... 500
8.205.3.1 get() .................................................. 501
8.205.3.2 set() ................................................................. 501

8.206 IDOMNullBrush Class Reference ................................................................. 501
  8.206.1 Detailed Description ................................................................. 502
  8.206.2 Member Function Documentation ................................................................. 502
    8.206.2.1 classID() ................................................................. 503
    8.206.2.2 create() ................................................................. 503

8.207 IDOMOPI Class Reference ................................................................. 503
  8.207.1 Detailed Description ................................................................. 504
  8.207.2 Member Function Documentation ................................................................. 504
    8.207.2.1 getVersion() ................................................................. 504

8.208 IDOMOPI13 Class Reference ................................................................. 505
  8.208.1 Detailed Description ................................................................. 506
  8.208.2 Member Function Documentation ................................................................. 507
    8.208.2.1 addTag() ................................................................. 507
    8.208.2.2 deleteTag() ................................................................. 507
    8.208.2.3 getColorName() ................................................................. 507
    8.208.2.4 getColorType() ................................................................. 508
    8.208.2.5 getColorValue() ................................................................. 508
    8.208.2.6 getComments() ................................................................. 508
    8.208.2.7 getCropFixed() ................................................................. 509
    8.208.2.8 getCropRect() ................................................................. 509
    8.208.2.9 getF() ................................................................. 510
    8.208.2.10 getGrayMap() ................................................................. 510
    8.208.2.11 getGrayMapSize() ................................................................. 510
    8.208.2.12 getID() ................................................................. 511
    8.208.2.13 getImageType() ................................................................. 511
    8.208.2.14 getOverPrint() ................................................................. 511
    8.208.2.15 getPosition() ................................................................. 512
    8.208.2.16 getResolution() ................................................................. 512
    8.208.2.17 getSize() ................................................................. 512
8.211.3.8 getTextStyle() ........................................ 524
8.211.3.9 setDescription() ........................................ 525
8.211.3.10 setExpanded() ........................................ 525
8.211.3.11 setLanguage() ........................................ 525
8.211.3.12 setStructureElement() ........................................ 526
8.211.3.13 setTarget() ........................................ 526
8.211.3.14 setTextColor() ........................................ 526
8.211.3.15 setTextStyle() ........................................ 528

8.212 IDOMPage Class Reference ........................................ 528
8.212.1 Detailed Description ........................................ 529
8.212.2 Member Function Documentation ........................................ 529
  8.212.2.1 classID() ........................................ 529

8.213 IDOMPageRectTarget Class Reference ........................................ 530
8.213.1 Detailed Description ........................................ 531
8.213.2 Member Function Documentation ........................................ 531
  8.213.2.1 create() ........................................ 531
  8.213.2.2 getBottom() ........................................ 532
  8.213.2.3 getFitType() ........................................ 532
  8.213.2.4 getLeft() ........................................ 533
  8.213.2.5 getPageId() ........................................ 533
  8.213.2.6 getRight() ........................................ 533
  8.213.2.7 getTargetType() ........................................ 533
  8.213.2.8 getTop() ........................................ 534
  8.213.2.9 getZoom() ........................................ 534
  8.213.2.10 setBottom() ........................................ 534
  8.213.2.11 setFitType() ........................................ 534
  8.213.2.12 setLeft() ........................................ 535
  8.213.2.13 setPageId() ........................................ 535
  8.213.2.14 setRight() ........................................ 535
  8.213.2.15 setTop() ........................................ 535
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.213.2.1</td>
<td>setZoom()</td>
<td>536</td>
</tr>
<tr>
<td>8.214</td>
<td>IDOMPageTarget Class Reference</td>
<td>536</td>
</tr>
<tr>
<td>8.214.1</td>
<td>Detailed Description</td>
<td>537</td>
</tr>
<tr>
<td>8.214.2</td>
<td>Member Function Documentation</td>
<td>537</td>
</tr>
<tr>
<td>8.214.2.1</td>
<td>getTargetPage()</td>
<td>538</td>
</tr>
<tr>
<td>8.214.2.2</td>
<td>getTargetType()</td>
<td>538</td>
</tr>
<tr>
<td>8.214.2.3</td>
<td>setTargetPage()</td>
<td>538</td>
</tr>
<tr>
<td>8.215</td>
<td>IDOMPathFigure Class Reference</td>
<td>538</td>
</tr>
<tr>
<td>8.215.1</td>
<td>Detailed Description</td>
<td>540</td>
</tr>
<tr>
<td>8.215.2</td>
<td>Member Function Documentation</td>
<td>540</td>
</tr>
<tr>
<td>8.215.2.1</td>
<td>addSegment()</td>
<td>540</td>
</tr>
<tr>
<td>8.215.2.2</td>
<td>classID()</td>
<td>541</td>
</tr>
<tr>
<td>8.215.2.3</td>
<td>getBounds()</td>
<td>541</td>
</tr>
<tr>
<td>8.215.2.4</td>
<td>getIsClosed()</td>
<td>541</td>
</tr>
<tr>
<td>8.215.2.5</td>
<td>getIsFilled()</td>
<td>542</td>
</tr>
<tr>
<td>8.215.2.6</td>
<td>getSegmentCollectionEnum()</td>
<td>542</td>
</tr>
<tr>
<td>8.215.2.7</td>
<td>getSegmentsCount()</td>
<td>542</td>
</tr>
<tr>
<td>8.215.2.8</td>
<td>getStartPoint()</td>
<td>542</td>
</tr>
<tr>
<td>8.215.2.9</td>
<td>setIsClosed()</td>
<td>542</td>
</tr>
<tr>
<td>8.215.2.10</td>
<td>setIsFilled()</td>
<td>543</td>
</tr>
<tr>
<td>8.215.2.11</td>
<td>setStartPoint()</td>
<td>543</td>
</tr>
<tr>
<td>8.216</td>
<td>IDOMPathGeometry Class Reference</td>
<td>544</td>
</tr>
<tr>
<td>8.216.1</td>
<td>Detailed Description</td>
<td>545</td>
</tr>
<tr>
<td>8.216.2</td>
<td>Member Enumeration Documentation</td>
<td>546</td>
</tr>
<tr>
<td>8.216.2.1</td>
<td>eFillRule</td>
<td>546</td>
</tr>
<tr>
<td>8.216.3</td>
<td>Member Function Documentation</td>
<td>546</td>
</tr>
<tr>
<td>8.216.3.1</td>
<td>addFigure()</td>
<td>546</td>
</tr>
<tr>
<td>8.216.3.2</td>
<td>classID()</td>
<td>547</td>
</tr>
<tr>
<td>8.216.3.3</td>
<td>create()</td>
<td>547</td>
</tr>
<tr>
<td>8.216.3.4</td>
<td>getBounds()</td>
<td>547</td>
</tr>
</tbody>
</table>
8.217.3.14 getNavigateLink() ................................................. 563
8.217.3.15 getOpacity() .................................................. 564
8.217.3.16 getOpacityMask() ............................................. 564
8.217.3.17 getPathData() .................................................. 564
8.217.3.18 getRenderTransform() ........................................ 565
8.217.3.19 getShape() ..................................................... 565
8.217.3.20 getShouldZeroWidthLinesBeVisible() ...................... 566
8.217.3.21 getSnapsToDevicePixels() .................................... 566
8.217.3.22 getStroke() ..................................................... 566
8.217.3.23 getStrokeDashCollectionEnum() .............................. 567
8.217.3.24 getStrokeDashLineCap() ..................................... 567
8.217.3.25 getStrokeDashOffset() ....................................... 567
8.217.3.26 getStrokeDashsCount() ...................................... 568
8.217.3.27 getStrokeEndLineCap() ....................................... 568
8.217.3.28 getStrokeLineJoin() .......................................... 568
8.217.3.29 getStrokeMiterLimit() ........................................ 569
8.217.3.30 getStrokeMiterLimitTreatment() ......................... 569
8.217.3.31 getStrokeStartLineCap() ..................................... 569
8.217.3.32 getStrokeThickness() ........................................ 570
8.217.3.33 setAutomationPropertiesHelpText() ........................ 570
8.217.3.34 setAutomationPropertiesName() ............................ 570
8.217.3.35 setBlendMode() ................................................ 571
8.217.3.36 setClip() ....................................................... 571
8.217.3.37 setEdgeMode() ................................................ 571
8.217.3.38 setFill() ........................................................ 572
8.217.3.39 setLanguage() ................................................. 572
8.217.3.40 setNavigateLink() ............................................ 573
8.217.3.41 setOpacity() ................................................... 573
8.217.3.42 setOpacityMask() ............................................. 573
8.217.3.43 setPathData() .................................................. 574
8.228 IDOMRawDataFile Class Reference ........................................... 609
  8.228.1 Detailed Description ..................................................... 610
  8.228.2 Member Function Documentation ....................................... 610
    8.228.2.1 classID() ........................................................... 611

8.229 IDOMRawImage Class Reference .............................................. 611
  8.229.1 Detailed Description ..................................................... 612
  8.229.2 Member Function Documentation ....................................... 612
    8.229.2.1 classID() ........................................................... 612
    8.229.2.2 createWriterAndImage() ........................................ 613
    8.229.2.3 getSynthetic() ................................................... 613

8.230 IDOMRecombineAlpha Class Reference ....................................... 614
  8.230.1 Detailed Description ..................................................... 614

8.231 IDOMRecombineImage Class Reference ....................................... 614
  8.231.1 Detailed Description ..................................................... 615
  8.231.2 Member Function Documentation ....................................... 615
    8.231.2.1 classID() ........................................................... 615
    8.231.2.2 getStream() ....................................................... 615
    8.231.2.3 setStream() ....................................................... 616

8.232 IDOMResource Class Reference .............................................. 616
  8.232.1 Detailed Description ..................................................... 617
  8.232.2 Member Function Documentation ....................................... 617
    8.232.2.1 getStream() ....................................................... 617
    8.232.2.2 getStreamLength() .............................................. 618
    8.232.2.3 getUri() ........................................................... 618
    8.232.2.4 setStream() ....................................................... 619
    8.232.2.5 setUri() ........................................................... 619

8.233 IDOMResourceDictionary Class Reference ................................... 619
  8.233.1 Detailed Description ..................................................... 620
  8.233.2 Member Function Documentation ....................................... 621
    8.233.2.1 classID() ........................................................... 621
8.241.2.3 difference() .......................................................... 660
8.241.2.4 getAsImage() ......................................................... 660
8.241.2.5 getBounds() ......................................................... 661
8.241.2.6 getIsEmpty() ....................................................... 661
8.241.2.7 getIsRect() .......................................................... 661
8.241.2.8 getResolution() ..................................................... 662
8.241.2.9 intersect() ........................................................... 662
8.241.2.10 intersects() ........................................................ 662
8.241.2.11 isEqualTo() ........................................................ 663
8.241.2.12 unite() ............................................................... 663

8.242 IDOMSoftMaskBrush Class Reference .................................. 663

8.242.1 Detailed Description .................................................. 665
8.242.2 Member Function Documentation .................................... 665
  8.242.2.1 classID() .......................................................... 665
  8.242.2.2 create() ........................................................... 665
  8.242.2.3 getBackdropColor() ............................................. 666
  8.242.2.4 getGroup() ......................................................... 666
  8.242.2.5 getSoftMaskType() ............................................. 666
  8.242.2.6 getTransferFunction() ......................................... 667

8.243 IDOMSolidColorBrush Class Reference .................................. 667

8.243.1 Detailed Description .................................................. 668
8.243.2 Member Function Documentation .................................... 668
  8.243.2.1 classID() .......................................................... 668
  8.243.2.2 create() ........................................................... 668
  8.243.2.3 getColor() ........................................................ 669
  8.243.2.4 setColor() ......................................................... 669

8.244 IDOMStandardPDFSecurityInfo Class Reference .................... 670

8.244.1 Detailed Description .................................................. 671
8.244.2 Member Enumeration Documentation ................................ 671
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.245</td>
<td>IDOMStitchingFunction Class Reference</td>
<td>671</td>
</tr>
<tr>
<td>8.245.1</td>
<td>Detailed Description</td>
<td>673</td>
</tr>
<tr>
<td>8.245.2</td>
<td>Member Function Documentation</td>
<td>673</td>
</tr>
<tr>
<td>8.245.2.1</td>
<td>classID()</td>
<td>673</td>
</tr>
<tr>
<td>8.245.2.2</td>
<td>evaluate()</td>
<td>673</td>
</tr>
<tr>
<td>8.245.2.3</td>
<td>evaluate()</td>
<td>674</td>
</tr>
<tr>
<td>8.245.2.4</td>
<td>getBoundsVector()</td>
<td>674</td>
</tr>
<tr>
<td>8.245.2.5</td>
<td>getEncodeVector()</td>
<td>674</td>
</tr>
<tr>
<td>8.245.2.6</td>
<td>getFunctionAtIndex()</td>
<td>675</td>
</tr>
<tr>
<td>8.245.2.7</td>
<td>getNumFunctions()</td>
<td>675</td>
</tr>
<tr>
<td>8.246</td>
<td>IDOMTarget Class Reference</td>
<td>675</td>
</tr>
<tr>
<td>8.246.1</td>
<td>Detailed Description</td>
<td>676</td>
</tr>
<tr>
<td>8.246.2</td>
<td>Member Enumeration Documentation</td>
<td>676</td>
</tr>
<tr>
<td>8.246.2.1</td>
<td>eTargetType</td>
<td>676</td>
</tr>
<tr>
<td>8.246.3</td>
<td>Member Function Documentation</td>
<td>677</td>
</tr>
<tr>
<td>8.246.3.1</td>
<td>targetType()</td>
<td>677</td>
</tr>
<tr>
<td>8.247</td>
<td>IDOMTIFFImage Class Reference</td>
<td>678</td>
</tr>
<tr>
<td>8.247.1</td>
<td>Detailed Description</td>
<td>679</td>
</tr>
<tr>
<td>8.247.2</td>
<td>Member Enumeration Documentation</td>
<td>679</td>
</tr>
<tr>
<td>8.247.2.1</td>
<td>eTIFFCompression</td>
<td>679</td>
</tr>
<tr>
<td>8.247.3</td>
<td>Member Function Documentation</td>
<td>679</td>
</tr>
<tr>
<td>8.247.3.1</td>
<td>classID()</td>
<td>679</td>
</tr>
<tr>
<td>8.247.3.2</td>
<td>create()</td>
<td>679</td>
</tr>
<tr>
<td>8.247.3.3</td>
<td>encode()</td>
<td>680</td>
</tr>
<tr>
<td>8.247.3.4</td>
<td>encode()</td>
<td>680</td>
</tr>
<tr>
<td>8.248</td>
<td>IDOMTilingPatternBrush Class Reference</td>
<td>681</td>
</tr>
<tr>
<td>8.248.1</td>
<td>Detailed Description</td>
<td>682</td>
</tr>
<tr>
<td>8.248.2</td>
<td>Member Function Documentation</td>
<td>683</td>
</tr>
<tr>
<td>8.248.2.1</td>
<td>classID()</td>
<td>683</td>
</tr>
<tr>
<td>8.248.2.2</td>
<td>create()</td>
<td>683</td>
</tr>
</tbody>
</table>
8.250.2.7 getOpacity() ................................................................. 696
8.250.2.8 getOpacityMask() ......................................................... 696
8.250.2.9 setBlendMode() ............................................................ 697
8.250.2.10 setColorSpace() .......................................................... 697
8.250.2.11 setIsIsolated() ............................................................. 698
8.250.2.12 setIsKnockout() ........................................................... 698
8.250.2.13 setOpacity() ............................................................... 698
8.250.2.14 setOpacityMask() ......................................................... 699

8.251 IDOMType3Font Class Reference ........................................... 699

8.251.1 Detailed Description ......................................................... 701
8.251.2 Member Function Documentation ....................................... 701
8.251.2.1 addGlyph() ............................................................... 701
8.251.2.2 classID() ................................................................. 701
8.251.2.3 deleteGlyphs() .......................................................... 702
8.251.2.4 getBBox() ................................................................. 702
8.251.2.5 getGlyph() [1/3] ........................................................ 702
8.251.2.6 getGlyph() [2/3] ........................................................ 703
8.251.2.7 getGlyph() [3/3] ........................................................ 703
8.251.2.8 getId() ................................................................. 703

8.252 IDOMVisualBrush Class Reference ........................................ 704

8.252.1 Detailed Description ......................................................... 705
8.252.2 Member Function Documentation ....................................... 705
8.252.2.1 classID() ............................................................... 706
8.252.2.2 create() ................................................................. 706
8.252.2.3 getEquivalentSimpleVisualBrush() ................................ 706
8.252.2.4 getEquivalentTilingBrush() ........................................ 707
8.252.2.5 getTileMode() .......................................................... 707
8.252.2.6 getViewBox() ............................................................ 708
8.252.2.7 getViewBoxUnits() ....................................................... 708
8.252.2.8 getViewPort() ............................................................ 708
8.257.2.2 getAscentDescentRatio() ................................................. 720
8.257.2.3 getAverageCharAspectRatio() ......................................... 721
8.257.2.4 getFontDecorationFlags() .............................................. 721
8.257.2.5 getFontFamily() .......................................................... 721
8.257.2.6 getFontFilePath() ....................................................... 721
8.257.2.7 getFontName() ............................................................ 722
8.257.2.8 getFontWeight() .......................................................... 722
8.257.2.9 loadFont() ............................................................... 723
8.257.2.10 setFontFilePath() ...................................................... 723

8.258 IEDLNamespace Class Reference ............................................. 724
8.258.1 Detailed Description .......................................................... 725
8.258.2 Member Function Documentation .......................................... 725
  8.258.2.1 classID() .............................................................. 725
  8.258.2.2 getNamespace() ....................................................... 725
  8.258.2.3 getPrefix() ........................................................... 725
  8.258.2.4 setNamespace() ...................................................... 726
  8.258.2.5 setPrefix() ........................................................... 726

8.259 IEDLObject Class Reference .................................................. 727
8.259.1 Detailed Description .......................................................... 728
8.259.2 Member Function Documentation .......................................... 728
  8.259.2.1 clone() .............................................................. 728
  8.259.2.2 getClassID() ....................................................... 729
  8.259.2.3 init() ............................................................... 729

8.260 IEDLStream Class Reference .................................................. 729
8.260.1 Detailed Description .......................................................... 730
8.260.2 Member Function Documentation .......................................... 730
  8.260.2.1 getPos() .............................................................. 731
  8.260.2.2 isValid() ........................................................... 731
  8.260.2.3 open() .............................................................. 731

8.261 IEDLTempStore Class Reference ............................................. 732
8.261.1 Detailed Description ........................................... 733
8.261.2 Member Function Documentation ............................... 733
  8.261.2.1 classID() ..................................................... 733
  8.261.2.2 createTemporaryObject() .................................. 733
  8.261.2.3 createTemporaryObjectWithContents() ..................... 733
  8.261.2.4 createTemporaryReaderWriter() ........................... 734
  8.261.2.5 createTemporaryReaderWriterPair() ...................... 734
  8.261.2.6 createTemporaryStreamWithContents() ..................... 734
8.262 IEDLTempStoreObject Class Reference ............................ 735
  8.262.1 Detailed Description .......................................... 736
  8.262.2 Member Function Documentation ............................... 736
    8.262.2.1 createReader() ............................................ 736
    8.262.2.2 createWriter() ............................................. 736
8.263 IEDLTime Class Reference ......................................... 737
  8.263.1 Detailed Description .......................................... 738
  8.263.2 Member Function Documentation ............................... 738
    8.263.2.1 compare() ................................................. 738
    8.263.2.2 fromPDFDate() ............................................. 739
    8.263.2.3 fromW3CDTF() .............................................. 739
    8.263.2.4 getDay() ................................................... 739
    8.263.2.5 getMonth() ................................................ 740
    8.263.2.6 getTime() ................................................. 740
    8.263.2.7 getYear() ................................................ 740
    8.263.2.8 isEqualTo() ............................................... 741
    8.263.2.9 setDay() .................................................. 741
    8.263.2.10 setMonth() ............................................... 741
    8.263.2.11 setTime() ................................................ 741
    8.263.2.12 setYear() ................................................ 742
    8.263.2.13 toPDFDate() ............................................. 742
    8.263.2.14 toW3CDTF() .............................................. 743

Generated by Doxygen
8.266.2.9 getPathToWidget() .............................................. 753
8.266.2.10 getPathToWidget() .......................................... 754
8.266.2.11 getWidgets() .................................................. 754
8.266.2.12 removeField() ................................................ 754
8.266.2.13 removeWidget() ............................................. 755
8.266.2.14 removeWidget() ............................................. 755
8.266.2.15 setNeedAppearances() .................................... 755
8.266.2.16 widgetInTree() ............................................. 755
8.266.2.17 widgetInTree() ............................................. 756

8.267 JawsMako::IFormField Class Reference ........................................... 756
8.267.1 Detailed Description ............................................. 757
8.267.2 Member Function Documentation .................................... 758
8.267.2.1 addChildField() ............................................ 758
8.267.2.2 addChildWidget() .......................................... 758
8.267.2.3 addChildWidget() .......................................... 758
8.267.2.4 clone() ...................................................... 759
8.267.2.5 fieldInSubtree() ............................................ 759
8.267.2.6 getChildFields() ........................................... 759
8.267.2.7 getChildWidgets() .......................................... 759
8.267.2.8 getFieldId() ................................................ 760
8.267.2.9 getType() ................................................... 760
8.267.2.10 getPartialName() ......................................... 760
8.267.2.11 removeChildField() ....................................... 760
8.267.2.12 removeChildWidget() ...................................... 761
8.267.2.13 removeChildWidget() ...................................... 761
8.267.2.14 setPartialName() .......................................... 761
8.267.2.15 widgetInSubtree() ....................................... 761
8.267.2.16 widgetInSubtree() ....................................... 762

8.268 JawsMako::IFormUnpackerTransform Class Reference ............................ 762
8.268.1 Detailed Description ............................................ 763
8.268.2 Member Function Documentation ........................................ 763
  8.268.2.1 create() ......................................................... 763

8.269 JawsMako::IFreeTextAnnotation Class Reference ............................ 764
  8.269.1 Detailed Description .................................................. 765
  8.269.2 Member Function Documentation ........................................ 765
    8.269.2.1 getCalloutLine() ............................................. 765
    8.269.2.2 getRectInset() ............................................... 765

8.270 JawsMako::IJawsRenderer::IHalftone Class Reference ....................... 766
  8.270.1 Detailed Description .................................................. 766

8.271 IImageDecoder Class Reference ............................................. 766
  8.271.1 Detailed Description .................................................. 767

8.272 JawsMako::IImageDownsamplerTransform Class Reference .................... 767
  8.272.1 Detailed Description .................................................. 769
  8.272.2 Member Function Documentation ........................................ 769
    8.272.2.1 create() ..................................................... 769
    8.272.2.2 setDownsampleMaskedImages() ................................ 770
    8.272.2.3 setUseMaskResolutionForMaskedImages() ....................... 770

8.273 IImageEncoder Class Reference ............................................. 771
  8.273.1 Detailed Description .................................................. 771

8.274 JawsMako::IImageEncoderTransform Class Reference ........................ 772
  8.274.1 Detailed Description .................................................. 773
  8.274.2 Member Enumeration Documentation .................................... 773
    8.274.2.1 eEncodeFormat ............................................... 773

8.274.3 Member Function Documentation ........................................... 774
  8.274.3.1 create() ..................................................... 774
  8.274.3.2 setColorTIFFCompression() .................................... 774
  8.274.3.3 setGrayTIFFCompression() ..................................... 774
  8.274.3.4 setJPEGQuality() ............................................... 775
  8.274.3.5 setMonoTIFFCompression() ..................................... 775
  8.274.3.6 setPreferredColorFormat() ..................................... 775
8.286.2.2 render() ........................................... 810
8.286.2.3 renderAntiAliased() ............................. 811
8.286.2.4 renderAntiAliasedToFrameBuffer() ............ 811
8.286.2.5 renderMonochrome() ............................ 812
8.286.2.6 renderMonochromeToFrameBuffer() ............ 813
8.286.2.7 renderSeparations() ............................ 813
8.286.2.8 renderToFrameBuffer() .......................... 814
8.286.2.9 renderToFrameBufferPadAndReverse() ........... 815

8.287 JawsMako::ILineAnnotation Class Reference ............... 816
8.287.1 Detailed Description ................................ 817
8.287.2 Member Function Documentation ..................... 818
8.287.2.1 getCaptionOffset() ................................ 818
8.287.2.2 getInteriorColor() ............................. 818
8.287.2.3 getLeaderLineExtensionsLength() ............... 818
8.287.2.4 getLeaderLineLength() .......................... 819
8.287.2.5 getLeaderLineOffset() .......................... 819
8.287.2.6 getLineEndpoints() ............................. 819
8.287.2.7 setInteriorColor() ............................. 819
8.287.2.8 setLineEndpoints() ............................. 820

8.288 JawsMako::ILinkAnnotation Class Reference ............... 820
8.288.1 Detailed Description ................................ 821
8.288.2 Member Function Documentation ..................... 821
8.288.2.1 getQuadPoints() ................................ 821
8.288.2.2 setQuadPoints() ................................ 821

8.289 JawsMako::IMarkedContentArtifactDetails Class Reference ............... 822
8.289.1 Detailed Description ................................ 822

8.290 JawsMako::IMarkedContentDetails Class Reference .......... 823
8.290.1 Detailed Description ................................ 823

8.291 JawsMako::IMarkedContentStructureDetails Class Reference ............... 824
8.291.1 Detailed Description ................................ 824
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.312</td>
<td>JawsMako::IPageLayoutData Class Reference</td>
<td>874</td>
</tr>
<tr>
<td>8.312.1</td>
<td>Detailed Description</td>
<td>875</td>
</tr>
<tr>
<td>8.312.2</td>
<td>Member Function Documentation</td>
<td>875</td>
</tr>
<tr>
<td>8.312.2.1</td>
<td>create()</td>
<td>875</td>
</tr>
<tr>
<td>8.313</td>
<td>JawsMako::IPageLayoutNode Class Reference</td>
<td>876</td>
</tr>
<tr>
<td>8.313.1</td>
<td>Detailed Description</td>
<td>877</td>
</tr>
<tr>
<td>8.313.2</td>
<td>Member Function Documentation</td>
<td>877</td>
</tr>
<tr>
<td>8.313.2.1</td>
<td>create()</td>
<td>877</td>
</tr>
<tr>
<td>8.314</td>
<td>JawsMako::IPCL5Input Class Reference</td>
<td>877</td>
</tr>
<tr>
<td>8.314.1</td>
<td>Detailed Description</td>
<td>878</td>
</tr>
<tr>
<td>8.314.2</td>
<td>Member Function Documentation</td>
<td>879</td>
</tr>
<tr>
<td>8.314.2.1</td>
<td>create()</td>
<td>879</td>
</tr>
<tr>
<td>8.314.2.2</td>
<td>enableUnencapsulatedMode()</td>
<td>879</td>
</tr>
<tr>
<td>8.314.2.3</td>
<td>setDefaultCloseOperation()</td>
<td>879</td>
</tr>
<tr>
<td>8.314.2.4</td>
<td>setDefaultCloseOperation()</td>
<td>880</td>
</tr>
<tr>
<td>8.314.2.5</td>
<td>setDefaultDuplexBindingMode()</td>
<td>880</td>
</tr>
<tr>
<td>8.314.2.6</td>
<td>setDefaultLandscape()</td>
<td>880</td>
</tr>
<tr>
<td>8.314.2.7</td>
<td>setDefaultManualFeed()</td>
<td>880</td>
</tr>
<tr>
<td>8.314.2.8</td>
<td>setDefaultPaperSize()</td>
<td>881</td>
</tr>
<tr>
<td>8.314.2.9</td>
<td>setDefaultsFromPjl()</td>
<td>881</td>
</tr>
<tr>
<td>8.314.2.10</td>
<td>setRopResolution()</td>
<td>881</td>
</tr>
<tr>
<td>8.315</td>
<td>JawsMako::IPCL5Output Class Reference</td>
<td>882</td>
</tr>
<tr>
<td>8.315.1</td>
<td>Detailed Description</td>
<td>883</td>
</tr>
<tr>
<td>8.315.2</td>
<td>Member Function Documentation</td>
<td>883</td>
</tr>
<tr>
<td>8.315.2.1</td>
<td>setEmitPjl()</td>
<td>883</td>
</tr>
<tr>
<td>8.315.2.2</td>
<td>setImageCompression()</td>
<td>883</td>
</tr>
<tr>
<td>8.315.2.3</td>
<td>setMediaSource()</td>
<td>884</td>
</tr>
<tr>
<td>8.315.2.4</td>
<td>setOpenStream()</td>
<td>884</td>
</tr>
<tr>
<td>8.315.2.5</td>
<td>setResolution()</td>
<td>884</td>
</tr>
<tr>
<td>8.315.2.6</td>
<td>setVersion()</td>
<td>885</td>
</tr>
</tbody>
</table>
8.316 JawsMako::IPCLXLInput Class Reference ................................. 885

8.316.1 Detailed Description .................................................. 886

8.316.2 Member Function Documentation ..................................... 886

8.316.2.1 create() ......................................................... 886
8.316.2.2 enableUnencapsulatedMode() .................................... 887
8.316.2.3 setDefaultCopies() .............................................. 887
8.316.2.4 setDefaultDuplex() .............................................. 887
8.316.2.5 setDefaultDuplexBindingMode() ................................ 888
8.316.2.6 setDefaultLandscape() ........................................... 888
8.316.2.7 setDefaultManualFeed() ......................................... 888
8.316.2.8 setDefaultPaperSize() ........................................... 888
8.316.2.9 setDefaultsFromPjl() ............................................ 889
8.316.2.10 setRopResolution() ............................................. 889

8.317 JawsMako::IPCLXLOutput Class Reference ............................... 890

8.317.1 Detailed Description .................................................. 890

8.317.2 Member Function Documentation ..................................... 891

8.317.2.1 setEmitPjl() ..................................................... 891
8.317.2.2 setOpenStream() ................................................ 891

8.318 JawsMako::IPDFInput Class Reference .................................. 891

8.318.1 Detailed Description .................................................. 892

8.318.2 Member Function Documentation ..................................... 893

8.318.2.1 create() ......................................................... 893
8.318.2.2 scanPdfForFonts() ............................................... 893
  [1/3]
8.318.2.3 scanPdfForFonts() ............................................... 893
  [2/3]
8.318.2.4 scanPdfForFonts() ............................................... 894
  [3/3]
8.318.2.5 setFailOnFontFallback() ........................................ 894
8.318.2.6 setPassword() ................................................... 894

8.319 JawsMako::IPDFOutput Class Reference .................................. 895

8.319.1 Detailed Description .................................................. 897

8.319.2 Member Enumeration Documentation .................................. 898
8.319.2.1 ePDFVersion ........................................... 898
8.319.2.2 ePdfXDeviceNHandling .............................. 899
8.319.2.3 ePdfXExtendedGraphicsStateHandling .............. 899
8.319.2.4 ePdfXOptionalContentHandling ...................... 899
8.319.3 Member Function Documentation ........................ 900
  8.319.3.1 setAllowRestrictedFonts() ......................... 900
  8.319.3.2 setAlwaysEmbedFonts() .......................... 900
  8.319.3.3 setAutoRotatePages() ............................ 901
  8.319.3.4 setColorImageMaxResolution() .................... 901
  8.319.3.5 setCompressObjects() ............................ 901
  8.319.3.6 setCompressPages() ............................... 902
  8.319.3.7 setConvertAllColors() ............................ 902
  8.319.3.8 setConvertGray() ................................ 902
  8.319.3.9 setDownsampleMaskedImages() ..................... 903
  8.319.3.10 setEmbedBase14Fonts() .......................... 903
  8.319.3.11 setEmbedFonts() ................................ 903
  8.319.3.12 setEnableIncrementalOutput() .................... 904
  8.319.3.13 setEncryption() ................................ 904
  8.319.3.14 setGrayImageMaxResolution() ..................... 905
  8.319.3.15 setJPEGQuality() ................................ 905
  8.319.3.16 setMonoImageMaxResolution() ..................... 906
  8.319.3.17 setNeverEmbedFonts() ........................... 906
  8.319.3.18 setOutputIntent() ................................ 906
  8.319.3.19 setPdfXDeviceNErrorHandling() .................... 907
  8.319.3.20 setPdfXExtendedGraphicsStateErrorHandling() .... 907
  8.319.3.21 setPdfXOptionalContentErrorHandling .......... 907
  8.319.3.22 setPreferredColorImageCompression() ........... 907
  8.319.3.23 setPreferredGrayImageCompression() ............ 908
  8.319.3.24 setPreferredMonoImageCompression() ............ 908
  8.319.3.25 setPreferredRenderedImageCompression() ....... 908
8.323.2.3 afterLastByte() ........................................ 920
8.323.2.4 beforeEndPageSetup() .................................. 920
8.323.2.5 beforeEndSetup() ........................................ 920
8.323.2.6 beforeFirstByte() ....................................... 922
8.323.2.7 beforePsHeader() ....................................... 922
8.323.2.8 beforeShowpage() ........................................ 922

8.324 JawsMako::IPSOutput Class Reference ........................................ 923
8.324.1 Detailed Description ........................................ 924
8.324.2 Member Function Documentation ................................ 924
  8.324.2.1 setConvertAllObjectsToTargetColorSpace() .................. 924
  8.324.2.2 setStreamingOutput() ................................... 924
  8.324.2.3 setTargetColorSpace() ................................... 925
  8.324.2.4 setTargetProfile() ..................................... 925

8.325 IPushbackStream Class Reference ....................................... 925
8.325.1 Detailed Description ......................................... 926
8.325.2 Member Function Documentation ................................ 926
  8.325.2.1 pushBack() [1/2] ...................................... 926
  8.325.2.2 pushBack() [2/2] ...................................... 927

8.326 IRAInputPushbackStream Class Reference ............................... 927
8.326.1 Detailed Description ......................................... 928

8.327 IRAInputStream Class Reference ...................................... 928
8.327.1 Detailed Description ......................................... 929

8.328 IRAOutputStream Class Reference .................................... 929
8.328.1 Detailed Description ......................................... 930

8.329 IRAStream Class Reference ......................................... 930
8.329.1 Detailed Description ......................................... 931
8.329.2 Member Function Documentation ................................ 931
  8.329.2.1 length() .............................................. 931
  8.329.2.2 setPos() .............................................. 931

8.330 IRCObject Class Reference ......................................... 932
8.335.2 Member Function Documentation ......................... 947
  8.335.2.1 getFactory() ........................................... 947
  8.335.2.2 getLiteMessageHandler() ............................... 947
  8.335.2.3 getMessageHandler() .................................. 947
  8.335.2.4 getStartupDirectory() ................................ 947
  8.335.2.5 getTemporaryDirectory() ............................... 948
  8.335.2.6 getTempStore() [1/2] .................................. 948
  8.335.2.7 getTempStore() [2/2] .................................. 948
  8.335.2.8 setFactory() .......................................... 949
  8.335.2.9 setStartupDirectory() ................................ 949
  8.335.2.10 setTemporaryDirectory() .............................. 949

8.336 JawsMako::IShapeAnnotation Class Reference ...................... 950
  8.336.1 Detailed Description ...................................... 951
  8.336.2 Member Function Documentation ............................ 951
    8.336.2.1 getInteriorColor() .................................. 951
    8.336.2.2 getRectInset() ...................................... 951
    8.336.2.3 setInteriorColor() .................................. 951

8.337 JawsMako::ISkiaRenderer Class Reference ......................... 952
  8.337.1 Detailed Description ...................................... 952
  8.337.2 Member Function Documentation ............................ 952
    8.337.2.1 create() ............................................. 953
    8.337.2.2 drawNode() ......................................... 953
    8.337.2.3 flushCaches() ...................................... 953

8.338 JawsMako::ISoundAnnotation Class Reference ...................... 953
  8.338.1 Detailed Description ...................................... 954
  8.338.2 Member Function Documentation ............................ 954
    8.338.2.1 getSoundAsWav() ................................... 954

8.339 JawsMako::IStampAnnotation Class Reference ...................... 955
  8.339.1 Detailed Description ...................................... 956
  8.339.2 Member Function Documentation ............................ 956
8.352.2 Member Function Documentation ........................................ 980
  8.352.2.1 create() ...................................................... 980

8.353 JawsMako::ITextSelect Class Reference ........................................ 980
  8.353.1 Detailed Description ................................................. 981
  8.353.2 Member Function Documentation ........................................ 981
    8.353.2.1 create() ...................................................... 981

8.354 JawsMako::IThreads Class Reference .......................................... 982
  8.354.1 Detailed Description ................................................. 982

8.355 JawsMako::ITransform Class Reference ......................................... 983
  8.355.1 Detailed Description ................................................. 984
  8.355.2 Member Function Documentation ........................................ 984
    8.355.2.1 transform() [1/5] .............................................. 984
    8.355.2.2 transform() [2/5] .............................................. 985
    8.355.2.3 transform() [3/5] .............................................. 985
    8.355.2.4 transform() [4/5] .............................................. 986
    8.355.2.5 transform() [5/5] .............................................. 986
    8.355.2.6 transformPage() ................................................ 987

8.356 JawsMako::ITransformChain Class Reference ..................................... 987
  8.356.1 Detailed Description ................................................. 988
  8.356.2 Member Function Documentation ........................................ 988
    8.356.2.1 create() ...................................................... 988
    8.356.2.2 getTransforms() .............................................. 989
    8.356.2.3 pushTransform() .............................................. 989
    8.356.2.4 pushTransformFront() ........................................ 989
    8.356.2.5 removeTransform() ............................................ 990
    8.356.2.6 transform() [1/2] .............................................. 990
    8.356.2.7 transform() [2/2] .............................................. 990
    8.356.2.8 transformPage() ................................................ 991

8.357 JawsMako::IType3UnpackerTransform Class Reference .......................... 991
  8.357.1 Detailed Description ................................................. 992
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.357.2</td>
<td>Member Function Documentation</td>
<td>992</td>
</tr>
<tr>
<td>8.357.2.1</td>
<td><code>create()</code></td>
<td>992</td>
</tr>
<tr>
<td>8.358</td>
<td>JawsMako::IUnicodeHelper Class Reference</td>
<td>992</td>
</tr>
<tr>
<td>8.358.1</td>
<td>Detailed Description</td>
<td>993</td>
</tr>
<tr>
<td>8.359</td>
<td>JawsMako::IWidgetAnnotation Class Reference</td>
<td>994</td>
</tr>
<tr>
<td>8.359.1</td>
<td>Detailed Description</td>
<td>994</td>
</tr>
<tr>
<td>8.359.2</td>
<td>Member Function Documentation</td>
<td>994</td>
</tr>
<tr>
<td>8.359.2.1</td>
<td><code>getFieldFlags()</code></td>
<td>995</td>
</tr>
<tr>
<td>8.359.2.2</td>
<td><code>getFieldType()</code></td>
<td>996</td>
</tr>
<tr>
<td>8.359.2.3</td>
<td><code>getPartialName()</code></td>
<td>996</td>
</tr>
<tr>
<td>8.359.2.4</td>
<td><code>setPartialName()</code></td>
<td>996</td>
</tr>
<tr>
<td>8.360</td>
<td>JawsMako::IXAMLGenerator Class Reference</td>
<td>997</td>
</tr>
<tr>
<td>8.360.1</td>
<td>Detailed Description</td>
<td>999</td>
</tr>
<tr>
<td>8.360.2</td>
<td>Member Function Documentation</td>
<td>999</td>
</tr>
<tr>
<td>8.360.2.1</td>
<td><code>create()</code></td>
<td>999</td>
</tr>
<tr>
<td>8.360.2.2</td>
<td><code>generateXAML() [1/2]</code></td>
<td>1000</td>
</tr>
<tr>
<td>8.360.2.3</td>
<td><code>generateXAML() [2/2]</code></td>
<td>1000</td>
</tr>
<tr>
<td>8.360.2.4</td>
<td><code>generateXAMLForAppearance() [1/2]</code></td>
<td>1000</td>
</tr>
<tr>
<td>8.360.2.5</td>
<td><code>generateXAMLForAppearance() [2/2]</code></td>
<td>1001</td>
</tr>
<tr>
<td>8.360.2.6</td>
<td><code>generateXAMLForPageAndAnnotationAppearances()</code></td>
<td>1001</td>
</tr>
<tr>
<td>8.360.2.7</td>
<td><code>generateXAMLForPageAnnotationAppearances()</code></td>
<td>1002</td>
</tr>
<tr>
<td>8.360.2.8</td>
<td><code>getResource()</code></td>
<td>1002</td>
</tr>
<tr>
<td>8.360.2.9</td>
<td><code>getResources()</code></td>
<td>1002</td>
</tr>
<tr>
<td>8.360.2.10</td>
<td><code>setColorImageMaxResolution()</code></td>
<td>1003</td>
</tr>
<tr>
<td>8.360.2.11</td>
<td><code>setGrayImageMaxResolution()</code></td>
<td>1003</td>
</tr>
<tr>
<td>8.360.2.12</td>
<td><code>setJPEGQuality()</code></td>
<td>1004</td>
</tr>
<tr>
<td>8.360.2.13</td>
<td><code>setMergeFonts()</code></td>
<td>1004</td>
</tr>
<tr>
<td>8.360.2.14</td>
<td><code>setMonoImageMaxResolution()</code></td>
<td>1004</td>
</tr>
<tr>
<td>8.360.2.15</td>
<td><code>setPreferredColorImageFormat()</code></td>
<td>1005</td>
</tr>
<tr>
<td>8.360.2.16</td>
<td><code>setPreferredGrayImageFormat()</code></td>
<td>1005</td>
</tr>
</tbody>
</table>
8.360.2.17 setPreferredMonoImageFormat() ........................................ 1005
8.360.2.18 setRenderResolution() .................................................. 1005
8.360.2.19 setSubsetFonts() ....................................................... 1006
8.360.2.20 setTargetColorSpace() ............................................... 1006
8.360.2.21 setTargetProfile() ..................................................... 1006
8.361 JawsMako::IXPSInput Class Reference ........................................ 1007
  8.361.1 Detailed Description ..................................................... 1007
  8.361.2 Member Function Documentation ...................................... 1007
    8.361.2.1 create() ............................................................. 1008
    8.361.2.2 openStreaming() .................................................. 1008
8.362 JawsMako::IXPSOutput Class Reference ....................................... 1009
  8.362.1 Detailed Description ................................................... 1010
  8.362.2 Member Function Documentation ...................................... 1010
    8.362.2.1 setColorImageMaxResolution() .................................. 1011
    8.362.2.2 setGrayImageMaxResolution() .................................. 1011
    8.362.2.3 setJPEGQuality() ................................................ 1011
    8.362.2.4 setMergeFonts() .................................................. 1013
    8.362.2.5 setMonoImageMaxResolution() .................................... 1013
    8.362.2.6 setPreferredColorImageFormat() .................................. 1013
    8.362.2.7 setPreferredGrayImageFormat() .................................. 1014
    8.362.2.8 setPreferredMonoImageFormat() .................................. 1014
    8.362.2.9 setRenderResolution() .......................................... 1014
    8.362.2.10 setSubsetFonts() ................................................ 1015
    8.362.2.11 setTargetColorSpace() ......................................... 1015
    8.362.2.12 setTargetProfile() ............................................. 1015
8.363 IDOMPDFImage::JBIG2Params Class Reference .................................. 1016
  8.363.1 Detailed Description ................................................... 1016
8.364 PValue Class Reference .......................................................... 1016
  8.364.1 Detailed Description ................................................... 1016
8.365 SignatureID Class Reference .................................................. 1017
  8.365.1 Detailed Description ................................................... 1017
9 File Documentation

9.1 edlblackpointcompensation.h File Reference

9.1.1 Detailed Description

9.1.2 Enumeration Type Documentation

9.1.2.1 eBlackPointCompensation

9.2 edlblend.h File Reference

9.2.1 Detailed Description

9.3 edlerrors.cpp File Reference

9.3.1 Detailed Description

9.3.2 Function Documentation

9.3.2.1 getEDLErrorString()

9.4 edlerrors.h File Reference

9.4.1 Detailed Description

9.4.2 Function Documentation

9.4.2.1 getEDLErrorString()

9.4.2.2 throwEDLError()

9.5 edlfwd.h File Reference

9.5.1 Detailed Description

9.6 edlgeom.h File Reference

9.6.1 Detailed Description

9.7 edlmath.h File Reference

9.7.1 Detailed Description

9.7.2 Macro Definition Documentation

9.7.2.1 PI

9.8 edlnamespaces.h File Reference

9.8.1 Detailed Description

9.9 edlproperty.h File Reference

9.9.1 Detailed Description

9.10 edlqname.h File Reference

9.10.1 Detailed Description
<table>
<thead>
<tr>
<th>Section</th>
<th>File Reference</th>
<th>Detailed Description</th>
<th>Type Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.11</td>
<td>edlquartz.h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.12</td>
<td>edlrenderingintent.h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.13</td>
<td>edlstream.h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.14</td>
<td>edlstring.h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.15</td>
<td>edltime.h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.16</td>
<td>edtypes.h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.17</td>
<td>edlvector.h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.18</td>
<td>edlversion.h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Section</td>
<td>File Reference</td>
<td>Detailed Description</td>
<td>File Reference</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------</td>
<td>----------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>9.30</td>
<td>idomgroup.h</td>
<td>1050</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9.30.1 Detailed Description</td>
<td>1050</td>
<td></td>
</tr>
<tr>
<td>9.31</td>
<td>idomhashable.h</td>
<td>1051</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9.31.1 Detailed Description</td>
<td>1051</td>
<td></td>
</tr>
<tr>
<td>9.32</td>
<td>idomid.h</td>
<td>1051</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9.32.1 Detailed Description</td>
<td>1051</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9.32.2 Function Documentation</td>
<td>1051</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9.32.2.1 allocateNewDOMid()</td>
<td>1052</td>
<td></td>
</tr>
<tr>
<td>9.33</td>
<td>idomimageresource.h</td>
<td>1054</td>
<td></td>
</tr>
<tr>
<td>9.34</td>
<td>idomjobtk.h</td>
<td>1055</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9.34.1 Detailed Description</td>
<td>1056</td>
<td></td>
</tr>
<tr>
<td>9.35</td>
<td>idomjobtk.h</td>
<td>1056</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9.35.2 Function Documentation</td>
<td>1056</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9.35.2.1 eDOMJobTkLevel</td>
<td>1056</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9.35.3 Function Documentation</td>
<td>1056</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9.35.3.1 cloneJobTkContentTree()</td>
<td>1056</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9.35.3.2 cloneJobTkTree()</td>
<td>1057</td>
<td></td>
</tr>
<tr>
<td>9.36</td>
<td>idommetadata.h</td>
<td>1057</td>
<td></td>
</tr>
<tr>
<td>9.37</td>
<td>idomnode.h</td>
<td>1057</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9.37.1 Detailed Description</td>
<td>1058</td>
<td></td>
</tr>
<tr>
<td>9.38</td>
<td>idomopi.h</td>
<td>1060</td>
<td></td>
</tr>
<tr>
<td>9.39</td>
<td>idomoutline.h</td>
<td>1061</td>
<td></td>
</tr>
<tr>
<td>9.40</td>
<td>idompage.h</td>
<td>1061</td>
<td></td>
</tr>
</tbody>
</table>
9.40 Detailed Description ............................ 1062

9.41 idompath.h File Reference .................. 1062
  9.41.1 Detailed Description ...................... 1062

9.42 idompathgeometry.h File Reference ......... 1062
  9.42.1 Detailed Description ...................... 1063

9.43 idomresources.h File Reference ............. 1063
  9.43.1 Detailed Description ...................... 1064

9.44 idomsecurity.h File Reference ............... 1064
  9.44.1 Typedef Documentation ................... 1065
    9.44.1.1 GetPasswordFunc ...................... 1065

9.45 idomshape.h File Reference .................. 1065
  9.45.1 Detailed Description ...................... 1066

9.46 idomtarget.h File Reference .................. 1066
  9.46.1 Detailed Description ...................... 1066

9.47 iedlcollection.h File Reference ............. 1066
  9.47.1 Detailed Description ...................... 1067

9.48 iedlenum.h File Reference .................... 1067
  9.48.1 Detailed Description ...................... 1067

9.49 iedlfactory.h File Reference .................. 1067
  9.49.1 Detailed Description ...................... 1067

9.50 iedlobject.h File Reference .................. 1068
  9.50.1 Detailed Description ...................... 1068
    9.50.2 Function Documentation .................. 1068
      9.50.2.1 clone() ............................ 1068

9.51 iedltempstore.h File Reference .............. 1069
  9.51.1 Detailed Description ...................... 1069

9.52 iedltree.h File Reference .................... 1069
  9.52.1 Detailed Description ...................... 1069

9.53 ifilespec.h File Reference ................... 1069
  9.53.1 Detailed Description ...................... 1070
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.66 1 Detailed Description</td>
<td></td>
<td>1079</td>
</tr>
<tr>
<td>9.67 pdfinput.h File Reference</td>
<td></td>
<td>1080</td>
</tr>
<tr>
<td>9.67.1 Detailed Description</td>
<td></td>
<td>1080</td>
</tr>
<tr>
<td>9.68 pdfoutput.h File Reference</td>
<td></td>
<td>1080</td>
</tr>
<tr>
<td>9.68.1 Detailed Description</td>
<td></td>
<td>1080</td>
</tr>
<tr>
<td>9.69 pj.h File Reference</td>
<td></td>
<td>1080</td>
</tr>
<tr>
<td>9.69.1 Detailed Description</td>
<td></td>
<td>1081</td>
</tr>
<tr>
<td>9.70 platform.h File Reference</td>
<td></td>
<td>1081</td>
</tr>
<tr>
<td>9.70.1 Detailed Description</td>
<td></td>
<td>1081</td>
</tr>
<tr>
<td>9.71 platform_utils.h File Reference</td>
<td></td>
<td>1081</td>
</tr>
<tr>
<td>9.71.1 Detailed Description</td>
<td></td>
<td>1082</td>
</tr>
<tr>
<td>9.71.2 Function Documentation</td>
<td></td>
<td>1082</td>
</tr>
<tr>
<td>9.71.2.1 edlExclusiveMakeTempDir()</td>
<td></td>
<td>1082</td>
</tr>
<tr>
<td>9.71.2.2 edlFopen()</td>
<td></td>
<td>1082</td>
</tr>
<tr>
<td>9.71.2.3 edlGetProcessId()</td>
<td></td>
<td>1084</td>
</tr>
<tr>
<td>9.71.2.4 edlMakeTempDir()</td>
<td></td>
<td>1084</td>
</tr>
<tr>
<td>9.71.2.5 edlMakeTempDirProvidingSubDirPath()</td>
<td></td>
<td>1085</td>
</tr>
<tr>
<td>9.71.2.6 edlMkdir()</td>
<td></td>
<td>1085</td>
</tr>
<tr>
<td>9.71.2.7 edlRmdir()</td>
<td></td>
<td>1085</td>
</tr>
<tr>
<td>9.71.2.8 edlSnprintf()</td>
<td></td>
<td>1086</td>
</tr>
<tr>
<td>9.71.2.9 edlSnprintfE()</td>
<td></td>
<td>1086</td>
</tr>
<tr>
<td>9.71.2.10 edlVsnprintf()</td>
<td></td>
<td>1087</td>
</tr>
<tr>
<td>9.72 psoutput.h File Reference</td>
<td></td>
<td>1087</td>
</tr>
<tr>
<td>9.72.1 Detailed Description</td>
<td></td>
<td>1087</td>
</tr>
<tr>
<td>9.73 skiarenderer.h File Reference</td>
<td></td>
<td>1088</td>
</tr>
<tr>
<td>9.73.1 Detailed Description</td>
<td></td>
<td>1088</td>
</tr>
<tr>
<td>9.74 smartptr.h File Reference</td>
<td></td>
<td>1088</td>
</tr>
<tr>
<td>9.74.1 Detailed Description</td>
<td></td>
<td>1088</td>
</tr>
<tr>
<td>9.75 structure.h File Reference</td>
<td></td>
<td>1088</td>
</tr>
<tr>
<td>9.75.1 Detailed Description</td>
<td></td>
<td>1089</td>
</tr>
</tbody>
</table>
Chapter 1

Main Page

1.1 Introduction

This document describes the core API of the Global Graphics Mako™ Library (formerly known as gDoc Core). Mako provides the following key capabilities:

- A new, simplified Mako API that provides:
  - Easy access to the contents of PDF, XPS 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 PCLXL 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/X™ using Quartz 2D
  - An XPS-compatible renderer for Android™ using the Skia™ library
  - A renderer based on Jaws™ for rendering individual DOM nodes, subtrees, or entire pages to an image or an RGB frame buffer

For conversion, this release supports the following pathways:

- PDF to PDF, XPS, PCL 5c, PCL 5e, PCL/XL or PostScript
• XPS to XPS, PDF, PCL 5c, PCL 5e, PCL/XL or PostScript
• PCL5 to PDF, XPS, PCL 5c, PCL 5e, PCL/XL or PostScript
• PCL/XL to PDF, XPS, PCL 5c, PCL 5e, PCL/XL or PostScript

A conversion path for PostScript to PDF is also provided as part of the Mako SDK (Jaws PDF Library)

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).
2. The C++ header files describing the published Mako API (./interface/jawsmako and ./interface/edl).
3. A collection of sample applications using Mako including:
   • Mako Converter (./makoapps/makoconverter) A simple example showing how to convert from PDF or XPS to XPS using the Mako APIs (Windows and MacOS).
   • Simple Examples (./makoapps/simpleexamples) A sample application demonstrating various (but not exhaustive) uses of the Mako APIs (Windows and MacOS). This selection will grow in subsequent releases.
   • iOS XPS Viewer (./makoapps/iOSXPSViewer) for iOS that provides a simple XPS viewer using the Mako APIs.
   • Android XPS Viewer (./makoapps/AndroidXPSViewer) for Android that provides a simple XPS viewer using the Mako APIs.
   • A sample XPSDrv implementation (./makoapps/xpsdrv) supporting five streaming output formats (PS, PDF, PCLXL, PCL5e and PCL5c) selectable via individual .inf installation files. This driver encapsulates Mako as an XPSDrv filter with streaming input and streaming output.
4. HTML documentation (this document) describing the public interfaces (./html).

1.3 How to install this release

The release is provided in .zip file format. Unzip to a convenient location. Mako libraries for the supported platforms are found in these folders:

• Android
• iOS
• Linux (for Debian-based distributions, eg Ubuntu, Mint)
• Linux (for Debian Jessie)
• Linux (for musl distributions, eg Alpine Linux)
• macOS
• Windows (static and non-static libs, x86 & x64)
• Windows UWP (Store apps, Windows 10 IoT)

The sample applications (makoapps folder) have Visual Studio Solutions and Project files (Windows), Makefiles (macOS), JAM scripts (Linux) XCode projects (iOS) and Eclipse projects (Android). The XPSDrv has batch files to build driver packages.
Chapter 2

Copyright Notices

LZ4 Library

LZ4 Library Copyright (c) 2011-2016, Yann Collet All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

• Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

• Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution:

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Little CMS

Little CMS Copyright © 1998-2006 Marti Maria

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
Libtiff

Copyright © 1988-1997 Sam Leffler


THE SOFTWARE IS PROVIDED “AS-IS” AND WITHOUT WARRANTY OF ANY KIND EXPRESS, IMPLIED OR OTHERWISE, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL SAM LEFFLER OR SILICON GRAPHICS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES OF ANY KIND, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER OR NOT ADVISED OF THE POSSIBILITY OF DAMAGE, AND ON ANY THEORY OF LIABILITY, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

b64

Copyright (c) 2004-2009, Matthew Wilson and Synesis Software All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name(s) of Matthew Wilson and Synesis Software nor the names of any contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

zlib

Copyright (C) 1995-2005 Jean-loup Gailly and Mark Adler

RSA

“RSA® BSAFE® is a registered trademark of RSA Security Inc in the US and/or other countries. RSA Secured and the RSA Secured logo are trademarks of RSA Security Inc in the US and/or other countries.”

Fonts

“Fonts copyright © 2000-2004 Timo Lehtinen. All Rights Reserved.”
Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

API ......................................................... 37
Chapter 4

Hierarchical Index

4.1 Class Hierarchy

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

JawsMako::CAnnotationBorder .................................................. 45
JawsMako::IXAMLGenerator::CAnnotationXAML .............................. 47
CClassID ................................................................. 48
CClassParams ............................................................. 51
IDOMArcSegment::Data ..................................................... 104
IDOMAudioFile::Data ....................................................... 111
IDOMCanvas::Data ......................................................... 107
IDOMCharPathGroup::Data .................................................. 108
IDOMColorSpaceDeviceN::Data ............................................ 117
IDOMColorSpaceICCBased::Data ......................................... 114
IDOMColorSpaceIndexed::Data ............................................ 116
IDOMColorSpaceLAB::Data ............................................... 118
IDOMDePremultiplierFilter::Data ........................................ 87
IDOMDeviceNColorant::Data ............................................... 115
IDOMExponentialFunction::Data ......................................... 131
IDOMFixedPage::Data ..................................................... 102
IDOMFont::Data .......................................................... 122
IDOMFontOpenType::Data .................................................. 123
  IDOMFontOpenTypeTT::Data .............................................. 124
  IDOMFontPCL5::Data ..................................................... 127
  IDOMFontPCLXL::Data ................................................... 126
IDOMFontSource::Data ................................................... 118
  IDOMFontSourceObfuscationConverter::Data .......................... 121
  IDOMFontSourceStreamFilter::Data .................................. 119
    IDOMFontSourceFromStream::Data .................................. 120
IDOMForm::Data .......................................................... 129
IDOMFormInstance::Data .................................................. 130
IDOMGlyph::Data ........................................................ 134
IDOMGlyphs::Data ......................................................... 134
IDOMGradientStop::Data .................................................. 80
IDOMGroup::Data ........................................................ 135
IDOMGroupingFunction::Data ............................................. 132
IDOMICCProfile::Data .................................................... 110
IDOMImage::Data .......................................................... 73
IDOMCompositeImage::Data ................................................ 78
### 4.1 Class Hierarchy

<table>
<thead>
<tr>
<th>Class</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDOMImageBrush</td>
<td>403</td>
</tr>
<tr>
<td>IDOMShadingPatternBrush</td>
<td>628</td>
</tr>
<tr>
<td>IDOMShadingPatternType1Brush</td>
<td>633</td>
</tr>
<tr>
<td>IDOMShadingPatternType2Brush</td>
<td>637</td>
</tr>
<tr>
<td>IDOMShadingPatternType3Brush</td>
<td>643</td>
</tr>
<tr>
<td>IDOMShadingPatternType4567Brush</td>
<td>648</td>
</tr>
<tr>
<td>IDOMSoftMaskBrush</td>
<td>663</td>
</tr>
<tr>
<td>IDOMTilingPatternBrush</td>
<td>681</td>
</tr>
<tr>
<td>IDOMVisualBrush</td>
<td>704</td>
</tr>
<tr>
<td>IDOMCatalog</td>
<td>214</td>
</tr>
<tr>
<td>IDOMColor</td>
<td>227</td>
</tr>
<tr>
<td>IDOMColorSpace</td>
<td>233</td>
</tr>
<tr>
<td>IDOMDeviceNColorant</td>
<td>268</td>
</tr>
<tr>
<td>IDOMFontSource</td>
<td>314</td>
</tr>
<tr>
<td>IDOMFunction</td>
<td>339</td>
</tr>
<tr>
<td>IDOMGlyphIDEnumerator</td>
<td>351</td>
</tr>
<tr>
<td>IDOMGradientStop</td>
<td>381</td>
</tr>
<tr>
<td>IDOMImageProperties</td>
<td>427</td>
</tr>
<tr>
<td>IDOMJobTk</td>
<td>433</td>
</tr>
<tr>
<td>IDOMMatrix</td>
<td>475</td>
</tr>
<tr>
<td>IDOMMetadata</td>
<td>478</td>
</tr>
<tr>
<td>IDOMNode</td>
<td>483</td>
</tr>
<tr>
<td>IDOMForm</td>
<td>328</td>
</tr>
<tr>
<td>IDOMFormInstance</td>
<td>332</td>
</tr>
<tr>
<td>IDOMGlyph</td>
<td>343</td>
</tr>
<tr>
<td>IDOMGlyphs</td>
<td>354</td>
</tr>
<tr>
<td>IDOMGroup</td>
<td>384</td>
</tr>
<tr>
<td>IDOMCharPathGroup</td>
<td>221</td>
</tr>
<tr>
<td>IDOMTransparencyGroup</td>
<td>692</td>
</tr>
<tr>
<td>IDOMCanvas</td>
<td>207</td>
</tr>
<tr>
<td>IDOMJobTkContent</td>
<td>438</td>
</tr>
<tr>
<td>IDOMJobTkGenericCharacterData</td>
<td>449</td>
</tr>
<tr>
<td>IDOMJobTkGenericNode</td>
<td>451</td>
</tr>
<tr>
<td>IDOMJobTkNode</td>
<td>455</td>
</tr>
<tr>
<td>IDOMJobTkOwner</td>
<td>460</td>
</tr>
<tr>
<td>IDOMFixedPage</td>
<td>282</td>
</tr>
<tr>
<td>IDOMJob Tk Value</td>
<td>463</td>
</tr>
<tr>
<td>IDOMPage</td>
<td>528</td>
</tr>
<tr>
<td>IDOMPathNode</td>
<td>551</td>
</tr>
<tr>
<td>IDOMVisualRoot</td>
<td>712</td>
</tr>
<tr>
<td>IDOMOPI</td>
<td>503</td>
</tr>
<tr>
<td>IDOMOPI13</td>
<td>505</td>
</tr>
<tr>
<td>IDOMOPI20</td>
<td>515</td>
</tr>
<tr>
<td>IDOMOutline</td>
<td>517</td>
</tr>
<tr>
<td>IDOMOutlineEntry</td>
<td>520</td>
</tr>
<tr>
<td>IDOMPathFigure</td>
<td>538</td>
</tr>
<tr>
<td>IDOMPathGeometry</td>
<td>544</td>
</tr>
<tr>
<td>IDOMPathSegment</td>
<td>579</td>
</tr>
<tr>
<td>IDOMArcSegment</td>
<td>196</td>
</tr>
<tr>
<td>IDOMPolyBezierSegment</td>
<td>591</td>
</tr>
<tr>
<td>IDOMPolyLineSegment</td>
<td>594</td>
</tr>
<tr>
<td>IDOMPolyQuadraticBezierSegment</td>
<td>596</td>
</tr>
<tr>
<td>IDOMResource</td>
<td>616</td>
</tr>
<tr>
<td>IDOMAudioFile</td>
<td>203</td>
</tr>
<tr>
<td>IDOMICCProfile</td>
<td>394</td>
</tr>
<tr>
<td>IDOMImage</td>
<td>397</td>
</tr>
<tr>
<td>IDOMImageProperties</td>
<td>602</td>
</tr>
<tr>
<td>IDOMPrintTicket</td>
<td>609</td>
</tr>
<tr>
<td>IDOMRawDataFile</td>
<td>609</td>
</tr>
<tr>
<td>SignatureID</td>
<td>Page</td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td></td>
<td>1017</td>
</tr>
</tbody>
</table>
Chapter 5

Class Index

5.1 Class List

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

- **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. 45

- **JawsMako::IXAMLGenerator::CAnnotationXAML**
  Class for receiving XAML generated for annotation appearances in a bulk fashion. 47

- **IDOMPDFImage::CCTITTFaxParams**
  Class to hold filter parameters for CCITTFax-compressed image data. Please see the PDF specification for the meaning of these parameters. 48

- **CClassID**
  An object to represent a 128-bit globally unique ID. 48

- **CClassParams**
  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. 51

- **JawsMako::IOptionalContentConfiguration::COrderEntry**
  Class for presenting the order that groups should be displayed in a user interface. May be arranged in a tree. 51

- **JawsMako::IPDFInput::CPdfFontInfo**
  Information about a font in a PDF file, obtained by scanning the PDF font structures. 52

- **JawsMako::IPDFInput::CPdfScannedInk**
  Basic information about an ink used in a PDF file, obtained by scanning the PDF page tree. 53

- **JawsMako::IPJLParser::CPjlAttributeValue**
  A captured PJL attribute. 53

- **JawsMako::CQuadPoint**
  A representation of a PDF Quadpoint, in DOM coordinates. 54

- **JawsMako::CRectInset**
  A class which specifies an inset from a rectangle. 54

- **JawsMako::IJawsRenderer::CSpotHalftone**
  Description of a simple spot halftone, at 45 degrees, using Jaws's default spot function. 55

- **JawsMako::CTemporaryStoreParameters**
  Allows the temporary storage parameters to be optionally overridden. 56
JawsMako::IJawsRenderer::CThresholdArrayHalftone
Description of a Type 3 8-bit threshold array halftone for use with renderMonochrome. Please refer to section 7.4.5 of the PostScript language reference manual, 3rd edition

JawsMako::IJawsRenderer::CThresholdHalftone
A halftone representing a simple threshold

CTransformMatrix<TItem>
Matrix class - special 3x2 matrix

JawsMako::CTransformState
Class for tracking the graphics state leading to the point where a transform is applied

JawsMako::IAnnotationUtils::CXMLResource
Simple class for tracking streams associated with XML generated by generateXMLForDocument()
<table>
<thead>
<tr>
<th>Class List</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDOMImageMatteRemoverFilter::Data</td>
<td>88</td>
</tr>
<tr>
<td>Initialization data</td>
<td></td>
</tr>
<tr>
<td>IDOMImageBitScalerFilter::Data</td>
<td>89</td>
</tr>
<tr>
<td>Initialization data</td>
<td></td>
</tr>
<tr>
<td>IDOMImageBrush::Data</td>
<td>89</td>
</tr>
<tr>
<td>Initialization data</td>
<td></td>
</tr>
<tr>
<td>IDOMImageColorKeyFilter::Data</td>
<td>90</td>
</tr>
<tr>
<td>Initialization data</td>
<td></td>
</tr>
<tr>
<td>IDOMImageDecodeFilter::Data</td>
<td>91</td>
</tr>
<tr>
<td>Initialization data</td>
<td></td>
</tr>
<tr>
<td>IDOMMaskedBrush::Data</td>
<td>91</td>
</tr>
<tr>
<td>Initialization data</td>
<td></td>
</tr>
<tr>
<td>IDOMFilteredImage::Data</td>
<td>91</td>
</tr>
<tr>
<td>Initialization data</td>
<td></td>
</tr>
<tr>
<td>IDOMJobTkGenericNode::Data</td>
<td>92</td>
</tr>
<tr>
<td>Initialization data</td>
<td></td>
</tr>
<tr>
<td>IDOMVisualBrush::Data</td>
<td>93</td>
</tr>
<tr>
<td>Initialization data</td>
<td></td>
</tr>
<tr>
<td>IDOMJobTkGenericCharacterData::Data</td>
<td>93</td>
</tr>
<tr>
<td>Initialization data</td>
<td></td>
</tr>
<tr>
<td>IDOMJobTkNode::Data</td>
<td>94</td>
</tr>
<tr>
<td>Initialization data</td>
<td></td>
</tr>
<tr>
<td>IDOMSoftMaskBrush::Data</td>
<td>95</td>
</tr>
<tr>
<td>Initialization data</td>
<td></td>
</tr>
<tr>
<td>IDOMJobTkValue::Data</td>
<td>95</td>
</tr>
<tr>
<td>Initialization data</td>
<td></td>
</tr>
<tr>
<td>IDOMJobTkContent::Data</td>
<td>96</td>
</tr>
<tr>
<td>Initialization data</td>
<td></td>
</tr>
<tr>
<td>IDOMTilingPatternBrush::Data</td>
<td>97</td>
</tr>
<tr>
<td>Initialization data</td>
<td></td>
</tr>
<tr>
<td>IDOMShadingPatternType1Brush::Data</td>
<td>97</td>
</tr>
<tr>
<td>Initialization data</td>
<td></td>
</tr>
<tr>
<td>IDOMOutlineEntry::Data</td>
<td>98</td>
</tr>
<tr>
<td>Initialization data</td>
<td></td>
</tr>
<tr>
<td>IDOMShadingPatternType2Brush::Data</td>
<td>99</td>
</tr>
<tr>
<td>Initialization data</td>
<td></td>
</tr>
<tr>
<td>IDOMShadingPatternType3Brush::Data</td>
<td>100</td>
</tr>
<tr>
<td>Initialization data</td>
<td></td>
</tr>
<tr>
<td>IDOMOutline::Data</td>
<td>100</td>
</tr>
<tr>
<td>Initialization data</td>
<td></td>
</tr>
<tr>
<td>IDOMFixedPage::Data</td>
<td>101</td>
</tr>
<tr>
<td>Initialization data</td>
<td></td>
</tr>
<tr>
<td>IDOMPathNode::Data</td>
<td>102</td>
</tr>
<tr>
<td>Initialization data</td>
<td></td>
</tr>
<tr>
<td>IDOMShadingPatternType4567Brush::Data</td>
<td>102</td>
</tr>
<tr>
<td>Initialization data</td>
<td></td>
</tr>
<tr>
<td>IDOMArcSegment::Data</td>
<td>103</td>
</tr>
<tr>
<td>Initialization data</td>
<td></td>
</tr>
<tr>
<td>IDOMPolyLineSegment::Data</td>
<td>104</td>
</tr>
<tr>
<td>Initialization data</td>
<td></td>
</tr>
<tr>
<td>IDOMNullBrush::Data</td>
<td>104</td>
</tr>
<tr>
<td>Initialization data</td>
<td></td>
</tr>
<tr>
<td>IDOMPolyBezierSegment::Data</td>
<td>105</td>
</tr>
<tr>
<td>Initialization data</td>
<td></td>
</tr>
<tr>
<td>IDOMPolyQuadraticBezierSegment::Data</td>
<td>106</td>
</tr>
<tr>
<td>Initialization data</td>
<td></td>
</tr>
<tr>
<td>IDOMCanvas::Data</td>
<td>106</td>
</tr>
<tr>
<td>Initialization data</td>
<td></td>
</tr>
<tr>
<td>Class Index</td>
<td>Page</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>IDOMCharPathGroup::Data</td>
<td>108</td>
</tr>
<tr>
<td>IDOMPathFigure::Data</td>
<td>108</td>
</tr>
<tr>
<td>IDOMPathGeometry::Data</td>
<td>108</td>
</tr>
<tr>
<td>IDOMICCProfile::Data</td>
<td>109</td>
</tr>
<tr>
<td>IDOMPrintTicket::Data</td>
<td>110</td>
</tr>
<tr>
<td>IDOMAudioFile::Data</td>
<td>110</td>
</tr>
<tr>
<td>IDOMRawDataFile::Data</td>
<td>111</td>
</tr>
<tr>
<td>IDOMResourceDictionary::Data</td>
<td>112</td>
</tr>
<tr>
<td>IDOMMatrix::Data</td>
<td>112</td>
</tr>
<tr>
<td>IDOMColorSpaceICCBased::Data</td>
<td>113</td>
</tr>
<tr>
<td>IDOMShape::Data</td>
<td>114</td>
</tr>
<tr>
<td>IDOMDeviceNColorant::Data</td>
<td>114</td>
</tr>
<tr>
<td>IEDLTmpStore::Data</td>
<td>115</td>
</tr>
<tr>
<td>IDOMColorSpaceIndexed::Data</td>
<td>116</td>
</tr>
<tr>
<td>IDOMColorSpaceDeviceN::Data</td>
<td>116</td>
</tr>
<tr>
<td>IDOMColorSpaceLAB::Data</td>
<td>117</td>
</tr>
<tr>
<td>IDOMFontSource::Data</td>
<td>118</td>
</tr>
<tr>
<td>IDOMFontSourceStreamFilter::Data</td>
<td>118</td>
</tr>
<tr>
<td>IDOMFontSourceFromStream::Data</td>
<td>119</td>
</tr>
<tr>
<td>IDOMFontSourceObfuscationConverter::Data</td>
<td>120</td>
</tr>
<tr>
<td>IDOMFont::Data</td>
<td>121</td>
</tr>
<tr>
<td>IDOMFontOpenType::Data</td>
<td>122</td>
</tr>
<tr>
<td>IDOMFontOpenTypeTT::Data</td>
<td>123</td>
</tr>
<tr>
<td>IDOMType3Font::Data</td>
<td>124</td>
</tr>
<tr>
<td>IDOMFontPCLXL::Data</td>
<td>125</td>
</tr>
<tr>
<td>IDOMFontPCL5::Data</td>
<td>126</td>
</tr>
<tr>
<td>IEDLNamespace::Data</td>
<td>127</td>
</tr>
<tr>
<td>IDOMForm::Data</td>
<td>128</td>
</tr>
<tr>
<td>IDOMFormInstance::Data</td>
<td>129</td>
</tr>
</tbody>
</table>

Generated by Doxygen
5.1 Class List

IDOMSampledFunction::Data
Initialization data .................................. 130

IDOMExponentialFunction::Data
Initialization data .................................. 131

IDOMStitchingFunction::Data
Initialization data .................................. 132

IDOMGroupingFunction::Data
Initialization data .................................. 132

IDOMPPostScriptCalculatorFunction::Data
Initialization data .................................. 133

IDOMGlyph::Data
Initialization data .................................. 134

IDOMGlyphs::Data
Initialization data .................................. 134

IDOMGroup::Data
Initialization data .................................. 135

IDOMPDFImage::DCTParams
Class to hold filter parameters for DCT-compressed image data. Please see the PDF specification for the meaning of these parameters .................................. 136

EDLIFStream
An ifstream that can deal with UTF8 file names on all platforms .................................. 136

EDLOFStream
An ofstream that can deal with UTF8 file names on all platforms .................................. 137

EDLQName
Implementation of qualified name class .................................. 137

IDOMPDFImage::FlateLZWParams
Class to hold filter parameters for Flate or LZW-compressed image data. Please see the PDF specification for the meaning of these parameters .................................. 141

JawsMako::IAnnotation
An interface class for an annotation. It is intended that future releases of JawsMako will extend this interface .................................. 142

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 .................................. 152

JawsMako::IAnnotationReference
A generic reference to an annotation. The target annotation might not be loaded. Chiefly used to refer to annotations from a Form .................................. 155

JawsMako::IAnnotationUtils .................................. 156

JawsMako::ICaretAnnotation
A generic interface class for a caret annotation. It is intended that future releases of JawsMako will extend this interface .................................. 157

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 .................................. 159

JawsMako::IColorConverterTransform
A transform for color conversion, converting all appropriate DOM contents to a desired target color space .................................. 161

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 .................................. 164

JawsMako::IColorSpaceSimplifierTransform
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 .................................. 176
<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDOMPDFImage::IDecodeParams</td>
<td>Abstract interface for per-image decoding filter parameters</td>
<td>178</td>
</tr>
<tr>
<td>JawsMako::IDocument</td>
<td>A document from an IDocumentAssembly, allowing for high level document and page management, and providing on-demand lazy loading of page markup</td>
<td>179</td>
</tr>
<tr>
<td>JawsMako::IDocumentAssembly</td>
<td>A self contained collection of IDocuments</td>
<td>185</td>
</tr>
<tr>
<td>IDOMActionArray</td>
<td>IDOMActionArray interface</td>
<td>190</td>
</tr>
<tr>
<td>IDOMActionLaunch</td>
<td>IDOMActionLaunch interface</td>
<td>192</td>
</tr>
<tr>
<td>IDOMArcSegment</td>
<td>Interface to Arc Segment element</td>
<td>196</td>
</tr>
<tr>
<td>IDOMAudioFile</td>
<td>IDOMAudioFile interface</td>
<td>203</td>
</tr>
<tr>
<td>IDOMBrush</td>
<td>Interface to the brush element</td>
<td>205</td>
</tr>
<tr>
<td>IDOMCanvas</td>
<td>A canvas is a special form of an isolated, non-knockout, normal blended transparency group</td>
<td>207</td>
</tr>
<tr>
<td>IDOMCatalog</td>
<td>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</td>
<td>214</td>
</tr>
<tr>
<td>IDOMCharPathGroup</td>
<td>IDOMCharPathGroup interface</td>
<td>221</td>
</tr>
<tr>
<td>IDOMColor</td>
<td>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</td>
<td>227</td>
</tr>
<tr>
<td>IDOMColorSpace</td>
<td>IDOMColorSpace interface</td>
<td>233</td>
</tr>
<tr>
<td>IDOMColorSpaceDeviceCMY</td>
<td>Represents the default CMY color space. NOTE: Currently for internal use only; Do not use this color space in your own applications</td>
<td>238</td>
</tr>
<tr>
<td>IDOMColorSpaceDeviceCMYK</td>
<td>Represents the default CMYK color space</td>
<td>239</td>
</tr>
<tr>
<td>IDOMColorSpaceDeviceGray</td>
<td>IDOMColorSpaceDeviceGray interface</td>
<td>241</td>
</tr>
<tr>
<td>IDOMColorSpaceDeviceN</td>
<td>This color space is analogous to the PostScript/PDF DeviceN/Separation color spaces</td>
<td>243</td>
</tr>
<tr>
<td>IDOMColorSpaceDeviceRGB</td>
<td>IDOMColorSpaceDeviceRGB interface</td>
<td>249</td>
</tr>
<tr>
<td>IDOMColorSpaceICCBased</td>
<td>Represents a color space described by an ICC profile</td>
<td>250</td>
</tr>
<tr>
<td>IDOMColorSpaceIndexed</td>
<td>This color space is analogous to the PostScript/PDF Indexed color space</td>
<td>253</td>
</tr>
<tr>
<td>IDOMColorSpaceLAB</td>
<td>This color space is as described in section 4.5.4 of the PDF 1.7 Reference Manual</td>
<td>257</td>
</tr>
<tr>
<td>IDOMColorSpacescRGB</td>
<td>Represents the scRGB color space</td>
<td>260</td>
</tr>
<tr>
<td>IDOMColorSpacesGray</td>
<td>Represents a gray color space using the sRGB gamma and WhitePoint</td>
<td>262</td>
</tr>
<tr>
<td>IDOMColorSpacesRGB</td>
<td>Represents the RGB color space</td>
<td>263</td>
</tr>
<tr>
<td>IDOMCompositImage</td>
<td>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</td>
<td>265</td>
</tr>
</tbody>
</table>
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 268

IDOMDeviceNColorant
This class enables the specification of colorant information for PDF style NChannel variants of DeviceN color spaces 268

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 271

IDOMExternalTarget
IDOMExternalTarget interface 275

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 277

IDOMFixedPage
Represents <FixedPage> element 282

IDOMFont
IDOMFont Base Class 294

IDOMFontOpenType
IDOMFontOpenType interface 297

IDOMFontOTFTrueType
Opentype Font 307

IDOMFontPCL5
IDOMFontPCL5 (PCL5 Truetype) derived from an OpenType font source 307

IDOMFontPCLXL
This class models PCL XL TrueType and bitmap fonts derived from an OpenType font source 311

IDOMFontSource
The font source for the class IDOMFont. This class describes the different ways fonts are constructed 314

IDOMFontSourceFromStream
The source for IDOMFont when sourced from an existing stream 317

IDOMFontSourceObfuscationConverter
Interface for a font sourced from a converter that performs obfuscation and deobfuscation 320

IDOMFontSourceStreamFilter
An abstract interface for fonts sourced from a font stream filter 323

IDOMForm
IDOMForm interface. 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 328

IDOMFormInstance
IDOMFormInstance interface. This describes an instance of an IDOMForm in a DOM tree 332

IDOMFunction
Base class for PDF/PS Style functions 339

IDOMGlyph
Abstract class modelling a single character from a font 343

IDOMGlyphIDEnumerator
DOM GlyphID Enumerator 351

IDOMGlyphName
354

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 354
IDOMGradientBrush
A common interface for both IDOMLinearGradient and IDOMRadialGradient. Provides straight-forward access to common attributes .................................................. 377

IDOMGradientStop
IDOMGradientStop defines the ramp of colors to use on a gradient ...................... 381

IDOMGroup
IDOMGroup interface .................................................. 384

IDOMGroupingFunction
Interface to encapsulate an array of x-input-1-output functions .......................... 389

IDOMHashable
Abstract interface for EDL objects that may be hashed .................................... 392

IDOMICCProfile
IDOMICCProfile interface ............................................. 394

IDOMImage
The base class describing an image. This class is subclassed to create a number of more specific image types .......................................................... 397

IDOMImageBitScalerFilter
An image filter that presents an image as an image with a different bits per sample ...... 400

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 ............................... 401

IDOMImageBrush
Provides an interface to a DOM image brush object .......................................... 403

IDOMImageChannelSelectorFilter
An image filter that presents optionally an image stripped of alpha, or alternatively a Gray image representing the extra channel (i.e. Alpha or Mask) ........................................ 414

IDOMImageColorConverterFilter
An image filter that presents a colour converted version of an image ................... 415

IDOMImageColorKeyFilter
An image filter that presents a masked image where colours within a given range are masked out, analogous to a green screen. The source image must not have a mask or alpha channel .................................... 416

IDOMImageColorSpaceSubstitutionFilter
An image filter that presents an identical image, just with the colourspace substituted .......................................................... 418

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 ........................................ 419

IDOMImageDeindexerFilter
An image filter that presents an image with an Indexed colour space as a simple eight bit image .......................................................... 420

IDOMImageDeviceNToBaseFilter
An image filter that presents an image with a DeviceN colour space as a simple image in the alternate space .......................................................... 422

IDOMImageDownsamplerFilter
An image filter that presents a downscaled version of an image .......................... 423

IDOMImageInverterFilter
An image filter that presents a bitwise inverted form of the source image ............. 424

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 a IDOMMaskedBrush where the brush masked by the image is a solid color ........................................ 426

IDOMImageProperties
The IDOMImageProperties interface provides access to an underlying implementation which stores miscellaneous information about the associated image ........................................ 427

IDOMInternalTarget
The IDOMInternalTarget interface describes the targets of hyperlinks that are in the same document but not on the current page ........................................ 432
5.1 Class List

IDOMJobTk
Represents an EDL JobTicket ................................................. 433

IDOMJobTkContent
Represents the content element of the JobTicket ............................ 438

IDOMJobTkGenericCharacterData
Interface to the IDOMJobTkGenericCharacterData node ..................... 449

IDOMJobTkGenericNode
Interface to the IDOMJobTkGenericNode node .................................. 451

IDOMJobTkNode
Represents a Job Ticket Node .................................................... 455

IDOMJobTkOwner
Interface to the IDOMJobTkOwner node ........................................ 460

IDOMJobTkValue
Represents a Job Ticket value element ......................................... 463

IDOMJPEGImage
Interface to a class representing a JPEG (.jpg or .jpeg) image ............. 465

IDOMLinearGradientBrush
IDOMLinearGradientBrush interface. A linear gradient brush is used to specify a gradient along a vector ........................................ 468

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 ......................................................... 471

IDOMMatrix
Defines the render transform matrix ............................................. 475

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 ......................................................... 478

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 ............................................. 478

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 ......................................................... 483

IDOMNodeFlags
A collection of bit flags used to signal various conditions of the node. For example, the eNodeRenderFlag flag identifies nodes that require rendering ............................................. 499

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 ............................................. 501

IDOMOPI
Base class representing OPI proxy. Has two descendant interfaces IDOMOPI13 and IDOMOPI20 ......................................................... 503

IDOMOPI13
The interface representing OPI proxy with accordance to Open Prepress Interface Specification 1.3 ......................................................... 505

IDOMOPI20
The interface representing OPI proxy with accordance to Open Prepress Interface Specification 2.0 ......................................................... 515

IDOMOutline
Represents the outline of the document, which is the collection of bookmarks for the document ......................................................... 517
IDOMOutlineEntry
Represents an index to a specific location in the document or a specific location external to the document ........................................ 520

IDOMPage
The base class for DOM page classes such as IDOMFixedPage ....................... 528

IDOMPageRectTarget
IDOMPageRectTarget nodes are used to describe hyperlinks on a page rectangle to targets on the same page .................................. 530

IDOMPageTarget
IDOMPageTarget nodes are used to describe hyperlinks on a page to targets on the same page ........................................ 536

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 PathFigures ................. 538

IDOMPathGeometry
Interface to a path geometry node .................................................. 544

IDOMPathNode
Interface to an EDL path node. A path node specifies a geometry that can be filled with a brush .................................................. 551

IDOMPathSegment
Interface to path segment element. The path segment is the smallest unit in a path geometry .................................................. 579

IDOMPCLImage
Interface to a class representing an image extracted from a PCLXL file .................. 582

IDOMPDFImage
Interface to a class representing an image extracted from a PDF file. Intended to be only used with the JawsMako APIs ...................................... 584

IDOMPNGImage
Interface to a class representing a PNG (.png) image .................................. 587

IDOMPolyBezierSegment
Interface to a path segment node describing a set of cubic Bézier curves ............... 591

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 ........................................ 594

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) ............... 596

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 .................................................. 599

IDOMPrintTicket
IDOMPrintTicket interface ............................................................... 602

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 ........................................ 604

IDOMRawDataFile
IDOMRawDataFile interface ............................................................... 609

IDOMRawImage
Interface to a class representing a raw image ........................................ 611
5.1 Class List

IDOMRecombineAlpha
Similar to IDOMRecombineImage, but instead combines an image comprising the colour 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 colour image. Images with Indexed colour spaces will be converted to the base spaces ......................................................... 614

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 ......................................................... 614

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 ......................................................... 616

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 ......................................................... 619

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 ......................................................... 622

IDOMSecurityInfo
Base DOM security class ......................................................... 628

IDOMShadingPatternBrush
IDOMShadingBrush provides a way of representing a PS style shading pattern ......................................................... 628

IDOMShadingPatternType1Brush
IDOMShadingBrush provides a way of representing a PS style type 1 shading pattern ......................................................... 633

IDOMShadingPatternType2Brush
IDOMShadingBrush provides a way of representing a PS style type 2 shading pattern ......................................................... 637

IDOMShadingPatternType3Brush
IDOMShadingPatternType3Brush provides a way of representing a PS style type 2 shading pattern ......................................................... 643

IDOMShadingPatternType4567Brush
IDOMShadingPatternType4567Brush provides a way of representing a PS style type 4 shading pattern ......................................................... 648

IDOMShape
Interface to an IDOMShape ......................................................... 657

IDOMSoftMaskBrush
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 ......................................................... 663

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 ......................................................... 667

IDOMStandardPDFSecurityInfo
Represents security information from PDF Standard encryption handler ......................................................... 670

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 ......................................................... 671

IDOMTarget
Base class for defining hyperlink targets in a document ......................................................... 675

IDOMTIFFImage
IDOMTIFFImage interface ......................................................... 678
IDOMTilingPatternBrush provides a way of representing a PS style tiling pattern.

IDOMTransformableBrush
Abstract interface for a brush to which a render transform may be applied.

IDOMTransparencyGroup
IDOMTransparencyGroup interface. Analogous to PDF Transparency groups.

IDOMType3Font
Representation of a PostScript/PDF Type 3 Font. At present, the stream cannot be set, only retrieved.

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

IDOMVisualRoot
IDOMVisualRoot interface.

IDOMWMPImage
IDOMWMPImage interface.

IEDLClassFactory
IEDL Class Factory.

IEDLError
An abstract class for EDL exceptions.

IEDLFontSystemFont
Representation of fonts installed on the target system (OS dependant).

IEDLNamespace
Interface to EDL Namespace 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.

IEDLStream
Generic stream. Abstract base class for EDL stream subsystem.

IEDLTempStore
A self-cleaning area for storage of temporary data in the form of streams. One per session, obtainable from an ISession.

IEDLTempStoreObject
A temporary, file-like object, stored with an IEDLTempStore.

IEDLTime
Interface to EDL date-time class.

IFontHeaderWriteSegmentBlockEnumerator
Enumerates over the PCLXL Font Header block items for the XL ReadFontHeader operator.

IFontPCL5WriteSegmentBlockEnumerator
Enumerates over the PCL5 font blocks.

JawsMako::IForm
An interface class for an interactive form, which tracks a tree of IFormField fields and widgets.

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

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.

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

JawsMako::JawsRenderer::IHalftone
An abstract base class for communicating halftone information to the Jaws renderer, for use with renderMonochrome() and renderMonochromeToFrameBuffer().

ImageDecoder
ImageDecoder returns ImageFrame objects as requested by the client. This object knows about the imageformat internals and knows how to unpack the image.
5.1 Class List

JawsMako::IImageDownsamplerTransform
A transform for downsampling images above a given effective resolution to a desired target effective resolution .................................................. 767

IImageEncoder
IImageEncoder accepts IImageFrame objects and streams it out to an image format .......... 771

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 colour converted if they are not compatible with the desired image type .......... 772

IImageFrame
IImageFrame encapsulates an EDL image with its details ........................................ 777

IImageFrameReader
IImageFrameReader reads an image into an imageframe ........................................ 783

IImageFrameWriter
IImageFrameWriter writes an image from an imageframe ........................................ 785

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 .......... 787

JawsMako::IIInput
A generic interface class for a ink annotation It is intended that future releases of JawsMako will extend this interface ........................................ 789

JawsMako::IIInput
Abstract input source that can open files from disk or a stream and create an IDocumentAssembly for the contents ........................................ 791

IInputEnum< typename T >
Iterator template class to allow iteration over a collection of instances of type <T> .......... 793

IInputEnumRC< typename T >
Reference-counted iterator template class to allow iteration over a collection of instances of type <T> ........................................ 793

IInputPushbackStream
Input Stream with pushback support ........................................ 794

IInputStream
Generic input stream. Abstract base class for all input streams ........................................ 795

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 ........................................ 805

JawsMako::IJawsRenderer
A renderer that uses the Jaws RIP to create images from arbitrary DOM ........................................ 808

JawsMako::ILLineAnnotation
An interface class for a line annotation. It is intended that future releases of JawsMako will extend this interface ........................................ 816

JawsMako::ILinkAnnotation
A generic interface class for a link annotation It is intended that future releases of JawsMako will extend this interface ........................................ 820

JawsMako::IMarkedContentArtifactDetails
A subclass of IMarkedContentDetails that is created when the content is a logical structure Artifact ........................................ 822

JawsMako::IMarkedContentDetails
Details of Marked Content applied to an IDOMGroup ........................................ 823

JawsMako::IMarkedContentStructureDetails
A subclass of IMarkedContentDetails that is created when the marked content is associated with the document’s structure ........................................ 824

JawsMako::IMarkupAnnotation
An interface class for markup annotations. It is intended that future releases of JawsMako will extend this interface ........................................ 825
<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>JawsMako::INamedDestination</td>
<td>A named destination in a PDF Document</td>
<td>828</td>
</tr>
<tr>
<td>JawsMako::IOptionalContent</td>
<td>Root level optional content information for an entire document</td>
<td>831</td>
</tr>
<tr>
<td>JawsMako::IOptionalContentConfiguration</td>
<td>A configuration for optional content</td>
<td>835</td>
</tr>
<tr>
<td>JawsMako::IOptionalContentDetails</td>
<td>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.</td>
<td>837</td>
</tr>
<tr>
<td>JawsMako::IOptionalContentFixerTransform</td>
<td>A simple transform that strips the DOM of any PDF optional content that is not visible for the given document use</td>
<td>839</td>
</tr>
<tr>
<td>JawsMako::IOptionalContentGroup</td>
<td>Interface for an optional content group</td>
<td>841</td>
</tr>
<tr>
<td>JawsMako::IOptionalContentGroupReference</td>
<td>A reference to an optional content group</td>
<td>842</td>
</tr>
<tr>
<td>JawsMako::IOptionalContentGroupUsage</td>
<td>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</td>
<td>843</td>
</tr>
<tr>
<td>JawsMako::IOptionalContentGroupUsageApplication</td>
<td>Interface for controlling how IOptionalContentGroupUsage is applied, and for what groups</td>
<td>846</td>
</tr>
<tr>
<td>JawsMako::IOptionalContentVisibilityExpression</td>
<td>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</td>
<td>848</td>
</tr>
<tr>
<td>JawsMako::IOutput</td>
<td>Abstract output sink that can output DOM to a file or stream in a given output format</td>
<td>850</td>
</tr>
<tr>
<td>JawsMako::IOutputAbort</td>
<td>A simple class, usable with IOutput and IOutputWriter to signal an abort of output</td>
<td>855</td>
</tr>
<tr>
<td>JawsMako::IOutputIntent</td>
<td>Interface class representing a PDF output intent</td>
<td>857</td>
</tr>
<tr>
<td>JawsMako::IOutputStream</td>
<td>Generic output stream. Abstract base class for output streams</td>
<td>859</td>
</tr>
<tr>
<td>JawsMako::IOutputWriter</td>
<td>A writer for writing individual pages and documents to an output in piecemeal fashion</td>
<td>864</td>
</tr>
<tr>
<td>JawsMako::IOverprintSimulationTransform</td>
<td>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</td>
<td>866</td>
</tr>
<tr>
<td>JawsMako::IPage</td>
<td>A page from an IDocument, allowing high level page management, and providing on-demand access to page contents</td>
<td>867</td>
</tr>
<tr>
<td>JawsMako::IPageCropperTransform</td>
<td>Very simple transform for cropping pages to one of the standard boxes</td>
<td>871</td>
</tr>
<tr>
<td>JawsMako::IPageLayout</td>
<td>Analyze the layout of a FixedPage, grouping together text deemed to be in horizontal and/or vertical blocks. Useful for text search and selection</td>
<td>872</td>
</tr>
<tr>
<td>JawsMako::IPageLayoutData</td>
<td>Provides a representation of the analyzed page layout by organizing and allowing access to collections of IPageLayoutNodes</td>
<td>874</td>
</tr>
<tr>
<td>JawsMako::IPageLayoutNode</td>
<td>Simple data type representing a part of an analyzed page</td>
<td>876</td>
</tr>
<tr>
<td>JawsMako::IPCL5Input</td>
<td>An instance of the JawsMako PCL5 input class</td>
<td>877</td>
</tr>
<tr>
<td>JawsMako::IPCL5Output</td>
<td>Interface for the PCL5 IOutput class</td>
<td>882</td>
</tr>
<tr>
<td>JawsMako::IPCLXLInput</td>
<td>An instance of the JawsMako PCL/XL input class</td>
<td>885</td>
</tr>
</tbody>
</table>
JawsMako::IPCLXLOutput
    Interface for the PCLXL IOutput class .............................................. 890
JawsMako::IPDFInput
    An instance of the JawsMako PDF input class .................................... 891
JawsMako::IPDFOutput
    Interface for the PDF IOutput class .............................................. 895
JawsMako::IPJLParser
    An instance of the Mako PJL Parser ................................................. 912
JawsMako::IPolyAnnotation
    An interface class for a polygon or polyline annotation. It is intended that future releases of JawsMako will extend this interface .............................................. 915
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 .............................................. 916
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 .............................................. 918
JawsMako::IPSOutput
    Interface for the PS IOutput class .............................................. 923
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() .............................................. 925
IRAInputPushbackStream
    Random-access Input Stream with pushback support ................................... 927
IRAInputStream
    Random Access Input Stream ......................................................... 928
IRAOutputStream
    Random Access Output Stream ....................................................... 929
IRAStruct
    Abstract base class for "Random-Access" streams i.e. streams that can be arbitrarily re-positioned ......................................................... 930
IRXObject
    Base class Interface for all Reference Counted objects ................................... 932
JawsMako::IRedactionAnnotation
    A generic interface class for a redaction annotation .................................. 934
JawsMako::IRedactorTransform
    A transform for applying redaction redactions ..................................... 937
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 ..................................... 939
IRRunnable
    Interface to filter's runnable classes .............................................. 944
ISession
    EDL session class ......................................................... 946
JawsMako::IShapeAnnotation
    A generic interface class for circle and square annotations. It is intended that future releases of JawsMako will extend this interface .............................................. 950
JawsMako::ISkiaRenderer
    A renderer that can paint XPS compatible DOM into a Skia canvas using the Skia API .............................................. 952
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 .............................................. 953
JawsMako::ISetAnnotation
    A generic interface class for a stamp annotation .................................. 955

Generated by Doxygen
JawsMako::IStrokerTransform
A transform for converting some or all stroked paths into plain filled paths .......................... 957

JawsMako::IStructure
Top level tracking structure describing the logical structure of the document ........................... 960

JawsMako::IStructureElement
A structure element in the structure tree ................................................................. 962

JawsMako::IStructureElementChild
A child of a structure element. Either points to actual marked content, or another structure element ................................. 964

JawsMako::IStructureElementReference
A token-like class encapsulating a reference to a structure element .................................... 965

JawsMako::I StructureElementReferenceChild
A child of a structure element that points to another structure element .............................. 966

JawsMako::I StructureMarkedContentReferenceChild
A child of a structure element that points to a piece of marked content. Note; to create these, please see IStructureElement::createMarkedContentReferencePair() ........................................... 967

JawsMako::I StructureObjectReferenceChild
A child of a structure element that points to a piece of marked content. These cannot be created directly. Instead use IStructureElement::appendToObjectReferenceChild() or IStructureElement::insertObjectReferenceChild(). ................................. 968

JawsMako::ISVGGenerator
A SVG generator for JawsMako, allowing simple generation of SGG fragments for individual DOM nodes or entire pages ................................................................. 969

JawsMako::ITextAnnotation
A generic interface class for a text (sticky note) annotation ........................................... 973

JawsMako::ITextMarkupAnnotation
A generic interface class for a text markup annotation It is intended that future releases of JawsMako will extend this interface ................................. 975

JawsMako::ITextRun
A run of text, containing unicode information, the position, transformation and bounds of the text 977

JawsMako::ITextSearch
Perform text searching using the page information obtained from an IPageLayout ........................ 979

JawsMako::ITextSelect
Perform text selection using the page information obtained from an IPageLayout .................. 980

JawsMako::IThreads
An interface class for document threads, Currently a stub interface ................................. 982

JawsMako::ITransform
ITransforms provide a method of applying common operations on DOM objects such as brushes, nodes, colours, colourspaces or entire trees. Not all transforms will operate on all kinds of objects, as noted in their descriptions ................................................................. 983

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 ................................................................. 987

JawsMako::IType3UnpackerTransform
A transform for unpacking glyphs using a Type 3 font into regular DOM ................................ 991

JawsMako::IUnicodeHelper
An interface into language specific unicode helpers ................................................... 992

JawsMako::IWidgetAnnotation
An interface class for a widget annotation It is intended that future releases of JawsMako will extend this interface ................................................................. 994

JawsMako::IXAMLGenerator
A XAML generator for JawsMako, allowing simple generation of XAML fragments for individual DOM nodes or entire pages ................................................................. 997

JawsMako::IXPSInput
An instance of the JawsMako XPS input class ................................................................. 1007

JawsMako::IXPSOutput
Interface for the XPS IOutput class ................................................................. 1009

Generated by Doxygen
**IDOMPDFImage::JBIG2Params**

Class to hold filter parameters for JBIG2-compressed image data. Please see the PDF specification for the meaning of these parameters ................................................................. 1016

**PValue**

Stores a "property" value that is tagged with an enumeration value that indicates the underlying type ........................................................................................................... 1016

**SignatureID**

Opentype table signatures ................................................................................................................. 1017
Chapter 6

File Index

6.1 File List

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

- **edlblackpointcompensation.h**
  Simple enum for black point compensation ........................................ 1019
- **edliblend.h**
  An enum for transparency blend modes ............................................. 1020
- **edlerrors.cpp**
  Publishes a look-up table that maps integer error codes, as returned by many of the EDL APs, into a corresponding (English) string. It is expected that alternate language translations are implemented by creating alternate versions of this file ........................................ 1020
- **edlerrors.h**
  EDL errors ....................................................................................... 1021
- **edlfwd.h**
  A header containing forward declarations for certain EDL interfaces .................. 1027
- **edlgeom.h**
  Geometry primitives including: point, rectangle and matrix types supporting both integer and floating point values within ................................................................. 1027
- **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 ................................................................. 1028
- **edlnamespaces.h**
  EDL C++ namespace(s) ........................................................................ 1028
- **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.) ................................................................. 1029
- **edlqname.h**
  IEDLNamespace and EDLQName classes ................................................. 1030
- **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 ................................................................. 1030
- **edlrenderingintent.h**
  Simple enum for rendering intent ......................................................... 1030
- **edlstream.h**
  EDL provides a collection of "stream" classes that are supplied to, and returned by, EDL APIs that access files or data streams ................................................................. 1031
<table>
<thead>
<tr>
<th>Header File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>edcstring.h</td>
<td>EDLString and EDLSysString classes and associated EDL string manipulation functions</td>
</tr>
<tr>
<td>edltimetype.h</td>
<td>IEDLTime class that represents the EDL date-time</td>
</tr>
<tr>
<td>edltypes.h</td>
<td>EDL &quot;standard&quot; types including known bit-length signed and unsigned integer type[def]s and definitions of EDL_TRUE and EDL_FALSE</td>
</tr>
<tr>
<td>edlvector.h</td>
<td>Simple buffer class which preserves the allocation context where the buffers were created</td>
</tr>
<tr>
<td>edlversion.h</td>
<td>Defines EDL major and minor version numbers</td>
</tr>
<tr>
<td>icolormanager.h</td>
<td>Public interface to the EDL color manager</td>
</tr>
<tr>
<td>idombrush.h</td>
<td>Interfaces to DOM brush objects</td>
</tr>
<tr>
<td>idomcanvas.h</td>
<td>IDOMCanvas interface</td>
</tr>
<tr>
<td>idomcatalog.h</td>
<td>IDOMCatalog Interface</td>
</tr>
<tr>
<td>idomcharpathgroup.h</td>
<td>IDOMCharPathGroup interface</td>
</tr>
<tr>
<td>idomcolor.h</td>
<td>IDomColor Interface</td>
</tr>
<tr>
<td>idomcolors.h</td>
<td>Idomcolors Interface</td>
</tr>
<tr>
<td>idomedgemode.h</td>
<td>Enumeration for edge mode, used to flag when a node should not be rendered using anti-aliasing on anti-aliasing renderers</td>
</tr>
<tr>
<td>idomfont.h</td>
<td>IDOMFont interface</td>
</tr>
<tr>
<td>idomform.h</td>
<td>IDOMGroup interface</td>
</tr>
<tr>
<td>idomglyph.h</td>
<td>IDOMGlyph Interface</td>
</tr>
<tr>
<td>idomglyphs.h</td>
<td>IDOMGlyphs Interface</td>
</tr>
<tr>
<td>idomgroup.h</td>
<td>IDOMGroup interface</td>
</tr>
<tr>
<td>idomhashable.h</td>
<td>Abstract interface for objects that can be hashed</td>
</tr>
<tr>
<td>idomid.h</td>
<td>DOMid Interface</td>
</tr>
<tr>
<td>idomimageresource.h</td>
<td>IDOMResources interface</td>
</tr>
<tr>
<td>idomjobtk.h</td>
<td>DOM Job Ticket Interface</td>
</tr>
<tr>
<td>idommetadata.h</td>
<td>IDOMMetaData Interface</td>
</tr>
<tr>
<td>idomnode.h</td>
<td>IDOMNode Interface</td>
</tr>
<tr>
<td>idomopip.h</td>
<td>Interface to EDL DOM OPI</td>
</tr>
<tr>
<td>idomoutline.h</td>
<td>Page and Fixed page interfaces</td>
</tr>
<tr>
<td>idompath.h</td>
<td>Path related interfaces of the DOM</td>
</tr>
<tr>
<td>idompathgeometry.h</td>
<td>DOM interfaces for path geometry, figures and segments</td>
</tr>
<tr>
<td>File Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>idomresources.h</td>
<td>IDOMResources interface</td>
</tr>
<tr>
<td>idomsecurity.h</td>
<td></td>
</tr>
<tr>
<td>idomshape.h</td>
<td>Classes for handling Shapes</td>
</tr>
<tr>
<td>idomtarget.h</td>
<td>IDOMTarget Interface</td>
</tr>
<tr>
<td>iedcollection.h</td>
<td>EDL &quot;collection&quot; interface</td>
</tr>
<tr>
<td>iedenum.h</td>
<td>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 EDL SysStrings that can then be iterated over</td>
</tr>
<tr>
<td>iedifactory.h</td>
<td>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</td>
</tr>
<tr>
<td>iediobject.h</td>
<td>EDL Object Interface</td>
</tr>
<tr>
<td>ieditempstore.h</td>
<td>A mechanism for storing and accessing temporary data for use with EDL</td>
</tr>
<tr>
<td>iedtree.h</td>
<td>EDL &quot;tree&quot; interface</td>
</tr>
<tr>
<td>ifilespec.h</td>
<td>Interface to File specification</td>
</tr>
<tr>
<td>imagecodec.h</td>
<td></td>
</tr>
<tr>
<td>interactive.h</td>
<td>JawsMako interactive features</td>
</tr>
<tr>
<td>ircobject.h</td>
<td>Interface for Reference Counted Object</td>
</tr>
<tr>
<td>isession.h</td>
<td>The JawsMako library API</td>
</tr>
<tr>
<td>jawsmako.h</td>
<td></td>
</tr>
<tr>
<td>memutils.h</td>
<td>EDL portability wrappers around memset(),memcpy() and memcmp() to allow EDL to use alternate implementations where necessary</td>
</tr>
<tr>
<td>objclassid.h</td>
<td>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</td>
</tr>
<tr>
<td>optionalcontent.h</td>
<td>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</td>
</tr>
<tr>
<td>outputintent.h</td>
<td>Declarations of interfaces for querying output intents</td>
</tr>
<tr>
<td>pcl5input.h</td>
<td>PCL5 Input-Specific Features</td>
</tr>
<tr>
<td>pcl5output.h</td>
<td>JawsMako PCL5 Output</td>
</tr>
<tr>
<td>pclxlinput.h</td>
<td>PCL/XL Input-Specific Features</td>
</tr>
<tr>
<td>pclxloutput.h</td>
<td>JawsMako PCLXL Output</td>
</tr>
<tr>
<td>pdfinput.h</td>
<td>PDF Input-Specific Features</td>
</tr>
<tr>
<td>Header</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>pdfoutput.h</td>
<td>JawsMako PDF Output</td>
</tr>
<tr>
<td>pjl.h</td>
<td>A PJL (Printer Job Language) parser for Mako</td>
</tr>
<tr>
<td>platform.h</td>
<td>Platform-dependent defines, enumerations, types etc. that are visible through the EDL API</td>
</tr>
<tr>
<td>platform_utils.h</td>
<td>Platform-dependent functions that are visible through the EDL API</td>
</tr>
<tr>
<td>psoutput.h</td>
<td>JawsMako PS Output</td>
</tr>
<tr>
<td>skiarenderer.h</td>
<td>A Renderer for XPS compatible DOM using Skia</td>
</tr>
<tr>
<td>smartptr.h</td>
<td>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</td>
</tr>
<tr>
<td>structure.h</td>
<td>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</td>
</tr>
<tr>
<td>svggenerator.h</td>
<td>A svg generator for JawsMako</td>
</tr>
<tr>
<td>text.h</td>
<td>JawsMako Text Conveniences</td>
</tr>
<tr>
<td>transforms.h</td>
<td>Transforms for JawsMako, allowing direct modification of individual brushes, nodes, or entire pages</td>
</tr>
<tr>
<td>types.h</td>
<td>Common types and required headers for the JawsMako interface</td>
</tr>
<tr>
<td>xamlgenerator.h</td>
<td>A xaml generator for JawsMako</td>
</tr>
<tr>
<td>xpsinput.h</td>
<td>XPS Input-Specific Features</td>
</tr>
<tr>
<td>xpsoutput.h</td>
<td>JawsMako XPS Output</td>
</tr>
</tbody>
</table>
Chapter 7

Module Documentation

7.1 API

The JawsMako API.

Files

- `file interactive.h`
  JawsMako interactive features.
- `file jawsmako.h`
  The JawsMako library API.
- `file optionalcontent.h`
  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.
- `file outputintent.h`
  Declarations of interfaces for querying output intents.
- `file pcl5input.h`
  PCL5 Input-Specific Features.
- `file pcl5output.h`
  JawsMako PCL5 Output.
- `file pclxinput.h`
  PCL/XL Input-Specific Features.
- `file pclxoutput.h`
  JawsMako PCLXL Output.
- `file pdfinput.h`
  PDF Input-Specific Features.
- `file pdfoutput.h`
  JawsMako PDF Output.
- `file pjl.h`
  A PJL (Printer Job Language) parser for Mako.
- `file psoutput.h`
  JawsMako PS Output.
- `file skiarenderer.h`
  A Renderer for XPS compatible DOM using Skia.
• file `structure.h`
  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.

• file `svggenerator.h`
  A svg generator for JawsMako.

• file `text.h`
  JawsMako Text Conveniences.

• file `transforms.h`
  Transforms for JawsMako, allowing direct modification of individual brushes, nodes, or entire pages.

• file `types.h`
  Common types and required headers for the JawsMako interface.

• file `xamlgenerator.h`
  A xaml generator for JawsMako.

• file `xpsinput.h`
  XPS Input-Specific Features.

• file `xpsoutput.h`
  JawsMako XPS Output.

### Classes

- **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, colours, colourspaces 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 colour 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.

- **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.

- **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::IStrokerTransform**
  Generated by Doxygen

Generated by Doxygen
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::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::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::IField`
  An interface class for an interactive form, which tracks a tree of `IFormFields` and widgets.

- 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. It is intended that future releases of JawsMako will extend this interface.

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

- 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 an 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.

• 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.

• class JawsMako::IOutputIntent
  Interface class representing a PDF output intent.

• class JawsMako::IPCL5Input
  An instance of the JawsMako PCL5 input class.

• class JawsMako::IPCL5Output
  Interface for the PCL5 IOutput class.

• class JawsMako::IPCLXLInput
  An instance of the JawsMako PCL/XL input class.
• class JawsMako::IPCLXLOutput
  Interface for the PCLXL IOutput class.

• class JawsMako::IPDFInput
  An instance of the JawsMako PDF input class.

• class JawsMako::IPDFOutput
  Interface for the PDF IOutput class.

• class JawsMako::PJLParse
  An instance of the Mako PJL Parser.

• class JawsMako::IPSOutput
  Interface for the PS IOutput class.

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

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

• 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 IStructure::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()

• class JawsMako::ISVGGenerator
  A SVG generator for JawsMako, allowing simple generation of SGG fragments for individual DOM nodes or entire pages.

• 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::IPLayout
 
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.
- **class JawsMako::CTemporaryStoreParameters**: Allows the temporary storage parameters to be optionally overridden.
- **class JawsMako::IXAMLGenerator**: A XAML generator for JawsMako, allowing simple generation of XAML fragments for individual DOM nodes or entire pages.
- **class JawsMako::IXPSOutput**: Interface for the XPS IOutput class.

**Typedefs**

- typedef CEDLVector<ITransformPtr> JawsMako::CTransformVect
  A vector of transform instances.
- 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.

**Enumerations**

- enum JawsMako::eFieldType
  The type of a form field. These map to field types present in the pdf specification.
- enum JawsMako::eAppearanceUsage { JawsMako::eAUNormal, JawsMako::eAURollover, JawsMako::eAUDown }
  The usage scenario for an annotation appearance.
- 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.

### 7.1.1 Detailed Description

The JawsMako API.

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

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

This API will be extended in future JawsMako releases.
7.1.2 Enumeration Type Documentation

7.1.2.1 eAppearanceUsage

define JawsMako::eAppearanceUsage

The usage scenario for an annotation appearance.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eAUNormal</td>
<td>The normal appearance is used when the annotation is not interacting with the user. This appearance is also used for printing the annotation.</td>
</tr>
<tr>
<td>eAURollover</td>
<td>The rollover appearance is used when the user moves the cursor into the annotation’s active area without pressing the mouse button.</td>
</tr>
<tr>
<td>eAUDown</td>
<td>The down appearance is used when the mouse button is pressed or held down within the annotation’s active area.</td>
</tr>
</tbody>
</table>
Chapter 8

Class Documentation

8.1 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.

```cpp
#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`

8.1.1 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.
8.1.2 Member Enumeration Documentation

8.1.2.1 eBorderType

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

Types of border.

Enumerator

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eBTSolid</td>
<td>A solid rectangle surrounding the annotation.</td>
</tr>
<tr>
<td>eBTDashed</td>
<td>A dashed rectangle surrounding the annotation.</td>
</tr>
<tr>
<td>eBTBeveled</td>
<td>A simulated embossed rectangle that appears to be raised above the surface of the page.</td>
</tr>
<tr>
<td>eBTInset</td>
<td>A simulated engraved rectangle that appears to be recessed below the surface of the page.</td>
</tr>
<tr>
<td>eBTUnderline</td>
<td>A single line along the bottom of the annotation rectangle.</td>
</tr>
</tbody>
</table>
```

8.1.3 Constructor & Destructor Documentation

8.1.3.1 CAnnotationBorder()

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

Constructor.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inWidth</td>
<td>Border width. Default width is 1 PDF unit, or 1/72nd of an inch</td>
</tr>
<tr>
<td>inType</td>
<td>Border type. Default is eBTSolid</td>
</tr>
<tr>
<td>inDash</td>
<td>A vector of floats describing a eBTDashed dash pattern</td>
</tr>
</tbody>
</table>

Returns

- `CAnnotationBorder` The `CAnnotationBorder` object

8.1.4 Member Data Documentation
8.1.4.1 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).

8.1.4.2 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

8.2 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.

8.2.1 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

Generated by Doxygen
8.3 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:

```
IRCObject

IDOMPDFImage::IDecodeParams

IDOMPDFImage::CCITTFaxParams
```

Additional Inherited Members

8.3.1 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

8.4 CClassID Class Reference

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

#include <objclassid.h>
8.4 CClassID Class Reference

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.
- **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.

8.4.1 Detailed Description

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

8.4.2 Constructor & Destructor Documentation

8.4.2.1 CClassID()

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

Converts hexadecimal representation of a CClassID to a CClassID.

Parameters

| str | String containing the hexadecimal value |

Returns

  CClassID The CClassID

8.4.2.2 CClassID()

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

Copy to another CClassID.
## 8.4.2.3 `CClassID()` [3/3]

`CClassID::CClassID`

```cpp
    uint32 dw0,
    uint32 dw1,
    uint32 dw2,
    uint32 dw3 ) [inline]
```

Construct `CClassID` from 4 x `uint32`.

### Parameters

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

### Returns

- `CClassID` The new `CClassID`

## 8.4.3 Member Function Documentation

### 8.4.3.1 `equal()`

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

Compare to another `CClassID`.

### Parameters

- `id` `CClassID` with which to compare

### Returns

- `bool` True if equal, false if not

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

- `objcclassid.h`
8.5 CClassParams Class Reference

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.

#include <iedlobject.h>

Inherited by CFileStreamParams, CInputMemoryParams, CRAUserStreamReadParams, CUserStreamReadParams, CUserStreamWriteParams, IDOMArcSegment::Data, IDOMAudioFile::Data, IDOMCanvas::Data, IDOMCharPathGroup::Data, IDOMColorSpaceDeviceN::Data, IDOMColorSpaceCCBased::Data, IDOMColorSpaceIndexed::Data, IDOMColorSpaceLAB::Data, IDOMDePremultiplierFilter::Data, IDOMDeviceNColorant::Data, IDOMExponentialFunction::Data, IDOMFixedPage::Data, IDOMFont::Data, IDOMFontSource::Data, IDOMForm::Data, IDOMGroupingFunction::Data, IDOMICMPProfile::Data, IDOMImage::Data, IDOMImageBlitScalerFilter::Data, IDOMImageBleederFilter::Data, IDOMImageBrush::Data, IDOMImageChannelSelectorFilter::Data, IDOMImageColorConverterFilter::Data, IDOMImageColorKeyFilter::Data, IDOMImageColorSpaceSubstitutionFilter::Data, IDOMImageDownsamplerFilter::Data, IDOMImageDeindexerFilter::Data, IDOMImageDeviceNToBaseFilter::Data, IDOMImageDecodeFilter::Data, IDOMImageInverterFilter::Data, IDOMImageMaskExpanderFilter::Data, IDOMImageMatteRemoverFilter::Data, IDOMJobTkContent::Data, IDOMJobTkGenericCharacterData::Data, IDOMJobTkGenericNode::Data, IDOMJobTkNode::Data, IDOMJobTkValue::Data, IDOMLinearGradientBrush::Data, IDOMMaskedBrush::Data, IDOMMatrix::Data, IDOMNullBrush::Data, IDOMOPI13::Data, IDOMOPI20::Data, IDOMOutline::Data, IDOMOutlineEntry::Data, IDOMPathFigure::Data, IDOMPathGeometry::Data, IDOMPathNode::Data, IDOMPolyBezierSegment::Data, IDOMPolyLineSegment::Data, IDOMPolyQuadraticBezierSegment::Data, IDOMPostScriptCalculatorFunction::Data, IDOMPrintTicket::Data, IDOMRadialGradientBrush::Data, IDOMRawDataFile::Data, IDOMResourceDictionary::Data, IDOMSampledFunction::Data, IDOMShadingPatternType1Brush::Data, IDOMShadingPatternType2Brush::Data, IDOMShadingPatternType3Brush::Data, IDOMShadingPatternType4567Brush::Data, IDOMShape::Data, IDOMSoftMaskBrush::Data, IDOMSolidColorBrush::Data, IDOMStitchingFunction::Data, IDOMTilingPatternBrush::Data, IDOMTransparencyGroup::Data, IDOMType3Font::Data, IDOMVisualBrush::Data, IEDLNamespace::Data, IEDLTempStore::Data, IEDLTime::Data, IFileSpecAsEmbeddedData::Data, IFileSpecAsFileReference::Data, and IFileSpecAsUrl::Data.

8.5.1 Detailed Description

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

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

- iedlobject.h

8.6 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:

![Inheritance Diagram]

**Public Member Functions**

- `COrderEntryPtr clone ()`
  
  *Create a deep copy.*

**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**

**8.6.1 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`

**8.7 JawsMako::IPDFInput::CPdfFontInfo Class Reference**

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

```
#include <pdfinput.h>
```
8.7.1 Detailed Description

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

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

- \texttt{pdfinput.h}

8.8 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>

8.8.1 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 \texttt{#IRendererTransform::findInks()}

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

- \texttt{pdfinput.h}

8.9 JawsMako::IPJLParser::CPjIAttributeValue Class Reference

A captured PJL attribute.

\#include <pjll.h>

Public Attributes

- \texttt{RawString modifier}
- \texttt{RawString key}
- \texttt{RawString value}

8.9.1 Detailed Description

A captured PJL attribute.

8.9.2 Member Data Documentation
8.9.2.1 key

**RawString** JawsMako::IPJLPParser::CPjlAttributeValue::key

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

8.9.2.2 modifier

**RawString** JawsMako::IPJLPParser::CPjlAttributeValue::modifier

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

8.9.2.3 value

**RawString** JawsMako::IPJLPParser::CPjlAttributeValue::value

The PJL value, if present. Unmodified.

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

- pjl.h

8.10 JawsMako::CQuadPoint Class Reference

A representation of a PDF Quadpoint, in DOM coordinates.

#include <interactive.h>

8.10.1 Detailed Description

A representation of a PDF Quadpoint, in DOM coordinates.

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

- interactive.h

8.11 JawsMako::CRectInset Class Reference

A class which specifies an inset from a rectangle.

#include <interactive.h>
8.12 JawsMako::IJawsRenderer::CSpotHalftone Class Reference

8.11.1 Detailed Description

A class which specifies an inset from a rectangle.

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

- interactive.h

8.12 JawsMako::IJawsRenderer::CSpotHalftone Class Reference

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

#include <jawsmako.h>

Inheritance diagram for JawsMako::IJawsRenderer::CSpotHalftone:

Public Attributes

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

- bool useFullResolutionForFlattening
  
  *Normally, the resolution at which flattening for transparent.*

8.12.1 Detailed Description

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

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

- jawsmako.h
8.13 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.

8.13.1 Detailed Description

Allows the temporary storage parameters to be optionally overridden.

8.13.2 Constructor & Destructor Documentation

8.13.2.1 CTemporaryStoreParameters()

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

Initialise the temporary store parameter block.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>memoryLimit</td>
<td>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 8GB, 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.</td>
</tr>
<tr>
<td>diskLimit</td>
<td>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.</td>
</tr>
<tr>
<td>blockSize</td>
<td>Set the block size to use. The default is currently 4096 bytes. Each file in the temporary store will occupy a multiple of the block size. Note that currently, the total number of blocks must be less than 1. That is, (diskLimit + memoryLimit) / blockSize must be less than 2147483647.</td>
</tr>
</tbody>
</table>

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

• types.h
### 8.14 JawsMako::IJawsRenderer::CThresholdArrayHalftone Class Reference

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

```cpp
#include <jawsmako.h>
```

Inheritance diagram for JawsMako::IJawsRenderer::CThresholdArrayHalftone:

![Inheritance Diagram](image)

#### Public Attributes

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

#### 8.14.1 Detailed Description

Description of a Type 3 8-bit threshold array halftone for use with renderMonochrome. 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`
8.15 JawsMako::IJawsRenderer::CThresholdHalftone Class Reference

A halftone representing a simple threshold.

#include <jawsmako.h>

Inheritance diagram for JawsMako::IJawsRenderer::CThresholdHalftone:

8.15.1 Detailed Description

A halftone representing a simple threshold.

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

• jawsmako.h

8.16 CTransformMatrix< TItem > Class Template Reference

Matrix class - special 3x2 matrix.

#include <edlgeom.h>

Public Types

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

- **CTransformMatrix ()**
  Creates zero matrix.
- **CTransformMatrix (TItem _xx, TItem _xy, TItem _yx, TItem _yy, TItem _dx, TItem _dy)**
  Creates the matrix elementary.
- **CTransformMatrix (const CTransformMatrix< TItem >&m)**
  Copy constructor.
- **CTransformMatrix (const RectTmpl< TItem >&sourceRect, const RectTmpl< TItem >&destRect)**
  Creates a matrix that transforms from one rectangle to another.
- **void set (TItem _xx=1, TItem _xy=0, TItem _yx=0, TItem _yy=1, TItem _dx=0, TItem _dy=0)**
  Create blank matrix.
- **TItem xx () const**
  Returns xx component.
- **TItem xy () const**
  Returns xy component.
- **TItem yx () const**
  Returns yx component.
- **TItem yy () const**
  Returns yy component.
- **TItem dx () const**
  Returns dx component.
- **TItem dy () const**
  Returns dy component.
- **void setXX (TItem x)**
  Sets xx component.
- **void setXY (TItem x)**
  Sets xy component.
- **void setYX (TItem x)**
  Sets yx component.
- **void setYY (TItem x)**
  Sets yy component.
- **void setDX (TItem x)**
  Sets dx component.
- **void setDY (TItem x)**
  Sets dy component.
- **bool equal (const CTransformMatrix< TItem >&matrix, bool ignoreDXDY=false) const**
  Compare to another matrix.
- **bool identity (bool ignoreDXDY=false) const**
  Determine if identity matrix.
- **CTransformMatrix< TItem >& preMul (const CTransformMatrix< TItem >&matrix)**
  Premultiply by given matrix.
- **CTransformMatrix< TItem >& postMul (const CTransformMatrix< TItem >&matrix)**
  Postmultiply by given matrix.
- **bool invert ()**
  Invert the matrix.
- **void transform (PointTmpl< TItem >&result, 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.

- void **rotate** (double radians)
  
  Add a rotation, clockwise, in radians.

- void **scale** (TItem xscale, TItem yscale)
  
  Scale.

- void **translate** (TItem dx, TItem dy)
  
  Translate.

- void **transformRect** (RectTmpl<TItem> &rect, bool ignoreDXDY=false) const
  
  Transform a rectangle.

- uint32 **classify** () const
  
  Classify the transform.

### 8.16.1 Detailed Description

```cpp
template<typename TItem>
class CTransformMatrix<TItem>
```

Matrix class - special 3x2 matrix.

### 8.16.2 Member Enumeration Documentation

#### 8.16.2.1 eOperationTypes

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

Classification of operation type flags of the transform.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eDoesTranslate</td>
<td>Does translate.</td>
</tr>
<tr>
<td>eDoesScale</td>
<td>Does scale.</td>
</tr>
<tr>
<td>eDoesRotate</td>
<td>Does rotate.</td>
</tr>
<tr>
<td>eIsComplex</td>
<td>Is complex.</td>
</tr>
</tbody>
</table>

### 8.16.3 Constructor & Destructor Documentation

#### 8.16.3.1 CTransformMatrix() [1/3]

```cpp
template<typename TItem>
CTransformMatrix<TItem>::CTransformMatrix()
```
8.16 CTransformMatrix< TItem > Class Template Reference

```
TItem _xx,
TItem _xy,
TItem _yx,
TItem _yy,
TItem _dx,
TItem _dy ) [inline]
```

Creates the matrix elementary.

Parameters

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

Returns

CTransformMatrix The matrix

8.16.3.2 CTransformMatrix() [2/3]

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

Copy constructor.

Parameters

| m | Transform matrix |

8.16.3.3 CTransformMatrix() [3/3]

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

Creates a matrix that transforms from one rectangle to another.

Parameters

| sourceRect | Source rectangle |
| destRect   | Destination rectangle |
8.16.4  Member Function Documentation

8.16.4.1  classify()

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

Classify the transform.

Returns

```
uint32 Operation flags
```

8.16.4.2  dx()

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

Returns dx component.

Returns

```
TItem dx component
```

8.16.4.3  dy()

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

Returns dy component.

Returns

```
TItem dy component
```

8.16.4.4  equal()

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

Compare to another matrix.
8.16 CTransformMatrix<TItem> Class Template Reference

Parameters

<table>
<thead>
<tr>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>matrix</td>
<td>Matrix to compare current instance to</td>
</tr>
<tr>
<td>ignoreDXDY</td>
<td>Ignore DX &amp; DY for this comparison</td>
</tr>
</tbody>
</table>

Returns

**bool** True if equal, false if not

8.16.4.5 identity()

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

Determine if identity matrix.

Parameters

<table>
<thead>
<tr>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ignoreDXDY</td>
<td>Ignore DX &amp; DY for this operation</td>
</tr>
</tbody>
</table>

Returns

**bool** True if identity matrix, false if not

8.16.4.6 invert()

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

Invert the matrix.

Returns

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

8.16.4.7 iTransform()

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

Transform a point by the inverse of the matrix.

Generated by Doxygen
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>result</td>
<td>PointTmpl&lt;TItem&gt; to accept result</td>
</tr>
<tr>
<td>point</td>
<td>The point</td>
</tr>
<tr>
<td>ignoreDXDY</td>
<td>Ignore DX &amp; DY for this operation</td>
</tr>
</tbody>
</table>

Returns

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

8.16.4.8 postMul()

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

Postmultiply by given matrix.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>matrix</td>
<td>Matrix postmultiplier</td>
</tr>
</tbody>
</table>

Returns

CTransformMatrix<TItem> Resulting matrix

8.16.4.9 preMul()

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

Premultiply by given matrix.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>matrix</td>
<td>Matrix premultiplier</td>
</tr>
</tbody>
</table>

Returns

CTransformMatrix<TItem> Resulting matrix
8.16.4.10 rotate()

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

Add a rotation, clockwise, in radians.

**Parameters**

| radians | Angle in radians |

8.16.4.11 scale()

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

Scale.

**Parameters**

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

8.16.4.12 set()

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

Create blank matrix.

**Parameters**

| _xx  | xx component |
| _xy  | xy component |
| _yx  | yx component |
| _yy  | yy component |
| _dx  | dx component |
| _dy  | dy component |
8.16.4.13 setDX()

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

Sets dx component.

Parameters

| x | value to set |

8.16.4.14 setDY()

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

Sets dy component.

Parameters

| x | value to set |

8.16.4.15 setXX()

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

Sets xx component.

Parameters

| x | value to set |

8.16.4.16 setXY()

```cpp
template<typename TItem>
void CTransformMatrix<TItem>::setXY ( TItem x ) [inline]
```
Sets xy component.

Parameters

\[ x \text{ value to set} \]

8.16.4.17 setYX()

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

Sets yx component.

Parameters

\[ x \text{ value to set} \]

8.16.4.18 setYY()

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

Sets yy component.

Parameters

\[ x \text{ value to set} \]

8.16.4.19 transform()

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

Transform a point.

Parameters

\[ \begin{array}{|c|l|}
\hline
\text{result} & \text{The point after transformation} \\
\hline
\text{point} & \text{The point before transformation} \\
\hline
\text{ignoreDXDY} & \text{Ignore DX & DY for this operation} \\
\hline
\end{array} \]


8.16.4.20 transformRect()

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

Transform a rectangle.

Transform a rectangle

Parameters

<table>
<thead>
<tr>
<th>rect</th>
<th>Rectangle to transform</th>
</tr>
</thead>
<tbody>
<tr>
<td>ignoreDXDY</td>
<td>Ignore DX &amp; DY for this operation</td>
</tr>
</tbody>
</table>

8.16.4.21 translate()

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

Translate.

Parameters

<table>
<thead>
<tr>
<th>dx</th>
<th>x-axis transformation value</th>
</tr>
</thead>
<tbody>
<tr>
<td>dy</td>
<td>y-axis transformation value</td>
</tr>
</tbody>
</table>

8.16.4.22 xx()

```cpp
TItem CTransformMatrix<TItem>::xx ( ) const [inline]
```

Returns xx component.

Returns

| TItem | xx component |
8.16.4.23  xy()

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

Returns xy component.

Returns

   TItem xy component

8.16.4.24  yx()

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

Returns yx component.

Returns

   TItem yx component

8.16.4.25  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:

   • edgegeom.h

8.17  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 colour 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**

**8.17.1 Detailed Description**

Class for tracking the graphics state leading to the point where a transform is applied.

Consider for example the IImageDownsamplerTransform described later in this header. In order to determine how to downsample an image, the transform needs to know how large the image will eventually be. The CTransformState provides this information by providing the combined transform that applies to the image based on the RenderTransforms of all the nodes entered leading to the point where the image is actually encountered.

Other transforms need access to other information, such as the approximate clip area, the current group color space, the renderingIntent (if present), the current antialiasing mode (edge mode) and/or how a brush is used.

**8.17.2 Member Function Documentation**
8.17.2.1  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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>brush</td>
<td>The brush whose state we wish to apply.</td>
</tr>
<tr>
<td>brushUsage</td>
<td>The way the brush would be used; ie for filling, stroking, or for use as an opacity mask.</td>
</tr>
</tbody>
</table>

Returns

The new state.

8.17.2.2  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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>node</td>
<td>The node whose state we wish to apply.</td>
</tr>
<tr>
<td>updateTransform</td>
<td>If true, update the transform entry, and also the related black point compensation entry.</td>
</tr>
<tr>
<td>updateClip</td>
<td>If true, update the clipBounds entry</td>
</tr>
<tr>
<td>updateRenderingIntent</td>
<td>If true, update the rendering intent.</td>
</tr>
</tbody>
</table>

Returns

The new state.

8.17.3  Member Data Documentation

8.17.3.1  inUncoloredTilingBrush

bool JawsMako::CTransformState::inUncoloredTilingBrush

Set to true when we descend into an uncolored tiling brush (An IDOMTilingPatternBrush with PaintType == 2)
8.17.3.2 transformPriv

```cpp
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

8.18 JawsMako::IAnnotationUtils::CXMLResource Class Reference

Simple class for tracking streams associated with XML generated by `generateXMLForDocument()`

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

8.18.1 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

8.19 IDOMTransparencyGroup::Data Class Reference

Initialization data.

```cpp
#include <idomgroup.h>
```

Inheritance diagram for IDOMTransparencyGroup::Data:
8.19.1 Detailed Description

Initialization data.

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

- idomgroup.h

8.20 IDOMImage::Data Class Reference

Initialization data.

#include <idomimageresource.h>

Inheritance diagram for IDOMImage::Data:

8.20.1 Detailed Description

Initialization data.

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

- idomimageresource.h

Generated by Doxygen
8.21 IDOMRawImage::Data Class Reference

Initialization data.

#include <idomimageresource.h>

Inheritance diagram for IDOMRawImage::Data:

![Inheritance Diagram]

8.21.1 Detailed Description

Initialization data.

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

- idomimageresource.h

8.22 IDOMPDFImage::Data Class Reference

Initialization data.

#include <idomimageresource.h>
Inheritance diagram for IDOMPDFImage::Data:

8.22.1 Detailed Description

Initialization data.

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

- idomimageresource.h

8.23 IDOMPCLImage::Data Class Reference

Initialization data.

#include <idomimageresource.h>

Inheritance diagram for IDOMPCLImage::Data:
8.23.1 Detailed Description

Initialization data.

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

- idomimageresource.h

8.24 IDOMRecombineImage::Data Class Reference

Initialization data.

#include <idomimageresource.h>

Inheritance diagram for IDOMRecombineImage::Data:

8.24.1 Detailed Description

Initialization data.

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

- idomimageresource.h
8.25 IDOMRecombineAlphaImage::Data Class Reference

Initialization data.
#include <idomimageresource.h>

Inheritance diagram for IDOMRecombineAlphaImage::Data:

8.25.1 Detailed Description

Initialization data.
The documentation for this class was generated from the following file:
- idomimageresource.h

8.26 IEDLTime::Data Class Reference

Initialization data.
#include <edltime.h>

Inheritance diagram for IEDLTime::Data:
8.26.1 Detailed Description

Initialization data.

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

- edtime.h

8.27 IDOMCompositeImage::Data Class Reference

Initialization data.

#include <idomimageresource.h>

Inheritance diagram for IDOMCompositeImage::Data:

```
Class Diagram

CClassParams
   |
   v
IDOMImage::Data
   |
   v
IDOMCompositeImage::Data
```

8.27.1 Detailed Description

Initialization data.

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

- idomimageresource.h
8.28 IDOMImageChannelSelectorFilter::Data Class Reference

Initialization data.
#include <idomimageresource.h>

Inheritance diagram for IDOMImageChannelSelectorFilter::Data:

8.28.1 Detailed Description

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

• idomimageresource.h

8.29 IDOMImageColorSpaceSubstitutionFilter::Data Class Reference

Initialization data.
#include <idomimageresource.h>

Inheritance diagram for IDOMImageColorSpaceSubstitutionFilter::Data:
8.29.1 Detailed Description

Initialization data.

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

- idomimageresource.h

8.30 IDOMGradientStop::Data Class Reference

Initialization data.

#include <idombrush.h>

Inheritance diagram for IDOMGradientStop::Data:

```
+---+      +---+
|   |      |   |
|   |  CClassParams  |   |
|   |      |   |   |
|   |      +---+   |
|   |       |   |   |
|   |       |  IDOMGradientStop::Data |
```

8.30.1 Detailed Description

Initialization data.

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

- idombrush.h
8.31 IDOMImageColorConverterFilter::Data Class Reference

Initialization data.
#include <idomimageresource.h>

Inheritance diagram for IDOMImageColorConverterFilter::Data:

8.31.1 Detailed Description

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

- idomimageresource.h

8.32 IDOMImageBleederFilter::Data Class Reference

Initialization data.
#include <idomimageresource.h>

Inheritance diagram for IDOMImageBleederFilter::Data:
8.32.1 Detailed Description

Initialization data.

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

- idomimageresource.h

8.33 IDOMImageDownsamplerFilter::Data Class Reference

Initialization data.

#include <idomimageresource.h>

Inheritance diagram for IDOMImageDownsamplerFilter::Data:

![Inheritance Diagram]

8.33.1 Detailed Description

Initialization data.

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

- idomimageresource.h
8.34 IDOMImageMaskExpanderFilter::Data Class Reference

Initialization data.

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

Inheritance diagram for IDOMImageMaskExpanderFilter::Data:

![Inheritance diagram for IDOMImageMaskExpanderFilter::Data]

### 8.34.1 Detailed Description

Initialization data.

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

- idomimageresource.h

8.35 IDOMSolidColorBrush::Data Class Reference

Initialization data.

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

Inheritance diagram for IDOMSolidColorBrush::Data:

![Inheritance diagram for IDOMSolidColorBrush::Data]
8.35.1 Detailed Description

Initialization data.

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

- idombrush.h

8.36 IDOMImageDeindexerFilter::Data Class Reference

Initialization data.

#include <idomimageresource.h>

Inheritance diagram for IDOMImageDeindexerFilter::Data:

8.36.1 Detailed Description

Initialization data.

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

- idomimageresource.h
Initialization data.
#include <idomimageresource.h>

Inheritance diagram for IDOMImageDeviceNToBaseFilter::Data:

```
CClassParams

IDOMImageDeviceNToBaseFilter::Data
```

8.37.1 Detailed Description

Initialization data.
The documentation for this class was generated from the following file:
- idomimageresource.h

8.38 IDOMLinearGradientBrush::Data Class Reference

Initialization data.
#include <idombrush.h>

Inheritance diagram for IDOMLinearGradientBrush::Data:

```
CClassParams

IDOMLinearGradientBrush::Data
```
8.38.1 Detailed Description

Initialization data.

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

- idombrush.h

8.39 IDOMImageInverterFilter::Data Class Reference

Initialization data.

#include <idomimageresource.h>

Inheritance diagram for IDOMImageInverterFilter::Data:

8.39.1 Detailed Description

Initialization data.

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

- idomimageresource.h
8.40  IDOMDePremultiplierFilter::Data Class Reference

Initialization data.
#include <idomimageresource.h>

Inheritance diagram for IDOMDePremultiplierFilter::Data:

```
CClassParams

IDOMDePremultiplierFilter ::Data
```

8.40.1  Detailed Description

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

- idomimageresource.h

8.41  IDOMRadialGradientBrush::Data Class Reference

Initialization data.
#include <idombrush.h>

Inheritance diagram for IDOMRadialGradientBrush::Data:

```
CClassParams

IDOMRadialGradientBrush ::Data
```
8.41.1 Detailed Description

Initialization data.

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

- idombrush.h

8.42 IDOMImageMatteRemoverFilter::Data Class Reference

Initialization data.

#include <idomimageresource.h>

Inheritance diagram for IDOMImageMatteRemoverFilter::Data:

8.42.1 Detailed Description

Initialization data.

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

- idomimageresource.h
8.43  IDOMImageBitScalerFilter::Data Class Reference

Initialization data.

#include <idomimageresource.h>

Inheritance diagram for IDOMImageBitScalerFilter::Data:

8.43.1  Detailed Description

Initialization data.

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

• idomimageresource.h

8.44  IDOMImageBrush::Data Class Reference

Initialization data.

#include <idombrush.h>

Inheritance diagram for IDOMImageBrush::Data:
8.44.1 Detailed Description

Initialization data.

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

- idombrush.h

8.45 IDOMImageColorKeyFilter::Data Class Reference

Initialization data.

#include <idomimageresource.h>

Inheritance diagram for IDOMImageColorKeyFilter::Data:

```
+------------------
|                  |
|                  |
```

8.45.1 Detailed Description

Initialization data.

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

- idomimageresource.h

Generated by Doxygen
8.46 IDOMImageDecodeFilter::Data Class Reference

Initialization data.
#include <idomimageresource.h>

Inheritance diagram for IDOMImageDecodeFilter::Data:

8.46.1 Detailed Description

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

- idomimageresource.h

8.47 IDOMMaskedBrush::Data Class Reference

Initialization data.
#include <idombrush.h>

Inheritance diagram for IDOMMaskedBrush::Data:
8.47.1 Detailed Description

Initialization data.

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

- idombrush.h

8.48 IDOMFilteredImage::Data Class Reference

Initialization data.

#include <idomimageresource.h>

Inheritance diagram for IDOMFilteredImage::Data:

8.48.1 Detailed Description

Initialization data.

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

- idomimageresource.h
Initialization data.

#include <idomjobtk.h>

Inheritance diagram for IDOMJobTkGenericNode::Data:

8.49.1 Detailed Description

Initialization data.

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

- idomjobtk.h

8.50 IDOMVisualBrush::Data Class Reference

Initialization data.

#include <idombrush.h>

Inheritance diagram for IDOMVisualBrush::Data:
8.50.1 Detailed Description

Initialization data.

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

- idombrush.h

8.51 IDOMJobTkGenericCharacterData::Data Class Reference

Initialization data.

#include <idomjobtk.h>

Inheritance diagram for IDOMJobTkGenericCharacterData::Data:

```
CClassParams

IDOMJobTkGenericCharacterData::Data
```

8.51.1 Detailed Description

Initialization data.

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

- idomjobtk.h
8.52 IDOMJobTkNode::Data Class Reference

Initialization data.

```
#include <idomjobtk.h>
```

Inheritance diagram for IDOMJobTkNode::Data:

![Inheritance diagram](image)

Public Attributes

- **EDLQName name**
  
  Name of the node.
- **eDOMJobTkNodeType jobTkNodeType**
  
  Job Ticket node type.

8.52.1 Detailed Description

Initialization data.

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

- *idomjobtk.h*

8.53 IDOMSoftMaskBrush::Data Class Reference

Initialization data.

```
#include <idombrush.h>
```
Inheritance diagram for IDOMSoftMaskBrush::Data:

8.53.1 Detailed Description

Initialization data.

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

- idombrush.h

8.54 IDOMJobTkValue::Data Class Reference

Initialization data.

#include <idomjobtk.h>

Inheritance diagram for IDOMJobTkValue::Data:

8.54.1 Detailed Description

Initialization data.

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

- idomjobtk.h
8.55  IDOMJobTkContent::Data Class Reference

Initialization data.

#include <idomjobtk.h>

Inheritance diagram for IDOMJobTkContent::Data:

```
ClassDiagram
  IDOMJobTkContent::Data
  CClassParams

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.
```

8.55.1  Detailed Description

Initialization data.

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

• idomjobtk.h

8.56  IDOMTilingPatternBrush::Data Class Reference

Initialization data.

#include <idombrush.h>
Inheritance diagram for IDOMTilingPatternBrush::Data:

```plaintext
<table>
<thead>
<tr>
<th>CClassParams</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDOMTilingPatternBrush::Data</td>
</tr>
</tbody>
</table>
```

8.56.1 Detailed Description

Initialization data.

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

- `idombrush.h`

8.57 IDOMShadingPatternType1Brush::Data Class Reference

Initialization data.

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

Inheritance diagram for IDOMShadingPatternType1Brush::Data:

```plaintext
<table>
<thead>
<tr>
<th>CClassParams</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDOMShadingPatternType1Brush::Data</td>
</tr>
</tbody>
</table>
```
Initialization data.

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

- idombrush.h

8.58  IDOMOutlineEntry::Data Class Reference

Initialization data.

#include <idomoutline.h>

Inheritance diagram for IDOMOutlineEntry::Data:

```
#include <idomoutline.h>
```

8.58.1  Detailed Description

Initialization data.

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

- idomoutline.h
8.59 IDOMShadingPatternType2Brush::Data Class Reference

Initialization data.
#include <idombrush.h>

Inheritance diagram for IDOMShadingPatternType2Brush::Data:

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

8.59.1 Detailed Description

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

- idombrush.h

8.60 IDOMShadingPatternType3Brush::Data Class Reference

Initialization data.
#include <idombrush.h>

Inheritance diagram for IDOMShadingPatternType3Brush::Data:

```
#include <idombrush.h>
```
8.61 IDOMOutline::Data Class Reference

Initialization data.

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

- idombrush.h

8.61.1 Detailed Description

Initialization data.

#include <idomoutline.h>

Inheritance diagram for IDOMOutline::Data:

[Diagram of inheritance]

8.61.1 Detailed Description

Initialization data.

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

- idomoutline.h

Generated by Doxygen
8.62  IDOMFixedPage::Data Class Reference

Initialization data.

```c
#include <idompage.h>
```

Inheritance diagram for IDOMFixedPage::Data:

![Inheritance diagram for IDOMFixedPage::Data](image)

8.62.1  Detailed Description

Initialization data.

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

- `idompage.h`

8.63  IDOMPathNode::Data Class Reference

Initialization data.

```c
#include <idompath.h>
```

Inheritance diagram for IDOMPathNode::Data:

![Inheritance diagram for IDOMPathNode::Data](image)

Generated by Doxygen
8.63.1 Detailed Description

Initialization data.

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

- idompath.h

8.64 IDOMShadingPatternType4567Brush::Data Class Reference

Initialization data.

#include <idombrush.h>

Inheritance diagram for IDOMShadingPatternType4567Brush::Data:

```
CClassParams

IDOMShadingPatternType4567Brush::Data
```

8.64.1 Detailed Description

Initialization data.

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

- idombrush.h
8.65  IDOMArcSegment::Data Class Reference

Initialization data.

#include <idompathgeometry.h>

Inheritance diagram for IDOMArcSegment::Data:

```
CClassParams

IDOMArcSegment::Data
```

8.65.1  Detailed Description

Initialization data.

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

- idompathgeometry.h

8.66  IDOMPolyLineSegment::Data Class Reference

Initialization data.

#include <idompathgeometry.h>

Inheritance diagram for IDOMPolyLineSegment::Data:

```
CClassParams

IDOMPolyLineSegment::Data
```
8.67 IDOMNullBrush::Data Class Reference

Initialization data.

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

- idompathgeometry.h

8.67.1 Detailed Description

Initialization data.

#include <idombrush.h>

Inheritance diagram for IDOMNullBrush::Data:

```
  +-----------------+          +-----------------+
  |                 |          |                 |
  +-----------------+          +-----------------+
    |                 |          |                 |
  +-----------------+          +-----------------+
    |                 |          |                 |
  +-----------------+          +-----------------+
                CClassParams
                ↑          ↑
                ↓          ↓
IDOMNullBrush::Data
```

8.67.1 Detailed Description

Initialization data.

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

- idombrush.h
8.68 IDOMPolyBezierSegment::Data Class Reference

Initialization data.
#include <idompathgeometry.h>

Inheritance diagram for IDOMPolyBezierSegment::Data:

```
  CClassParams
   
IDOMPolyBezierSegment
::Data
```

8.68.1 Detailed Description

Initialization data.

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

- idompathgeometry.h

8.69 IDOMPolyQuadraticBezierSegment::Data Class Reference

Initialization data.
#include <idompathgeometry.h>

Inheritance diagram for IDOMPolyQuadraticBezierSegment::Data:

```
  CClassParams
   
IDOMPolyQuadraticBezierSegment
::Data
```
8.69.1 Detailed Description

Initialization data.

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

- idompathgeometry.h

8.70 IDOMCanvas::Data Class Reference

Initialization data.

#include <idomcanvas.h>

Inheritance diagram for IDOMCanvas::Data:

8.70.1 Detailed Description

Initialization data.

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

- idomcanvas.h
8.71 IDOMCharPathGroup::Data Class Reference

Initialization data.

#include <idomcharpathgroup.h>

Inheritance diagram for IDOMCharPathGroup::Data:

```
   +------------------
   |                  |
   |   CClassParams   |
   +------------------
       |               |
       |               |
       v               
+------------------
|                  |
|   IDOMCharPathGroup::Data |
|                  |
+------------------
```

8.71.1 Detailed Description

Initialization data.

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

- idomcharpathgroup.h

8.72 IDOMPathFigure::Data Class Reference

Initialization data.

#include <idompathgeometry.h>

Inheritance diagram for IDOMPathFigure::Data:

```
   +------------------
   |                  |
   |   CClassParams   |
   +------------------
       |               |
       |               |
       v               
+------------------
|                  |
|   IDOMPathFigure::Data |
|                  |
+------------------
```
8.72.1 Detailed Description

Initialization data.

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

- idompathgeometry.h

8.73 IDOMPathGeometry::Data Class Reference

Initialization data.

#include <idompathgeometry.h>

Inheritance diagram for IDOMPathGeometry::Data:

```
  CClassParams
  IDOMPathGeometry::Data
```

8.73.1 Detailed Description

Initialization data.

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

- idompathgeometry.h
8.74  IDOMICCProfile::Data Class Reference

Initialization data.
#include <idomresources.h>

Inheritance diagram for IDOMICCProfile::Data:

![Inheritance Diagram]

8.74.1  Detailed Description

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

- idomresources.h

8.75  IDOMPrintTicket::Data Class Reference

Initialization data.
#include <idomresources.h>

Inheritance diagram for IDOMPrintTicket::Data:

![Inheritance Diagram]
8.75.1 Detailed Description

Initialization data.

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

- idomresources.h

8.76 IDOMAudioFile::Data Class Reference

Initialization data.

#include <idomresources.h>

Inheritance diagram for IDOMAudioFile::Data:

```
+-----------------+
<p>| CClassParams    |
|                 |
|     ↑           |
|     |           |</p>
<table>
<thead>
<tr>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDOMAudioFile::Data</td>
</tr>
</tbody>
</table>
```

8.76.1 Detailed Description

Initialization data.

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

- idomresources.h
8.77 IDOMRawDataFile::Data Class Reference

Initialization data.

#include <idomresources.h>

Inheritance diagram for IDOMRawDataFile::Data:

```
CClassParams

IDOMRawDataFile::Data
```

8.77.1 Detailed Description

Initialization data.

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

- idomresources.h

8.78 IDOMResourceDictionary::Data Class Reference

Initialization data.

#include <idomresources.h>

Inheritance diagram for IDOMResourceDictionary::Data:

```
CClassParams

IDOMResourceDictionary::Data
```
8.79 IDOMMatrix::Data Class Reference

Initialization data.

#include <idomresources.h>

Inheritance diagram for IDOMMatrix::Data:

```
CClassParams
   `-- IDOMMatrix::Data
```

8.79.1 Detailed Description

Initialization data.

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

- idomresources.h
8.80  IDOMColorSpaceICCBased::Data Class Reference

Initialization data.

```
#include <idomcolorspace.h>
```

Inheritance diagram for IDOMColorSpaceICCBased::Data:

```
CClassParams
  └── IDOMColorSpaceICCBased::Data
```

8.80.1  Detailed Description

Initialization data.

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

- `idomcolorspace.h`

8.81  IDOMShape::Data Class Reference

Initialization data.

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

Inheritance diagram for IDOMShape::Data:

```
CClassParams
  └── IDOMShape::Data
```
8.81.1 Detailed Description

Initialization data.

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

- idomshape.h

8.82 IDOMDeviceNColorant::Data Class Reference

Initialization data.

#include <idomcolorspace.h>

Inheritance diagram for IDOMDeviceNColorant::Data:

![Inheritance Diagram]

8.82.1 Detailed Description

Initialization data.

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

- idomcolorspace.h
### 8.83 IEDLTempStore::Data Class Reference

Initialization data.

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

Inheritance diagram for IEDLTempStore::Data:

![Inheritance Diagram](edi20.png)

#### 8.83.1 Detailed Description

Initialization data.

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

- `iedltempstore.h`

### 8.84 IDOMColorSpaceIndexed::Data Class Reference

Initialization data.

```cpp
#include <idomcolorspace.h>
```

Inheritance diagram for IDOMColorSpaceIndexed::Data:

![Inheritance Diagram](edi21.png)
8.84.1 Detailed Description

Initialization data.

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

- idomcolorspace.h

8.85 IDOMColorSpaceDeviceN::Data Class Reference

Initialization data.

#include <idomcolorspace.h>

Inheritance diagram for IDOMColorSpaceDeviceN::Data:

![Inheritance Diagram]

8.85.1 Detailed Description

Initialization data.

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

- idomcolorspace.h

Generated by Doxygen
8.86  IDOMColorSpaceLAB::Data Class Reference

Initialization data.

#include <idomcolorsplace.h>

Inheritance diagram for IDOMColorSpaceLAB::Data:

![Inheritance Diagram](image)

8.86.1  Detailed Description

Initialization data.

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

- idomcolorsplace.h

8.87  IDOMFontSource::Data Class Reference

Initialization data.

#include <idomfont.h>
Inheritance diagram for IDOMFontSource::Data:

8.87.1 Detailed Description

Initialization data.

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

- idomfont.h

8.88 IDOMFontSourceStreamFilter::Data Class Reference

Initialization data.

#include <idomfont.h>
Inheritance diagram for IDOMFontSourceStreamFilter::Data:

8.88.1 Detailed Description

Initialization data.

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

- idomfont.h

8.89 IDOMFontSourceFromStream::Data Class Reference

Initialization data.

#include <idomfont.h>
Inheritance diagram for IDOMFontSourceFromStream::Data:

```plaintext
CClassParams

IDOMFontSource::Data

IDOMFontSourceStreamFilter::Data

IDOMFontSourceFromStream::Data
```

8.89.1 Detailed Description

Initialization data.

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

- idomfont.h

8.90 IDOMFontSourceObfuscationConverter::Data Class Reference

Initialization data.

```c
#include <idomfont.h>
```
Inheritance diagram for IDOMFontSourceObfuscationConverter::Data:

```
inheritance diagram:

  CClassParams
  IDOMFontSource::Data
  IDOMFontSourceObfuscationConverter::Data
```

### Public Attributes

- **IDOMFontSourcePtr inputFontSource**
  
  *The input font source for the converter.*

- **eOperation operation**
  
  *The obfuscation or deobfuscation operation.*

### 8.90.1 Detailed Description

Initialization data.

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

- **idomfont.h**

### 8.91 IDOMFont::Data Class Reference

Initialization data.

```
#include <idomfont.h>
```
Inheritance diagram for IDOMFont::Data:

![Inheritance Diagram]

8.91.1 Detailed Description

Initialization data.

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

- idomfont.h

8.92 IDOMFontOpenType::Data Class Reference

Initialization data.

#include <idomfont.h>
Inheritance diagram for IDOMFontOpenType::Data:

8.92.1 Detailed Description

Initialization data.

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

- idomfont.h

8.93 IDOMFontOpenTypeTT::Data Class Reference

Initialization data.

#include <idomfont.h>
8.93.1 Detailed Description

Initialization data.

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

- idomfont.h

8.94 IDOMType3Font::Data Class Reference

Initialization data.

#include <idomfont.h>
Inheritance diagram for IDOMType3Font::Data:

8.94.1 Detailed Description

Initialization data.

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

- idomfont.h

8.95 IDOMFontPCLXL::Data Class Reference

Initialization data.

#include <idomfont.h>
8.95.1 Detailed Description

Initialization data.

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

- idomfont.h

8.96 IDOMFontPCLXL::Data Class Reference

Initialization data.

```cpp
#include <idomfont.h>
```
Inheritance diagram for IDOMFontPCL5::Data:

8.96.1 Detailed Description

Initialization data.

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

- idomfont.h

8.97 IEDLNamespace::Data Class Reference

Initialization data.

#include <edlqname.h>
Inheritance diagram for IEDLNamespace::Data:

8.97.1 Detailed Description

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

- ediqlname.h

8.98 IDOMForm::Data Class Reference

Initialization data.
#include <idomform.h>
Inheritance diagram for IDOMForm::Data:

8.98.1 Detailed Description

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

- idomform.h
8.99 IDOMFormInstance::Data Class Reference

Initialization data.

#include <idomform.h>

Inheritance diagram for IDOMFormInstance::Data:

```
CTrace
`.
```

8.99.1 Detailed Description

Initialization data.

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

- idomform.h

8.100 IDOMSampledFunction::Data Class Reference

Initialization data.

#include <idomfunction.h>

Inheritance diagram for IDOMSampledFunction::Data:

```
CTrace
`.
```

Generated by Doxygen
Initialization data.

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

- idomfunction.h

8.101 IDOMExponentialFunction::Data Class Reference

Initialization data.

#include <idomfunction.h>

Inheritance diagram for IDOMExponentialFunction::Data:

```
+------------------+
| CClassParams     |
|                  |
+------------------+
| IDOMExponentialFunction::Data |
|                  |
+------------------+
```

8.101.1 Detailed Description

Initialization data.

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

- idomfunction.h
8.102  IDOMStitchingFunction::Data Class Reference

Initialization data.
#include <idomfunction.h>

Inheritance diagram for IDOMStitchingFunction::Data:

```
     CClassParams
       |
     IDOMStitchingFunction
       |
     ::Data
```

8.102.1  Detailed Description

Initialization data.

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

- idomfunction.h

8.103  IDOMGroupingFunction::Data Class Reference

Initialization data.
#include <idomfunction.h>

Inheritance diagram for IDOMGroupingFunction::Data:

```
     CClassParams
       |
     IDOMGroupingFunction
       |
     ::Data
```
8.103.1 Detailed Description

Initialization data.

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

- idomfunction.h

8.104 IDOMPostScriptCalculatorFunction::Data Class Reference

Initialization data.

```
#include <idomfunction.h>
```

Inheritance diagram for IDOMPostScriptCalculatorFunction::Data:

```
CClassParams

IDOMPostScriptCalculatorFunction::Data
```

8.104.1 Detailed Description

Initialization data.

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

- idomfunction.h
8.105  IDOMGlyph::Data Class Reference

Initialization data.

#include <idomglyph.h>

Inheritance diagram for IDOMGlyph::Data:

8.105.1  Detailed Description

Initialization data.

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

- idomglyph.h

8.106  IDOMGlyphs::Data Class Reference

Initialization data.

#include <idomglyphs.h>

Inheritance diagram for IDOMGlyphs::Data:
8.106.1 Detailed Description

Initialization data.

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

- `idomglyphs.h`

8.107 IDOMGroup::Data Class Reference

Initialization data.

```c
#include <idomgroup.h>
```

Inheritance diagram for IDOMGroup::Data:

```
CClassParams

IDOMGroup::Data
```

8.107.1 Detailed Description

Initialization data.

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

- `idomgroup.h`
8.108  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

8.108.1  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

8.109  EDLIFStream Class Reference

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

#include <edlstream.h>

Inherits ifstream.
8.109.1 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:

- edistream.h

8.110 EDLOFStream Class Reference

An ofstream that can deal with UTF8 file names on all platforms.
#include <edlstream.h>
Inherits ofstream.

8.110.1 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:

- edistream.h

8.111 EDLQName Class Reference

Implementation of qualified name class.
#include <edlqname.h>

Public Member Functions

- **EDLQName ()**
  Empty 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 setName (const IEDLNamespacePtr &ns)**
  Sets namespace.
- **EDLString getName () const**
  Returns member name of EDLQName.
- **bool setName (const EDLString &name)**
  Sets member name of EDLQName.
- **EDLSysString getSysName () const**
  Returns 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)
8.111.1 Detailed Description

Implementation of qualified name class.

8.111.2 Constructor & Destructor Documentation

8.111.2.1 EDLQName()

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

constructor of EDLQName

Parameters

<table>
<thead>
<tr>
<th>ptrNamespace</th>
<th>Smart pointer to the namespace interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name value</td>
</tr>
</tbody>
</table>

8.111.2.2 EDLQName()

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

constructor of EDLQName

Parameters

<table>
<thead>
<tr>
<th>ptrNamespace</th>
<th>Smart pointer to the namespace interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name value</td>
</tr>
</tbody>
</table>

8.111.3 Member Function Documentation

8.111.3.1 getName()

EDLString EDLQName::getName ( ) const [inline]   

Retrieves member name of EDLQName.
8.111.3.2 getNamespace()

IEDLNamespacePtr EDLQName::getNamespace ( ) const [inline]

Retrieves namespace.

Returns

IEDLNamespacePtr Returns smart pointer to the namespace interface

8.111.3.3 getNameWithPrefix()

EDLString EDLQName::getNameWithPrefix ( ) const [inline]

Returns EDLString that is combination of prefix and name (prefix:name)

Returns

EDLString Returns fullname

8.111.3.4 getSysName()

EDLSysString EDLQName::getSysName ( ) const [inline]

Retrieves member name of EDLQName.

Returns

EDLSysString Returns name value as EDLSysString

8.111.3.5 operator=()

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

operator =
### Parameters

| another | EDLQName |

---

### 8.111.3.6 operator==()

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

**operator ==**

**Parameters**

| another | EDLQName |

**Returns**

```cpp
bool True if equal, false if not
```

---

### 8.111.3.7 setName()[1/2]

```cpp
bool EDLQName::setName (const EDLString & name) [inline]
```

**Sets member name of EDLQName.**

**Parameters**

| name | The name value |

**Returns**

```cpp
bool Returns true
```

---

### 8.111.3.8 setName()[2/2]

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

**Sets member name of EDLQName.**

---

---

*Generated by Doxygen*
Parameters

- **sysname**: The name value

Returns

- bool Returns true

### 8.111.3.9 setNamespace()

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

Sets namespace.

Parameters

- **ns**: Smart pointer to the namespace interface

Returns

- bool Returns true

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

- edlqname.h

---

### 8.112 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.

```cpp
#include <idomimageresource.h>
```
Inheritance diagram for IDMPDFImage::FlateLZWParams:

```
  IRCObject
  IDMPDFImage::IDecodeParams
      IDMPDFImage::FlateLZWParams
```

Additional Inherited Members

8.112.1 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

8.113 JawsMako::IAnnotation Class Reference

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

```c++
#include <interactive.h>
```
Inheritance diagram for JawsMako::IAnnotation:

Public Types

- enum eAnnotationType {

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.
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 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.

Additional Inherited Members

8.113.1 Detailed Description

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

8.113.2 Member Enumeration Documentation

8.113.2.1 eAnnotationType

defined inside JawsMako::IAnnotation

\[
\text{enum JawsMako::IAnnotation::eAnnotationType}
\]

Types of annotations, listed with the earliest version of PDF that supported them.
### Enumerator

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eAT3D</td>
<td>(PDF 1.6) 3D annotation</td>
</tr>
<tr>
<td>eATCaret</td>
<td>(PDF 1.5) Caret annotation</td>
</tr>
<tr>
<td>eATCircle</td>
<td>(PDF 1.3) Circle annotation</td>
</tr>
<tr>
<td>eATFileAttachment</td>
<td>(PDF 1.3) File attachment annotation</td>
</tr>
<tr>
<td>eATFreeText</td>
<td>(PDF 1.3) Free text annotation</td>
</tr>
<tr>
<td>eATHighlight</td>
<td>(PDF 1.3) Highlight annotation</td>
</tr>
<tr>
<td>eATInk</td>
<td>(PDF 1.3) Ink annotation</td>
</tr>
<tr>
<td>eATLine</td>
<td>(PDF 1.3) Line annotation</td>
</tr>
<tr>
<td>eATLink</td>
<td>(PDF 1.2) Link annotation</td>
</tr>
<tr>
<td>eATMovie</td>
<td>(PDF 1.2) Movie annotation</td>
</tr>
<tr>
<td>eATPolygon</td>
<td>(PDF 1.5) Polygon annotation</td>
</tr>
<tr>
<td>eATPolyLine</td>
<td>(PDF 1.5) PolyLine annotation</td>
</tr>
<tr>
<td>eATPopup</td>
<td>(PDF 1.3) Popup annotation</td>
</tr>
<tr>
<td>eATPrinterMark</td>
<td>(PDF 1.4) Printer’s mark annotation</td>
</tr>
<tr>
<td>eATProjection</td>
<td>(PDF 1.7) Projection annotation</td>
</tr>
<tr>
<td>eATRedact</td>
<td>(PDF 1.7) Redact annotation</td>
</tr>
<tr>
<td>eATRichMedia</td>
<td>(PDF 1.7) RichMedia annotation</td>
</tr>
<tr>
<td>eATScreen</td>
<td>(PDF 1.5) Screen annotation</td>
</tr>
<tr>
<td>eATSound</td>
<td>(PDF 1.2) Sound annotation</td>
</tr>
<tr>
<td>eATSquare</td>
<td>(PDF 1.3) Square annotation</td>
</tr>
<tr>
<td>eATSquiggly</td>
<td>(PDF 1.4) Squiggly underline annotation</td>
</tr>
<tr>
<td>eATStamp</td>
<td>(PDF 1.3) Stamp annotation</td>
</tr>
<tr>
<td>eATStrikeOut</td>
<td>(PDF 1.3) Strikeout annotation</td>
</tr>
<tr>
<td>eATText</td>
<td>(PDF 1.2) Text annotation</td>
</tr>
<tr>
<td>eATTrapNet</td>
<td>(PDF 1.3) Trap network annotation</td>
</tr>
<tr>
<td>eATUnderline</td>
<td>(PDF 1.3) Underline annotation</td>
</tr>
<tr>
<td>eATWatermark</td>
<td>(PDF 1.6) Watermark annotation</td>
</tr>
<tr>
<td>eATWidget</td>
<td>(PDF 1.2) Widget annotation</td>
</tr>
<tr>
<td>eATOther</td>
<td>Other annotation.</td>
</tr>
</tbody>
</table>

### 8.113.3 Member Function Documentation

#### 8.113.3.1 addAppearance()

```cpp
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.
8.113.3.2 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

8.113.3.3 getAppearance()

virtual IAnnotationAppearancePtr JawsMako::IAnnotation::getAppearance (  
eAppearanceUsage usage,  
const U8String & state = U8String() ) const [pure virtual]

Fetch the annotation appearance that should be used for the given annotation usage and state according to the following rules:

- If an exact match is found, it will be returned.
- If an exact match is not found, but there exists a Normal appearance with the same state, it will be returned.
- If none of the above yield an appearance, an exception results.

Parameters  
| usage | The annotation usage |
| state | The annotation state |

Returns  
IAnnotationAppearancePtr A smart pointer to the annotation appearance object

8.113.3.4 getAppearances()

virtual CAnnotationAppearanceVect JawsMako::IAnnotation::getAppearances ( ) const [pure virtual]

Return all the annotation appearances in a vector.
Returns

**CAnnotationAppearanceVect** The vector of annotation appearances

---

### 8.113.3.5 getBorder()

```cpp
virtual CAnnotationBorder JawsMako::IAnnotation::getBorder ( ) const [pure virtual]
```

Get the annotation's border. See **CAnnotationBorder** for details.

Returns

**CAnnotationBorder** The annotation's border

---

### 8.113.3.6 getColor()

```cpp
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

---

### 8.113.3.7 getContents()

```cpp
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.

---

### 8.113.3.8 getFlags()

```cpp
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
8.113.9 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.

8.113.10 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

8.113.11 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

8.113.12 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
### 8.113.3.13 getType()

```cpp
class JawsMako::IAnnotation
{
public:

    virtual eAnnotationType getType() const [pure virtual]
```

Get the type of the annotation.

Returns

- **eAnnotationType** The annotation type

### 8.113.3.14 matchesReference()

```cpp
class JawsMako::IAnnotation
{
public:

    virtual bool matchesReference(const IAnnotationReferencePtr & reference) const [pure virtual]
```

Does this annotation match the given `IAnnotationReference`?

Parameters

- **reference** The annotation reference to be matched

Returns

- **bool** True if a match is found, otherwise false

### 8.113.3.15 rotate()

```cpp
class JawsMako::IAnnotation
{
public:

    virtual void rotate(uint16 rotate, double pageWidth, double pageHeight) [pure virtual]
```

Rotate the annotation clockwise as if the page was rotated by the same amount.

NB If an annotation has the NoRotate (bit 4) flag set, then the annotation will not be rotated, merely repositioned accordingly.

Parameters

- **rotate** The angle to rotate. Must be 0, 90, 180 or 270 degrees.
- **pageWidth** The width of the parent page.
- **pageHeight** The height of the parent page.
8.113.3.16 setBorder()

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

Set the annotation's border.

Parameters

| border    | The border. See CAnnotationBorder for details as not all attributes will be supported for all annotation types and all versions of PDF. |

8.113.3.17 setColor()

virtual void JawsMako::IAnnotation::setColor (  
    const IDOMColorPtr & color ) [pure virtual]

Set the color. The use of this color depends on the annotation type. See the PDF 1.7 specification for details.

Parameters

| color     | The required color. Setting a NULL value will remove any existing color. If a color is provided it must use a DeviceRGB, DeviceGray or DeviceCMYK color space. |

8.113.3.18 setContents()

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

Set the Contents entry in UTF-8.

Parameters

| contents  | The Contents entry |

8.113.3.19 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.
8.113.3.20 setModificationTime()

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

Set the Modification date and time of the annotation.

Parameters

| modificationTime | the Modification date and time. Must not be NULL. |

8.113.3.21 setRect()

virtual void JawsMako::IAnnotation::setRect ( const FRect & rect ) [pure virtual]

Set the rect in which the appearances should be displayed.

Parameters

| rect | The display rect, which must be non-empty |

8.113.3.22 setState()

virtual void JawsMako::IAnnotation::setState ( const U8String & state ) [pure virtual]

Set the current annotation state.

Parameters

| state | The annotation state, or an empty string if the state should be removed |

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

- interactive.h
8.114 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:

```
IRCOBJECT
```

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.

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

8.114.1 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.
8.114.2 Member Function Documentation

8.114.2.1 clone()

```cpp
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

8.114.2.2 create()

```cpp
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>jawsMako</td>
<td>The JawsMako instance</td>
</tr>
<tr>
<td>form</td>
<td>The graphical contents as an IDOMForm</td>
</tr>
<tr>
<td>usage</td>
<td>The usage for this appearance</td>
</tr>
<tr>
<td>state</td>
<td>The appearance state for this appearance</td>
</tr>
</tbody>
</table>

Returns

IAnnotationAppearancePtr A smart pointer to the new annotation appearance

8.114.2.3 getForm()

```cpp
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
8.114.2.4  getScaledAppearance()

virtual IDOMNodePtr JawsMako::IAnnotationAppearance::getScaledAppearance (const FRect & annotRect) const [pure virtual]

Get DOM for the annotation scaled appropriately to the given rect. Useful for preparing an annotation appearance for display or rendering. A clone of the contents will be returned.

Parameters

| annotRect | An FRect that defines the scale of the returned DOM node |

Returns

IDOMNodePtr The DOM for the annotation

8.114.2.5  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

8.114.2.6  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
A generic reference to an annotation. The target annotation might not be loaded. Chiefly used to refer to annotations from a Form.

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

Inheritance diagram for JawsMako::IAnnotationReference:

![Inheritance Diagram](image)

- **Public Member Functions**
  - virtual bool equals (const IAnnotationReferencePtr &other) const =0
    
    Determine if another annotation reference refers to the same annotation.

- **Additional Inherited Members**

  8.115.1 Detailed Description

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

  8.115.2 Member Function Documentation

  8.115.2.1 equals()

  ```c++
  virtual bool JawsMako::IAnnotationReference::equals ( 
    const IAnnotationReferencePtr & other ) const [pure virtual]
  ```

  Determine if another annotation reference refers to the same annotation.
Parameters

| other | The other annotation reference |

Returns

bool True if other refers to the same annotation

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

- interactive.h

8.116 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.

8.116.1 Detailed Description

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

8.116.2 Member Function Documentation

8.116.2.1 createAnnotationReferenceFromTag()

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

Generate an IAnnotationReference from an Annotation Tag attribute found in the XML.
8.117 JawsMako::ICaretAnnotation Class Reference

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

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

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

- `interactive.h`
Inheritance diagram for JawsMako::ICaretAnnotation:

```
IRCOBJECT

JawsMako::IAnnotation

JawsMako::IMarkupAnnotation

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.

Additional Inherited Members

8.117.1 Detailed Description

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

8.117.2 Member Function Documentation

8.117.2.1 `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`
A simple transform that looks for CID CFF Fonts containing multiple SubFonts. Some viewers do not support these fonts, or do so poorly. If found, this transform will split out the sub fonts into individual font streams, and adjust the Glyphs nodes where they are used accordingly.

#include <transforms.h>

Inheritance diagram for JawsMako::ICFFCIDSplitterTransform:

Static Public Member Functions

- static JAWSMAKO_API ICFFCIDSplitterTransformPtr create (const IJawsMakoPtr &jawsMako)
  
  Create the transform.

Additional Inherited Members

8.118.1 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.

8.118.2 Member Function Documentation
8.118.2.1 create()

```cpp
static JAWSMAKO_API ICFFCIDSplitterTransformPtr JawsMako::ICFFCIDSplitterTransform::create ( const IJawsMakoPtr & jawsMako ) [static]
```

Create the transform.
Parameters

| JawsMako | The JawsMako instance. |

Returns

The new instance.

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

- transforms.h

### 8.119 JawsMako::IColorConverterTransform Class Reference

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

```cpp
#include <transforms.h>
```

Inheritance diagram for JawsMako::IColorConverterTransform:

![Inheritance Diagram](image)

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 setConvertOnlyDeviceIndependentColors (bool convert)=0
  
  Sets whether or not only device-independent colors should be converted. Useful for cases where device independent colour 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 colours will be converted to DeviceRGB and so forth. If false, the target space will be used. 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 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() will affect this default.

- 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.

Static Public Member Functions

- static JAWSMAKO_API IColorConverterTransformPtr create (const IJawsMakoPtr &jawsMako)
  
  Create the transform.

Additional Inherited Members

8.119.1 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 colour spaces, instances of this transform may opt to use a modified indexed color space rather than actually convert the underlying color. This is usually much faster. An IComplexColorSimplifierTransform can be applied after this transform if required.

Objects using DeviceN color spaces are converted only if the alternate space would require conversion. Again, an IComplexColorSimplifierTransform can be used if required.

8.119.2 Member Function Documentation

8.119.2.1 create()

static JAWSMAKO_API IColorConverterTransformPtr JawsMako::IColorConverterTransform::create (const IJawsMakoPtr & jawsMako) [static]

Create the transform.

Parameters

JawsMako The JawsMako instance.

Returns

The new instance.

8.119.2.2 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 colour space, then this will affect the calculation of the opacity, and potentially undesirable results in some contexts.

8.119.2.3 setTargetProfile()

virtual void JawsMako::IColorConverterTransform::setTargetProfile (const IDOMICCProfilePtr & profile) [pure virtual]

Set the desired target space using an ICC profile.
Parameters

| profile | The ICC profile to use. |

8.119.2.4 setTargetSpace()

virtual void JawsMako::IColorConverterTransform::setTargetSpace ( const IDOMColorSpacePtr & targetSpace ) [pure virtual]

Set the desired target space for conversion. The default is DeviceRGB. Indexed or DeviceN color spaces cannot be used as a target space.

Parameters

| targetSpace | The target color space to be used. |

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

- transforms.h

8.120 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 (int numColors, bool hasAlpha, IDOMColorSpacePtr &sourceSpace, IDOMColorSpacePtr &destSpace, eRenderingIntent intent, eBlackPointCompensation bpc, float *in, float *out)=0`
  
  Convert colors from one color space to another, using floating point samples.

- `virtual void convertColors (int numColors, bool hasAlpha, IDOMColorSpacePtr &sourceSpace, IDOMColorSpacePtr &destSpace, eRenderingIntent intent, eBlackPointCompensation bpc, uint8 *in, uint8 *out, int inBPS, int 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 void getProfileVersionForICCBasedSpace (const IDOMColorSpaceICCBasedPtr &space, uint8 &majorVersion, uint8 &minorVersion)=0`
  
  Get the version number of the given ICC profile.

- `virtual void getProfileNameForICCBasedSpace (const IDOMColorSpaceICCBasedPtr &space, EDLSysString &name)=0`
  
  Get the name of the profile for an ICCBased color space.

- `virtual IDOMICCProfilePtr getCMYKSWOPProfile ()=0`
  
  Retrieve the built-in CMYK SWOP profile.

- `virtual IDOMICCProfilePtr getGrayProfile ()=0`
  
  Retrieve the built-in sGray profile.

- `virtual IDOMICCProfilePtr getRGBProfile ()=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>(), const CEDLVector<double> &matrix=CEDLVector<double>(), 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> &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 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 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.

- virtual void getPostScriptCSAForICCBasedSpace (CEDLSimpleBuffer &csaMemory, const IDOMColorSpaceICCBasedPtr &space, eRenderingIntent intent)=0
  
  Get an equivalent PostScript CSA for an ICC based colour space. This will be a CIEBased color space in the form of PostScript code.

- virtual bool getEmbeddedPSCSAForICCBasedSpace (CEDLSimpleBuffer &csaMemory, const IDOMColorSpaceICCBasedPtr &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.

### Static Public Member Functions

- static EDL_API IColorManagerPtr get (IEDLClassFactory *pFactory)
  
  Get the color manager singleton. Throws an IEDLError on failure.

### Additional Inherited Members

#### 8.120.1 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.

#### 8.120.2 Member Function Documentation
8.120.2.1 convertColors() [1/2]

virtual void IColorManager::convertColors ( 
    int numColors,
    bool hasAlpha,
    IDOMColorSpacePtr & sourceSpace,
    IDOMColorSpacePtr & destSpace,
    eRenderingIntent intent,
    eBlackPointCompensation bpc,
    float * in,
    float * out ) [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.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>numColors</td>
<td>The number of colors to convert.</td>
</tr>
<tr>
<td>hasAlpha</td>
<td>Set to true if the color data contains alpha.</td>
</tr>
<tr>
<td>sourceSpace</td>
<td>The color space describing the input samples.</td>
</tr>
<tr>
<td>destSpace</td>
<td>The desired color space for the output samples.</td>
</tr>
<tr>
<td>intent</td>
<td>The desired rendering intent for the conversion.</td>
</tr>
<tr>
<td>bpc</td>
<td>The desired handling of black point compensation. If in doubt, use eBPCDefault.</td>
</tr>
<tr>
<td>in</td>
<td>Pointer to the input sample buffer.</td>
</tr>
<tr>
<td>out</td>
<td>Pointer to the output sample buffer.</td>
</tr>
</tbody>
</table>

8.120.2.2 convertColors() [2/2]

virtual void IColorManager::convertColors ( 
    int numColors,
    bool hasAlpha,
    IDOMColorSpacePtr & sourceSpace,
    IDOMColorSpacePtr & destSpace,
    eRenderingIntent intent,
    eBlackPointCompensation bpc,
    uint8 * in,
    uint8 * out,
    int inBPS,
    int 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.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>numColors</td>
<td>The number of colors to convert.</td>
</tr>
<tr>
<td>hasAlpha</td>
<td>Set to true if the color data contains alpha.</td>
</tr>
</tbody>
</table>
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sourceSpace</td>
<td>The color space describing the input samples.</td>
</tr>
<tr>
<td>destSpace</td>
<td>The desired color space for the output samples.</td>
</tr>
<tr>
<td>intent</td>
<td>The desired rendering intent for the conversion.</td>
</tr>
<tr>
<td>bpc</td>
<td>The desired handling of black point compensation. If in doubt, use eBPCDefault.</td>
</tr>
<tr>
<td>in</td>
<td>Pointer to the input sample buffer.</td>
</tr>
<tr>
<td>out</td>
<td>Pointer to the output sample buffer.</td>
</tr>
<tr>
<td>inBPS</td>
<td>The bits per sample of the input samples. Currently 1, 2, 4, 8, 12 and 16 bits are supported.</td>
</tr>
<tr>
<td>outBPS</td>
<td>The desired bits per sample of the output samples. Currently 1, 2, 4, 8, 12 and 16 bits are supported.</td>
</tr>
</tbody>
</table>

8.120.2.3 createCalibratedGrayProfile()

virtual IDOMICCProfilePtr IColorManager::createCalibratedGrayProfile (const CEDLVector< double >& whitePoint, double gamma = 1.0, const CEDLVector< double >& blackPoint = CEDLVector< double >()) [pure virtual]

Create a profile for a calibrated gray color space with the given parameters. Such a profile will be analogous to the PDF CalGray color space.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>whitePoint</td>
<td>A three entry vector describing the white point of the color space, in the CIE XYZ color space.</td>
</tr>
<tr>
<td>gamma</td>
<td>Optional. The gamma for the color space, 1.0 if not provided</td>
</tr>
<tr>
<td>blackPoint</td>
<td>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.</td>
</tr>
</tbody>
</table>

Returns

IDOMICCProfilePtr A smart pointer to the resulting profile.

8.120.2.4 createCalibratedRGBProfile()

virtual IDOMICCProfilePtr IColorManager::createCalibratedRGBProfile (const CEDLVector< double >& whitePoint, const CEDLVector< double >& gamma = CEDLVector< double >(), const CEDLVector< double >& matrix = CEDLVector< double >(), const CEDLVector< double >& blackPoint = CEDLVector< double >()) [pure virtual]

Create a profile for a calibrated RGB color space with the given parameters. Such a profile will be analogous to the PDF CalRGB color space.
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>whitePoint</td>
<td>A three entry vector describing the white point of the color space, in the CIE XYZ color space.</td>
</tr>
<tr>
<td>gamma</td>
<td>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.</td>
</tr>
<tr>
<td>matrix</td>
<td>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 identity matrix (1, 0, 0, 1, 0, 0, 0, 1, 0) will be used.</td>
</tr>
<tr>
<td>blackPoint</td>
<td>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.</td>
</tr>
</tbody>
</table>

Returns

**IDOMICCProfilePtr** A smart pointer to the resulting profile.

### 8.120.2.5 createGrayProfile()

```cpp
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>whitePoint</td>
<td>A three entry vector describing the white point of the color space, in the CIE XYZ color space.</td>
</tr>
<tr>
<td>gamma</td>
<td>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).</td>
</tr>
<tr>
<td>blackPoint</td>
<td>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.</td>
</tr>
</tbody>
</table>

Returns

**IDOMICCProfilePtr** A smart pointer to the resulting profile.

### 8.120.2.6 get()

```cpp
static EDL_API IColorManagerPtr IColorManager::get {
    IEDLClassFactory * pFactory ) [static]
```

Get the color manager singleton. Throws an IEDLError on failure.
Parameters

| pFactory | The EDL Class factory to use |

Returns

IColorManagerPtr Pointer to the color manager

8.120.2.7 getCMYKSWOPProfile()

virtual IDOMICCProfilePtr IColorManager::getCMYKSWOPProfile ( ) [pure virtual]

Retrieve the built-in CMYK SWOP profile.

Returns

IDOMICCProfilePtr A smart pointer to the resulting profile.

8.120.2.8 getDefaultIntentForICCBasedSpace()

virtual eRenderingIntent IColorManager::getDefaultIntentForICCBasedSpace ( const IDOMColorSpaceICCBasedPtr & space ) [pure virtual]

Find the default rendering intent for an ICCBased color space.

Parameters

| space | The ICC space to interrogate |

Returns

eRenderingIntent The default intent

8.120.2.9 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.
8.120.2.10  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

8.120.2.11  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.

8.120.2.12  getEmbeddedPSCSAForICCBasedSpace()

virtual bool IColorManager::getEmbeddedPSCSAForICCBasedSpace (CEDLSimpleBuffer & csaMemory, const IDOMColorSpaceICCBasedPtr & space ) [pure virtual]

Get any PostScript CSA present in an ICC Based color space. These are found under the 'ps2s' tag in an ICC profile, and are optional.

Parameters

| csaMemory | A reference to a simple buffer to store the given csa. |
| space     | The ICC Based color space to process.                  |

Returns

bool True if an embedded CSA was found

8.120.2.13  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

---

### 8.120.2.14 `getNumComponentsForICCBasedSpace()`

```cpp
virtual uint8 IColorManager::getNumComponentsForICCBasedSpace (
    const IDOMColorSpaceICCBasedPtr & space ) [pure virtual]
```

Find the number of components for an ICCBased color space.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>space</code></td>
<td>The ICC space to interrogate</td>
</tr>
</tbody>
</table>

**Returns**

`uint8` The number of components in the ICC space

---

### 8.120.2.15 `getNumComponentsForICCProfile()`

```cpp
virtual uint8 IColorManager::getNumComponentsForICCProfile (    
    const IDOMICCProfilePtr & profile ) [pure virtual]
```

Find the number of components for an ICC profile.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>profile</code></td>
<td>The ICC profile to interrogate</td>
</tr>
</tbody>
</table>

**Returns**

`uint8` The number of components in the ICC space

---

### 8.120.2.16 `getPostScriptCSAForICCBasedSpace()`

```cpp
virtual void IColorManager::getPostScriptCSAForICCBasedSpace (    
    CEDLSimpleBuffer & csaMemory, 
```

Generated by Doxygen
Get an equivalent PostScript CSA for an ICC based colour space. This will be a CIEBased color space in the form of PostScript code.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>csaMemory</td>
<td>A reference to a simple buffer to store the generated csa.</td>
</tr>
<tr>
<td>space</td>
<td>The ICC Based color space to process.</td>
</tr>
<tr>
<td>intent</td>
<td>The intent for which the CSA should be generated.</td>
</tr>
</tbody>
</table>

8.120.2.17  getProfileForSpace()

virtual IDOMICCProfilePtr IColorManager::getProfileForSpace (  
    const IDOMColorSpacePtr & space ) [pure virtual]

Get a profile for the given space, if possible.

Convenience.

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 colour space types, NULL will be returned.

8.120.2.18  getProfileNameForICCBasedSpace()

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

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

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>space</td>
<td>The ICC color space to interrogate.</td>
</tr>
<tr>
<td>name</td>
<td>Reference to receive the profile name</td>
</tr>
</tbody>
</table>

8.120.2.19  getProfileVersionForICCBasedSpace()

virtual void IColorManager::getProfileVersionForICCBasedSpace (  
    const IDOMColorSpaceICCBasedPtr & space,  
    eRenderingIntent intent ) [pure virtual]
Get the version number of the given ICC profile.

**Parameters**

<table>
<thead>
<tr>
<th>space</th>
<th>The ICC color space to interrogate.</th>
</tr>
</thead>
<tbody>
<tr>
<td>majorVersion</td>
<td>Reference to receive the major version</td>
</tr>
<tr>
<td>minorVersion</td>
<td>Reference to receive the minor version</td>
</tr>
</tbody>
</table>

### 8.120.2.20  getscRGBProfile()

```cpp
virtual IDOMICCProfilePtr IColorManager::getscRGBProfile ( ) [pure virtual]
```

Retrieve the built in scRGB profile.

**Returns**

`IDOMICCProfilePtr` A smart pointer to the resulting profile.

### 8.120.2.21  getsGrayProfile()

```cpp
virtual IDOMICCProfilePtr IColorManager::getsGrayProfile ( ) [pure virtual]
```

Retrieve the built-in sGray profile.

**Returns**

`IDOMICCProfilePtr` A smart pointer to the resulting profile.

### 8.120.2.22  getsRGBProfile()

```cpp
virtual IDOMICCProfilePtr IColorManager::getsRGBProfile ( ) [pure virtual]
```

Retrieve the built-in sRGB profile.

**Returns**

`IDOMICCProfilePtr` A smart pointer to the resulting profile.

### 8.120.2.23  interceptSpace()

```cpp
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>space</td>
<td>The color space to find the intercept for.</td>
</tr>
</tbody>
</table>

Returns

**IDOMColorSpacePtr** Returns the intercepted space, if applicable. Otherwise the original space is returned.

### 8.120.2.24 setDeviceCMYKIntercept()

```cpp
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>space</td>
<td>The desired DeviceCMYK intercept color space.</td>
</tr>
</tbody>
</table>

### 8.120.2.25 setDeviceGrayIntercept()

```cpp
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>space</td>
<td>The desired DeviceGray intercept color space.</td>
</tr>
</tbody>
</table>

### 8.120.2.26 setDeviceRGBIntercept()

```cpp
virtual void IColorManager::setDeviceRGBIntercept (
    const IDOMColorSpacePtr & space) [pure virtual]
```

Set the intercept color space for DeviceRGB. This is the color space that will be used for conversion if the color manager is asked to convert from or to DeviceRGB.
Parameters

| space       | The desired DeviceRGB intercept colorspace. |

8.120.2.27  setMapDeviceGrayToCMYKBlack()

virtual void IColorManager::setMapDeviceGrayToCMYKBlack (
bool mapGrayDirectly ) [pure virtual]

Configure how DeviceGray colors are converted to DeviceCMYK.

Normally, all color conversions in JawsMako are achieved using color management. In particular, this includes the DeviceGray, DeviceCMYK and DeviceRGB color spaces, which are converted using the intercept color spaces configured in the functions below.

However this means that conversions between DeviceGray and DeviceCMYK generally pass through ICC conversion. Gray values may not map exclusively to DeviceCMYK K values, resulting in non-zero values in the Cyan, Magenta and Yellow components.

This may be unsuitable in some environments, in particular when dealing with overprint simulation.

This behavior can be changed with this member. If the argument is

- true: All color conversions from DeviceGray to DeviceCMYK will simply map and invert the gray value to the K channel. This will also apply to any rendering operations performed by JawsMako.
- false (default): The intercept color spaces will be used to convert from DeviceGray to DeviceCMYK.

Parameters

| mapGrayDirectly | Determine how DeviceGray colors are converted to DeviceCMYK |

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

- icolormananger.h

8.121  JawsMako::IColorComplexColorSimplifierTransform 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.

#include <transforms.h>
Inheritance diagram for JawsMako::IComplexColorSimplifierTransform:

Static Public Member Functions

- static JAWSMAKO_API IComplexColorSimplifierTransformPtr create (const IJawsMakoPtr &jawsMako)
  Create the transform.

Additional Inherited Members

8.121.1 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.

8.121.2 Member Function Documentation

8.121.2.1 create()

static JAWSMAKO_API IComplexColorSimplifierTransformPtr JawsMako::IComplexColorSimplifierTransform::create (const IJawsMakoPtr &jawsMako) [static]
Create the transform.
Parameters

- **JawsMako**

The JawsMako instance.

Returns

The new instance.

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

- `transforms.h`

### 8.122 IDOMPDFImage::IDecodeParams Class Reference

Abstract interface for per-image decoding filter parameters.

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

Inheritance diagram for IDOMPDFImage::IDecodeParams:

Additional Inherited Members

#### 8.122.1 Detailed Description

Abstract interface for per-image decoding filter parameters.

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

- `idomimageresource.h`
8.123 JawsMako::IDocument Class Reference

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

#include <jawsmako.h>

Inheritance diagram for JawsMako::IDocument:

IRObject

| 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 void insertPage (const IPagePtr &page, uint32 index=0, const IDocumentPtr &sourceDocument=NULL)=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=NULL)=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 CFFileSpecAsEmbeddedDataVect 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.

**Static Public Member Functions**

- static JAWSMAKO_API IDocumentPtr create (const IJawsMakoPtr &jawsMako)
  Create an empty document.

**Additional Inherited Members**

8.123.1 Detailed Description

A document from an IDocumentAssembly, allowing for high level document and page management, and providing on-demand lazy loading of page markup.
8.123.2 Member Function Documentation

8.123.2.1 addNamedDestination()

virtual void JawsMako::IDocument::addNamedDestination (  
    const INamedDestinationPtr & namedDestination ) [pure virtual]

Add a named destination to the document. Note that this will override any named destination with the same name that may be present.

Parameters

| namedDestination | The named destination to be added |

8.123.2.2 appendPage()

virtual void JawsMako::IDocument::appendPage (  
    const IPagePtr & page,  
    const IDocumentPtr & sourceDocument = IDocumentPtr(NULL) ) [pure virtual]

Append a page to the document.

Parameters

| page | smart pointer to the page to be inserted. |
| sourceDocument | (optional) the source document that the page came from. If provided, the target document's outline will be update with any outline entries present on the inserted page. The interactive forms will be updated with any widget annotations present on the page. Note that if the names of any widgets being added clash with existing widgets or fields they will be forcibly renamed. Provide a sourceDocument if possible for improved form merging. |

8.123.2.3 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 method**

```cpp
static JAWSMAKO_API IDocumentPtr JawsMako::IDocument::create (const IJawsMakoPtr &jawsMako) [static]
```

Create an empty document.

**Returns**

*IDocumentPtr* the new page.

### getForm()

**Virtual method**

```cpp
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 method**

```cpp
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 method**

```cpp
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.
8.123.2.8  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.

8.123.2.9  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.

8.123.2.10  getThreads()

virtual IThreadsPtr JawsMako::IDocument::getThreads ( ) const [pure virtual]

Get the document threads, if present.

Returns

IThreadsPtr The document threads, if present.

8.123.2.11  insertPage()

virtual void JawsMako::IDocument::insertPage ( 
    const IPagePtr & page, 
    uint32 index = 0, 
    const IDocumentPtr & sourceDocument = IDocumentPtr(NULL) ) [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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>page</td>
<td>smart pointer to the page to be inserted.</td>
</tr>
<tr>
<td>index</td>
<td>The position in the document to insert the page, 0 being the first position.</td>
</tr>
<tr>
<td>sourceDocument</td>
<td>(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.</td>
</tr>
</tbody>
</table>

8.123.2.12  removePage()

```cpp
virtual void JawsMako::IDocument::removePage ( uint32 index ) [pure virtual]
```

Remove the page at the given index.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>The index of the page to be removed (0 being the first page)</td>
</tr>
</tbody>
</table>

8.123.2.13  setNamedDestinations()

```cpp
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>namedDestinations</td>
<td>The vector of named destinations</td>
</tr>
</tbody>
</table>

8.123.2.14  setOutline()

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

Set the document outline, if present.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>outline</td>
<td>The desired document outline, or NULL to set no outline.</td>
</tr>
</tbody>
</table>
The documentation for this class was generated from the following file:

- jawsmako.h

### 8.124 JawsMako::IDocumentAssembly Class Reference

A self contained collection of IDocuments.

```cpp
#include <jawsmako.h>
```

Inheritance diagram for JawsMako::IDocumentAssembly:

```
IRCOBJECT

JawsMako::IDocumentAssembly
```

#### Public Member Functions

- **virtual uint32 getNumDocuments ()=0**
  
  Return the number of documents in the assembly, if known.

- **virtual IDocumentPtr getDocument (uint32 index=0)=0**

  Get the IDocument from the assembly at the given index.

- **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 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 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, 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.

Static Public Member Functions

• static JAWSMAKO_API IDocumentAssemblyPtr create (const IJawsMakoPtr &jawsMako)
  
  Create a new, empty document assembly.

Additional Inherited Members

8.124.1 Detailed Description

A self contained collection of IDocuments.

8.124.2 Member Function Documentation

8.124.2.1 appendDocument()

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

Append a document to the assembly.
Parameters

| page | smart pointer to the page to be appended. |

8.124.2.2 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.

8.124.2.3 create()

static JAWSMAKO_API IDocumentAssemblyPtr JawsMako::IDocumentAssembly::create ( const IJawsMakoPtr & jawsMako ) [static]

Create a new, empty document assembly.

Returns

IDocumentAssemblyPtr the new assembly.

8.124.2.4 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.
8.124.2.5 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.

8.124.2.6 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

8.124.2.7 getNumDocuments()

virtual uint32 JawsMako::IDocumentAssembly::getNumDocuments ( ) [pure virtual]

Return the number of documents in the assembly, if known.

Returns

uint32 The number of pages in the assembly.

8.124.2.8 getSecurityInfo()

virtual IDOMSecurityInfoPtr JawsMako::IDocumentAssembly::getSecurityInfo ( ) const [pure virtual]

Get the security information that applied to the source file/stream. Currently only relevant for PDF.

Returns

IDOMPDFSecurityInfoPtr. Returns NULL if no security information was provided by the input.
8.124.2.9  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.

8.124.2.10  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.

8.124.2.11  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

<table>
<thead>
<tr>
<th>document</th>
<th>smart pointer to the document to be inserted.</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>The position in the assembly to insert the document, 0 being the first position.</td>
</tr>
</tbody>
</table>

8.124.2.12  removeDocument()

virtual void JawsMako::IDocumentAssembly::removeDocument ( 
    uint32 index ) [pure virtual]

Remove the document at the given index.
Parameters

| index | The index of the page to be removed (0 being the first document). |

8.124.2.13  setThumbnail()

virtual void JawsMako::IDocumentAssembly::setThumbnail(
    const IDOMImagePtr & thumbnail) [pure virtual]

Sets a new thumbnail image for the document assembly. The image must be in either JPEG or PNG format. Setting thumbnail to NULL deletes the document assembly thumbnail.

Parameters

| thumbnail | Smart pointer to new thumbnail image. |

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

- jawsmako.h

8.125  IDOMActionArray Class Reference

IDOMActionArray interface.

#include <idomtarget.h>

Inheritance diagram for IDOMActionArray:

```
IRCOObject

IEDLObject

IDOMTarget

IDOMActionArray
```
Public Member Functions

- virtual IDOMTargetCollectionEnumPtr getActionsEnum ()=0
  Retrieves enumerator of the action collection. Action collection is analog of entry "Next" of PDF action dictionary.
- virtual uint32 getActionsCount ()=0
  Retrieves count of the actions.
- virtual void clearActions ()=0
  Clears collection of actions.
- virtual bool addAction (const IDOMTargetPtr &ptrAction)=0
  Append a action to the action collection.
- virtual eTargetType getTargetType () const
  Implementation of geTargetType for IDOMActionArray.

Additional Inherited Members

8.125.1 Detailed Description

IDOMActionArray interface.

8.125.2 Member Function Documentation

8.125.2.1 addAction()

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

Append a action to the action collection.

Parameters

| ptrAction | The smart pointer to the target interface |

Returns

bool Returns true on success

8.125.2.2 getActionsCount()

virtual uint32 IDOMActionArray::getActionsCount () [pure virtual]

Retrieves count of the actions.

Returns

uint32 Size of action collection
8.125.2.3 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

8.125.2.4 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

8.126 IDOMActionLaunch Class Reference

IDOMActionLaunch interface.
#include <idomtarget.h>

Inheritance diagram for IDOMActionLaunch:
Public Member Functions

- virtual EDLString getFile () const =0
  Retrieves the file specification.
- virtual void setFile (const EDLString &file)=0
  Sets the file specification.
- virtual EDLString getWinParameters () const =0
  Retrieves the Windows-specific launch parameters.
- virtual void setWinParameters (const EDLString &params)=0
  Sets the Windows-specific launch parameters.
- virtual EDLString getMacParameters () const =0
  Retrieves the Mac OS-specific launch parameters.
- virtual void setMacParameters (const EDLString &params)=0
  Sets the Mac OS-specific launch parameters.
- virtual EDLString getUnixParameters () const =0
  Retrieves the UNIX-specific launch parameters.
- virtual void setUnixParameters (const EDLString &params)=0
  Sets the UNIX-specific launch parameters.
- virtual bool getNewWindow () const =0
  Getter for NewWindow flag.
- virtual void setNewWindow (bool bNewWindow)=0
  Setter for NewWindow flag.
- virtual eTargetType getTargetType () const
  Implementation of geTargetType for IDOMActionLaunch.

Additional Inherited Members

8.126.1 Detailed Description

IDOMActionLaunch interface.

8.126.2 Member Function Documentation

8.126.2.1 getFile()

virtual EDLString IDOMActionLaunch::getFile ( ) const [pure virtual]

Retrieves the file specification.

Returns

EDLSysString The file specification
8.126.2.2 getMacParameters()

Virtual EDLString IDOMActionLaunch::getMacParameters ( ) const [pure virtual]

Retrieves the Mac OS-specific launch parameters.

Returns

EDLSysString The launch parameters

8.126.2.3 getNewWindow()

Virtual bool IDOMActionLaunch::getNewWindow ( ) const [pure virtual]

Getter for NewWindow flag.

Returns

bool Returns the value of the NewWindow flag

8.126.2.4 getTargetType()

Virtual eTargetType IDOMActionLaunch::getTargetType ( ) const [inline], [virtual]

Implementation of geTargetType for IDOMActionLaunch.

Returns

eTargetType Returns eActionLaunch;

Implements IDOMTarget.

8.126.2.5 getUnixParameters()

Virtual EDLString IDOMActionLaunch::getUnixParameters ( ) const [pure virtual]

Retrieves the UNIX-specific launch parameters.

Returns

EDLSysString The launch parameters
8.126.2.6  getWinParameters()

virtual EDLString IDOMActionLaunch::getWinParameters ( ) const  [pure virtual]

Retrieves the Windows-specific launch parameters.

Returns

EDLSysString The launch parameters

8.126.2.7  setFile()

virtual void IDOMActionLaunch::setFile ( const EDLString & file )  [pure virtual]

Sets the file specification.

Parameters

| file | File specification |

8.126.2.8  setMacParameters()

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

Sets the Mac OS-specific launch parameters.

Parameters

| params | The launch parameters |

8.126.2.9  setNewWindow()

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

Setter for NewWindow flag.

Parameters

| bNewWindow | New value of the NewWindow flag |
8.126.2.10  setUnixParameters()

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

Sets the UNIX-specific launch parameters.

Parameters

| params | The launch parameters |

8.126.2.11  setWinParameters()

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

Sets the Windows-specific launch parameters.

Parameters

| params | The launch parameters |

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

- idomtarget.h

8.127  IDOMArcSegment Class Reference

Interface to Arc Segment element.

#include <idompathgeometry.h>
Inheritance diagram for IDOMArcSegment:

```
IRCOBJECT
  ↓
IEDLObject
  ↓
IDOMPathSegment
  ↓
IDOMArcSegment
```

### Classes

- **class Data**
  
  *Initialization data.*

### Public Types

- **enum SweepDirection**
  
  *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 bool setPoint (const FPoint &pt)=0**
  
  *Sets the end point of the arc.*

- **virtual double getRadiusX () const =0**
  
  *Retrieves the x-radius of the arc.*

- **virtual bool setRadiusX (double radiusX)=0**
  
  *Sets the x-radius of the base ellipses.*

- **virtual double getRadiusY () const =0**
  
  *Retrieves the y-radius of the base ellipses.*

- **virtual bool 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 bool setRotationAngle (double rA)=0
  Sets rotation angle of the arc.

- virtual bool getIsLargeArc () const =0
  Retrieves isLargeArc.

- virtual bool setIsLargeArc (bool isLA)=0
  Sets isLargeArc. See getIsLargeArc() for a description of how isLargeArc determines the arc to be drawn.

- virtual enum SweepDirection getSweepDirection () const =0
  Retrieves the sweep direction.

- virtual bool setSweepDirection (enum SweepDirection sd)=0
  Sets sweep direction.

- virtual bool convertToSimpleSegment (IDOMPathSegmentPtr &newSegment, FPoint &startPoint, IEDLClassFactory *factory)=0
  Create a segment that represents this arc using a simpler segment type.

Static Public Member Functions

- static const CClassID & classID ()
  Retrieves the class id of IDOMArcSegment.

Additional Inherited Members

8.127.1 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.

8.127.2 Member Enumeration Documentation

8.127.2.1 SweepDirection

type enum IDOMArcSegment::SweepDirection

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()).
8.127.3 Member Function Documentation

8.127.3.1 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.

8.127.3.2 convertToSimpleSegment()

virtual bool IDOMArcSegment::convertToSimpleSegment (  
    IDOMPathSegmentPtr & newSegment,  
    FPoint & startPoint,  
    IEDLClassFactory * factory ) [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

| newSegment | Smart pointer to receive the resulting segment |
| startPoint | The start point for the arc |
| factory    | Pointer to the class factory to be used |

Returns

bool. The function returns true on success, false on failure.

8.127.3.3 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().

Generated by Doxygen
Returns

bool. The function returns true if the arc is large

8.127.3.4 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.

8.127.3.5 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.

8.127.3.6 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.
8.127.3.7 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.

8.127.3.8 getSweepDirection()

virtual enum SweepDirection 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

SweepDirection. The return value specifies whether the stroke direction is clockwise or counter clockwise from the start point.

8.127.3.9 setIsLargeArc()

virtual bool IDOMArcSegment::setIsLargeArc ( bool isLA ) [pure virtual]

Sets isLargeArc. See getIsLargeArc() for a description of how isLargeArc determines the arc to be drawn.

Parameters

| isLA   | New value for isLargeArc. |

Returns

bool. The function returns true on success, false on failure.
8.127.3.10 setPoint()

virtual bool IDOMArcSegment::setPoint (  
    const FPoint & pt ) [pure virtual]

Sets the end point of the arc.

Parameters

| pt | The new end point of the arc. |

Returns

bool. The function returns true on success, false on failure.

8.127.3.11 setRadiusX()

virtual bool IDOMArcSegment::setRadiusX (  
    double radiusX ) [pure virtual]

Sets the x-radius of the base ellipses.

Parameters

| radiusX | The new x-radius of the base ellipses. |

Returns

bool. The function returns true on success, false on failure.

8.127.3.12 setRadiusY()

virtual bool IDOMArcSegment::setRadiusY (  
    double radiusY ) [pure virtual]

Sets the y-radius of the base ellipses.

Parameters

| radiusY | The new y-radius of the base ellipses. |

Returns

bool. The function returns true on success, false on failure.
8.127.3.13 setRotationAngle()

virtual bool IDOMArcSegment::setRotationAngle ( double rA ) [pure virtual]

Sets rotation angle of the arc.

Parameters

| rA | New rotation angle. Positive values indicate clockwise rotation and negative values indicate counter clockwise rotation. The rotated ellipses must still be capable of intersecting with the specified start and end points, as before. |

Returns

bool. The function returns true on success, false on failure.

8.127.3.14 setSweepDirection()

virtual bool IDOMArcSegment::setSweepDirection ( enum SweepDirection sd ) [pure virtual]

Sets sweep direction.

Parameters

| sd | SweepDirection The direction in which the arc is drawn from the start point. |

Returns

bool. The function returns true on success, false on failure.

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

- idompathgeometry.h

8.128 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

8.128.1 Detailed Description

IDOMAudioFile interface.

8.128.2 Member Function Documentation
8.129 IDOMBrush Class Reference

Interface to the brush element.

#include <idombrush.h>

Inheritance diagram for IDOMBrush:

Public Types

- enum eBrushType
  
  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 bool setOpacity (float opc)=0
  
  Sets the opacity value of a brush element.
Additional Inherited Members

8.129.1 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.

8.129.2 Member Function Documentation

8.129.2.1 getBrushType()

virtual eBrushType IDOMBrush::getBrushType ( ) const [pure virtual]

Retrieves the type of the brush.

Returns

eBrushType. the brush type

8.129.2.2 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

8.129.2.3 setOpacity()

virtual bool IDOMBrush::setOpacity ( float opc ) [pure virtual]

Sets the opacity value of a brush element.
Parameters

 opc  The opacity value

Returns

bool. The function returns true on success, false on failure.

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

- idombrush.h

8.130  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 bool **getLanguage** (EDLString &lang) const =0
  
  Retrieves the default language of the `<Canvas>` element and any of its children.

- virtual bool **setLanguage** (const EDLString &lang)=0
  
  Sets the default language of the `<Canvas>` element and any of its children.

- virtual bool **getAutomationPropertiesName** (EDLString &propname) const =0
  
  Retrieves the automation properties name of the `<Canvas>` element.

- virtual bool **setAutomationPropertiesName** (const EDLString &propname)=0
  
  Sets the automation properties name of the `<Canvas>` element.

- virtual bool **getAutomationPropertiesHelpText** (EDLString &helptext) const =0
  
  Retrieves the automation properties help text of the `<Canvas>` element.

- virtual bool **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 bool **getNavigateLink** (IDOMTargetPtr &target) const =0
  
  Retrieves the target of a hyperlink.

- virtual bool **setNavigateLink** (const IDOMTargetPtr &target)=0
  
  Sets the target of a hyperlink.

- virtual bool **getResourceDictionary** (IDOMResourceDictionaryPtr &ptrResourceDictionary) const =0
  
  Retrieves a smart pointer to the resource dictionary.

- virtual bool **setResourceDictionary** (const IDOMResourceDictionaryPtr &ptrResourceDictionary)=0
  
  Sets the resource dictionary.

Static Public Member Functions

- static const CClassID & **classID** ()
  
  Retrieves the class id of IDOMCanvas.

Additional Inherited Members

8.130.1 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.
8.130.2 Member Function Documentation

8.130.2.1 classID()

static const CClassID & IDOMCanvas::classID ( ) [inline], [static]

Retrieves the class id of IDOMCanvas.

Returns

CClassID Class id of the element

8.130.2.2 getAutomationPropertiesHelpText()

virtual bool IDOMCanvas::getAutomationPropertiesHelpText ( 
    EDLString & helptext ) 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."

Parameters

helptext The automation properties help text.

Returns

bool. Returns true on success, false on failure.

8.130.2.3 getAutomationPropertiesName()

virtual bool IDOMCanvas::getAutomationPropertiesName ( 
    EDLString & propropname ) 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.
8.130.2.4 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.

8.130.2.5 getLanguage() 

virtual bool IDOMCanvas::getLanguage ( 
    EDLString & lang ) 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.

Parameters

| lang | The language setting of the <Canvas> element |

Returns

 bool. Returns true on success, false on failure.
8.130.2.6 getNavigateLink()

virtual bool IDOMCanvas::getNavigateLink ( 
   IDOMTargetPtr & target ) const [pure virtual]

Retrieves the target of a hyperlink.

Parameters

| target | Smart pointer to receive the target of the hyperlink |

Returns

bool. Returns true on success, false on failure.

8.130.2.7 getResourceDictionary()

virtual bool IDOMCanvas::getResourceDictionary ( 
   IDOMResourceDictionaryPtr & ptrResourceDictionary ) 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.

Parameters

| ptrResourceDictionary | A smart pointer to receive the reference to the resource dictionary. |

Returns

bool. Returns true on success, false on failure.

8.130.2.8 setAutomationPropertiesHelpText()

virtual bool IDOMCanvas::setAutomationPropertiesHelpText ( 
   const EDLString & helptext ) [pure virtual]

Sets automation properties help text of the <Canvas> element.

The automation properties help text of a canvas is a more detailed description of the canvas content for accessibility purposes. This is particularly useful if the canvas is filled with a set of vector graphics and text elements intended to comprise a single vector graphic. For example, “This is a picture of the Earth.”

Parameters

| helptext | The new automation properties help text. |
Returns

bool. Returns true on success, false on failure.

8.130.2.9  setAutomationPropertiesName()

virtual bool IDOMCanvas::setAutomationPropertiesName (const EDLString & propname) [pure virtual]

Sets the automation properties name of the <Canvas> element.

The automation properties name is a brief description of the <Canvas> content for accessibility purposes, particularly if the canvas is filled with a set of vector graphics and text elements intended to comprise a single vector graphic.

Parameters

| propname | The new automation properties name. |

Returns

bool. Returns true on success, false on failure.

8.130.2.10  setEdgeMode()

virtual void IDOMCanvas::setEdgeMode (eEdgeMode em) [pure virtual]

Sets render the options edge mode of the <Canvas> element.

The EdgeMode property controls how the edges of paths within the canvas are rendered. The only valid value is Aliased. Omitting this attribute causes the edges to be rendered in the consumers default manner. The EdgeMode property can be set in a Canvas to instruct anti-aliasing consumers to render the contents of the Canvas and all child and descendant nodes without performing antialiasing.

Parameters

| em | The new edge mode for the canvas. |

Returns

bool. Returns true on success, false on failure.
8.130.2.11 setLanguage()

```cpp
virtual bool IDOMCanvas::setLanguage (  
    const EDLString & lang ) [pure virtual]
```

Sets the default language of the `<Canvas>` element and any of its children.

**Parameters**

<table>
<thead>
<tr>
<th>lang</th>
<th>The new language setting for the canvas.</th>
</tr>
</thead>
</table>

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/](http://www.w3.org/)←International/articles/language-tags/.

The language is specified according to RFC 3066.

**Returns**

bool. Returns true on success, false on failure.

8.130.2.12 setNavigateLink()

```cpp
virtual bool IDOMCanvas::setNavigateLink (  
    const IDOMTargetPtr & target ) [pure virtual]
```

Sets the target of a hyperlink.

**Parameters**

<table>
<thead>
<tr>
<th>target</th>
<th>The new target for the hyperlink.</th>
</tr>
</thead>
</table>

**Returns**

bool. Returns true on success, false on failure.

8.130.2.13 setResourceDictionary()

```cpp
virtual bool IDOMCanvas::setResourceDictionary (  
    const IDOMResourceDictionaryPtr & ptrResourceDictionary ) [pure virtual]
```

Sets the resource dictionary.

The resource dictionary contains the IDs of resources stored at this level of the document.
Parameters

| ptrResourceDictionary | Smart pointer to the resource dictionary |

Returns

bool. Returns true on success, false on failure.

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

- idomcanvas.h

8.131 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:

```
IRCOBject

IEDLObject

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.

Static Public Member Functions

• static const CClassID & classID ()
  Retrieves the class id of IDOMCatalog.

Additional Inherited Members

8.131.1 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.

8.131.2 Member Function Documentation

8.131.2.1 classID()

static const CClassID & IDOMCatalog::classID ( ) [inline], [static]

Retrieves the class id of IDOMCatalog.

Returns

  CClassID Class id of the catalog.

8.131.2.2 createNewDOMid() [1/2]

virtual DOMid IDOMCatalog::createNewDOMid ( const EDLSysString & uri ) [pure virtual]

Creates a new DOM id from a URI string.
Parameters

**uri**  A string uniquely addressing a resource.

Returns

**DOMid**. The newly created DOM id. The newly allocated ID will be unique across the current process.

8.131.2.3 `createNewDOMid()` [2/2]

```cpp
virtual DOMid IDOMCatalog::createNewDOMid (
    uint32 id1,
    uint32 id2 = 0 ) [pure virtual]
```

Creates a new DOM id based on an integer identifier.

Parameters

<table>
<thead>
<tr>
<th>id1</th>
<th>A numeric identifier.</th>
</tr>
</thead>
<tbody>
<tr>
<td>id2</td>
<td>A second numeric identifier (optional. Presently not used, and set to zero.)</td>
</tr>
</tbody>
</table>

Returns

**DOMid**. The newly created DOM id. The newly allocated ID will be unique across the current process.

8.131.2.4 `getCount()`

```cpp
virtual uint32 IDOMCatalog::getCount ( ) const [pure virtual]
```

Retrieves count of registered resource objects.

Returns

**uint32** Returns the count of catalog entries.

8.131.2.5 `getIdByIndex()`

```cpp
virtual DOMid IDOMCatalog::getIdByIndex ( 
    uint32 index ) const [pure virtual]
```

Retrieves the DOM ID of a registered resource by its index in the internal vector.
8.131 IDOMCatalog Class Reference

Parameters

| index | The registered resource index |

Returns

**DOMid** Returns the DOM ID of the resource.

### 8.131.2.6 getIdByNumbers()

```cpp
virtual DOMid IDOMCatalog::getIdByNumbers ( 
  uint32 id1, 
  uint32 id2 = 0, 
  bool createIfNotExist = false ) [pure virtual]
```

Retrieve the DOM ID of a registered resource by using the integer identifier(s) used to create the DOM ID, optionally creating it.

Parameters

| id1   | A numeric identifier. |
| id2   | A second numeric identifier (optional. Presently not used, and set to zero.) |
| createIfNotExist | If set to true, create a new DOM ID if it doesn't already exist |

Returns

**DOMid** The DOM id of the identified resource. If a new id has been created it will be unique across the process.

### 8.131.2.7 getIdByURI()

```cpp
virtual DOMid IDOMCatalog::getIdByURI ( 
  const EDLSysString & uri, 
  bool createIfNotExist = false ) [pure virtual]
```

Retrieve the DOM ID of a registered resource by using its URI, optionally creating it.

Parameters

| uri   | URI of the resource whose DOM ID is required. |
| createIfNotExist | If set to true, create a new DOM ID if it doesn't already exist |
Returns

**DOMid** The DOM ID of the resource.

### 8.131.2.8 `getObject()`

```cpp
virtual bool IDOMCatalog::getObject (  
    DOMid id,  
    IEDLObjectPtr & ptrObject ) [pure virtual]
```

Retrieves an *IEDLObject* by using its DOM id.

**Parameters**

<table>
<thead>
<tr>
<th>id</th>
<th>The DOM ID of the <em>IEDLObject</em> to retrieve.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ptrObject</td>
<td>A Smart pointer to receive a reference to the retrieved <em>IEDLObject</em>.</td>
</tr>
</tbody>
</table>

**Returns**

**bool** True on success, false if the call fails.

### 8.131.2.9 `getURI()`

```cpp
virtual bool IDOMCatalog::getURI (  
    DOMid id,  
    EDLSysString & uri ) [pure virtual]
```

Retrieves the URI of a registered object by using its DOM ID.

**Parameters**

<table>
<thead>
<tr>
<th>id</th>
<th>DOM ID of the resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>uri</td>
<td>Pointer to receive the URI for the resource</td>
</tr>
</tbody>
</table>

**Returns**

**bool** True on success, false if the call fails.

### 8.131.2.10 `registerNumbers()`

```cpp
virtual bool IDOMCatalog::registerNumbers (  
    DOMid id,  
```

Generated by Doxygen
uint32 id1,
uint32 id2 ) [pure virtual]

Registers the number pair in the catalog, under the ID specified by id.
Parameters

<table>
<thead>
<tr>
<th>id</th>
<th>The ID to register the object under.</th>
</tr>
</thead>
<tbody>
<tr>
<td>id1</td>
<td>A numeric identifier.</td>
</tr>
<tr>
<td>id2</td>
<td>A second numeric identifier.</td>
</tr>
</tbody>
</table>

Returns

bool True on success, false if the call fails.

8.131.2.11 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

<table>
<thead>
<tr>
<th>id</th>
<th>The ID to register the object under.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ptrObject</td>
<td>A smart pointer to the IEDLObject to register.</td>
</tr>
</tbody>
</table>

Returns

bool True on success, false if the call fails.

8.131.2.12 unregisterObject()

virtual bool IDOMCatalog::unregisterObject (  
    DOMid id ) [pure virtual]

Unregisters the IEDLObject currently registered under the specified ID in the catalog.

Parameters

| id     | The DOM id of the object to unregister. |

Returns

bool True on success, false if the call fails.

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

- idomcatalog.h
**8.132  IDOMCharPathGroup Class Reference**

`IDOMCharPathGroup` interface.

```cpp
#include <idomcharpathgroup.h>
```

Inheritance diagram for `IDOMCharPathGroup`:

```
IRCOObject
    
IEDLObject
    
IDOMNode
    
IDOMGroup
    
IDOMCharPathGroup
```

**Classes**

- class **Data**
  
  *Initialization data.*

**Public Member Functions**

- virtual `eCharPathType getCharPathType ()=0`
  
  *Retrieves the type of charpath rendering needed which can be Stroke, Clip or Fill.*

- virtual `bool setCharPathType (eCharPathType charPathType)=0`
  
  *Sets the type of charpath rendering needed which can be Stroke, Clip or Fill.*

- virtual `bool getStrokePath (IDOMPathNodePtr &strokePath)=0`
  
  *Gets the equivalent path that the charpath makes.*

- virtual `bool setStrokePath (IDOMPathNodePtr &strokePath)=0`
  
  *Sets the equivalent path that the charpath makes.*
virtual bool getClippedGroup (IDOMGroupPtr &clippedGroup)=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 bool setClippedGroup (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.

virtual bool getFill (IDOMBrushPtr &brush)=0  

Gets the brush that is used to fill the charpath when the charpath type is Fill.

virtual bool setFill (IDOMBrushPtr &brush)=0  

Sets the brush that is used to fill the charpath when the charpath type is Fill.

virtual eBlendMode getBlendMode () const =0  

Get the blend mode to be used for rendering this char path group.

virtual bool setBlendMode (eBlendMode blendMode)=0  

Set the blend mode to be used for rendering this char path group.

Static Public Member Functions

- static const CClassID & classID ()  

Retrieves class id of IDOMGroup.

Additional Inherited Members

8.132.1 Detailed Description

IDOMCharPathGroup interface.

8.132.2 Member Function Documentation

8.132.2.1 classID()

static const CClassID& IDOMCharPathGroup::classID ( ) [inline], [static]

Retrieves class id of IDOMGroup.

Returns

CClassID class id of the element
8.132.2.2 getBlendMode()

virtual eBlendMode IDOMCharPathGroup::getBlendMode() const [pure virtual]

Get the blend mode to be used for rendering this char path group.

Returns
blendMode. The blend mode.

8.132.2.3 getCharPathType()

virtual eCharPathType IDOMCharPathGroup::getCharPathType() [pure virtual]

Retrieves the type of charpath rendering needed which can be Stroke, Clip or Fill.

Returns
eCharPathType Returns the type of charpath rendering

8.132.2.4 getClippedGroup()

virtual bool IDOMCharPathGroup::getClippedGroup(
    IDOMGroupPtr & clippedGroup) [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.

Parameters

clippedGroup Smart pointer to group of objects that should be clipped

Returns
bool Returns true on success

8.132.2.5 getFill()

virtual bool IDOMCharPathGroup::getFill(
    IDOMBrushPtr & brush) [pure virtual]

Gets the brush that is used to fill the charpath when the charpath type is Fill.
Parameters

| brush         | Smart pointer to the brush used to fill the charpath |

Returns

bool. Returns true on success

8.132.2.6 getStrokePath()

virtual bool IDOMCharPathGroup::getStrokePath (IDOMPathNodePtr & strokePath) [pure virtual]

Gets the equivalent path that the charpath makes.

Parameters

| strokePath | Smart pointer to stroke path of the charpath |

Returns

bool. Returns true on success

8.132.2.7 setBlendMode()

virtual bool IDOMCharPathGroup::setBlendMode (eBlendMode blendMode) [pure virtual]

Set the blend mode to be used for rendering this char path group.

Parameters

| blendMode     | The desired blend mode. |

Returns

bool. Returns true on success, false if the call fails.

8.132.2.8 setCharPathType()

virtual bool IDOMCharPathGroup::setCharPathType (eCharPathType charPathType) [pure virtual]
Sets the type of charpath rendering needed which can be Stroke, Clip or Fill.
Parameters

| charPathType | The type of charpath |

Returns

bool Returns true on success

8.132.2.9 setClippedGroup()

virtual bool IDOMCharPathGroup::setClippedGroup (IDOMGroupPtr & clippedGroup) [pure virtual]

Sets the objects that need to be drawn that must be clipped by the charpath Only used when the charpath type is Clip.

Parameters

| clippedGroup | Smart pointer to group of objects that should be clipped |

Returns

bool Returns true on success

8.132.2.10 setFill()

virtual bool IDOMCharPathGroup::setFill (IDOMBrushPtr & brush) [pure virtual]

Sets the brush that is used to fill the charpath when the charpath type is Fill.

Parameters

| brush | Smart pointer to the brush used to fill the charpath |

Returns

bool Returns true on success

8.132.2.11 setStrokePath()

virtual bool IDOMCharPathGroup::setStrokePath (IDOMPathNodePtr & strokePath) [pure virtual]
Sets the equivalent path that the charpath makes.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>strokePath</code></td>
<td>Smart pointer to stroke path of the charpath</td>
</tr>
</tbody>
</table>

**Returns**

`bool` Returns true on success

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

- `idomcharpathgroup.h`

### 8.133 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:

![Inheritance Diagram](image)

**Public Member Functions**

- `virtual float getAlpha () const =0`
  
  Retrieves the alpha channel value. The alpha channel value specifies the transparency of the color.

- `virtual bool setAlpha (float a)=0`
  
  Sets the alpha channel value.

- `virtual bool getColorSpace (IDOMColorSpacePtr &colorSpace)=0`
  
  Retrieves the color space.
• virtual bool setColorSpace (const IDOMColorSpacePtr &colorSpace, bool setDefaultColor=true)=0
  Set the color space.
• virtual bool setColorSpace (const IDOMColorSpacePtr &colorSpace, eRenderingIntent intent, eBlackPoint Compensation bpc, IEDLClassFactory *pFactory)=0
  Set the color space, converting color values from the previous color space to the new.
• virtual bool getComponentValue (uint32 component, float &value)=0
  Retrieves the value of a component.
• virtual bool setComponentValue (uint32 component, float value)=0
  Sets a component value. The value will be clipped to the range of the color space, if known.

Static Public Member Functions

• static EDL_API IDOMColorPtr create (IEDLClassFactory *pFactory, const IDOMColorSpacePtr &space, float alpha,...)
  Simplified color creation routine. Throws an IEDLError on failure.
• static EDL_API IDOMColorPtrcreateFromVec (IEDLClassFactory *pFactory, const IDOMColorSpacePtr &space, float alpha, const CEDLVecto r< 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

8.133.1 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.

8.133.2 Member Function Documentation

8.133.2.1 classID()

static const CClassID & IDOMColor::classID ( ) [inline], [static]

Retrieves the class id of IDOMColor.

Returns

CClassID Returns the class id of the element.
8.133.2 create()

```cpp
static EDL_API IDOMColorPtr IDOMColor::create (  
    IEDLClassFactory * pFactory,  
    const IDOMColorSpacePtr & space,  
    float alpha,  
    ... ) [static]
```

Simplified color creation routine. Throws an IEDLError on failure.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pFactory</code></td>
<td>The EDL Class factory to use.</td>
</tr>
<tr>
<td><code>space</code></td>
<td>The color space to use.</td>
</tr>
<tr>
<td><code>alpha</code></td>
<td>The alpha to use.</td>
</tr>
<tr>
<td>...</td>
<td>The color components to use, as floats or doubles. The number of arguments MUST match the number of components expected of the color space.</td>
</tr>
</tbody>
</table>

**Returns**

`IDOMColorPtr` The new color.

8.133.2.3 createFromArray()

```cpp
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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pFactory</code></td>
<td>The EDL Class factory to use.</td>
</tr>
<tr>
<td><code>space</code></td>
<td>The color space to use.</td>
</tr>
<tr>
<td><code>alpha</code></td>
<td>The alpha to use.</td>
</tr>
<tr>
<td><code>components</code></td>
<td>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.</td>
</tr>
</tbody>
</table>

**Returns**

`IDOMColorPtr` The new color.
### 8.133.2.4 createFromVect()

```c
class 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pFactory</code></td>
<td>The EDL Class factory to use.</td>
</tr>
<tr>
<td><code>space</code></td>
<td>The color space to use.</td>
</tr>
<tr>
<td><code>alpha</code></td>
<td>The alpha to use.</td>
</tr>
<tr>
<td><code>components</code></td>
<td>A vector of color components to use, as floats. The number of components MUST match the number of components expected of the color space.</td>
</tr>
</tbody>
</table>

#### Returns

- `IDOMColorPtr` The new color.

### 8.133.2.5 getAlpha()

```c
class 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

### 8.133.2.6 getColorSpace()

```c
class virtual bool IDOMColor::getColorSpace ( 
    IDOMColorSpacePtr & colorSpace ) [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.
Parameters

| colorSpace | Reference parameter to receive a smart pointer to the color space. |

Returns

bool True on success

8.133.2.7 getComponentValue()

virtual bool IDOMColor::getComponentValue ( uint32 component, float & value ) [pure virtual]

Retrieves the value of a component.

Parameters

| component | Index of the component. |
| value     | Reference to receive the value. |

Returns

bool True on success, false if the call fails.

8.133.2.8 setAlpha()

virtual bool IDOMColor::setAlpha ( float a ) [pure virtual]

Sets the alpha channel value.

Parameters

| a | Desired new alpha channel value. |

Returns

@ bool True on success, false if the call fails.
8.133.9  setColorSpace()  [1/2]

virtual bool 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

<table>
<thead>
<tr>
<th>colorSpace</th>
<th>Smart pointer to the desired new color space</th>
</tr>
</thead>
<tbody>
<tr>
<td>setDefaultColor</td>
<td>If true, and the new space has a different number of components than the previous space, a default color will be set.</td>
</tr>
</tbody>
</table>

Returns

bool True on success

8.133.10  setColorSpace()  [2/2]

virtual bool 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 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

<table>
<thead>
<tr>
<th>colorSpace</th>
<th>Smart pointer to the desired new color space.</th>
</tr>
</thead>
<tbody>
<tr>
<td>intent</td>
<td>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.</td>
</tr>
<tr>
<td>bpc</td>
<td>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.</td>
</tr>
<tr>
<td>pFactory</td>
<td>The EDL class factory.</td>
</tr>
</tbody>
</table>
Returns

bool True on success, false if the call fails.

8.133.2.11 setComponentValue()

virtual bool 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

<table>
<thead>
<tr>
<th>component</th>
<th>Index of the component.</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>New value of the component.</td>
</tr>
</tbody>
</table>

Returns

bool True on success, false if the call fails.

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

- idomcolor.h

8.134 IDOMColorSpace Class Reference

IDOMColorSpace interface.

#include <idomcolorspace.h>
Public Types

- enum eColorSpaceType {
  eDeviceRGB, eDeviceGray, eDeviceCMYK, esRGB,
  esGray, escRGB, eICCBased, eIndexed,
  eDeviceN, eLAB, eDeviceCMY }

  Color spaces type enumeration.

Public Member Functions

- virtual eColorSpaceType getColorSpaceType ()=0

  Retrieves the color space type.

- virtual int 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 bool equals (const IDOMColorSpacePtr &colorSpace)=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.

Additional Inherited Members

8.134.1 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 IDELError exceptions on failure.

8.134.2 Member Enumeration Documentation

8.134.2.1 eColorSpaceType

enum IDOMColorSpace::eColorSpaceType

Color spaces type enumeration.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eDeviceRGB</td>
<td>Device RGB color space.</td>
</tr>
<tr>
<td>eDeviceGray</td>
<td>Device Gray color space.</td>
</tr>
<tr>
<td>eDeviceCMYK</td>
<td>Device CMYK color space.</td>
</tr>
<tr>
<td>esRGB</td>
<td>XPS sRGB color space with input range scaled to between 0 and 1.</td>
</tr>
<tr>
<td>esGray</td>
<td>Gray with sRGB gamma color space.</td>
</tr>
<tr>
<td>escRGB</td>
<td>XPS scRGB color space.</td>
</tr>
<tr>
<td>eICCBased</td>
<td>Color space with an ICC profile.</td>
</tr>
<tr>
<td>eIndexed</td>
<td>A color space analogous to the PostScript language/PDF Indexed color space, 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.</td>
</tr>
<tr>
<td>eDeviceN</td>
<td>A color space analogous to the PostScript language/PDF DeviceN/Separation color spaces as described in section 4.5.5 of the PDF 1.7 Reference Manual.</td>
</tr>
<tr>
<td>eLAB</td>
<td>Lab color space as described in section 4.5.4 of the PDF 1.7 Reference Manual.</td>
</tr>
<tr>
<td>eDeviceCMY</td>
<td>Device CMY color space to support PCL input. Internal use only.</td>
</tr>
</tbody>
</table>

Generated by Doxygen
8.134.3 Member Function Documentation

8.134.3.1 equals()

virtual bool IDOMColorSpace::equals ( 
    const IDOMColorSpacePtr & colorSpace ) [pure virtual]

Determines if the given color space is equivalent to this color space.

Parameters

\textbf{colorSpace} \hspace{1em} \text{The color space to compare with.}

Returns

\textbf{bool} \hspace{1em} \text{True if equal, false otherwise.}

8.134.3.2 getColorSpaceType()

virtual eColorSpaceType IDOMColorSpace::getColorSpaceType ( ) [pure virtual]

Retrieves the color space type.

See also

\textbf{eColorSpaceType}

Returns

\textbf{eColorSpaceType}. Retrieves the color space type

8.134.3.3 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

<table>
<thead>
<tr>
<th>component</th>
<th>The index of the component whose range is being queried</th>
</tr>
</thead>
<tbody>
<tr>
<td>low</td>
<td>Reference parameter to receive the low bound of the component's value range.</td>
</tr>
<tr>
<td>high</td>
<td>Reference parameter to receive the high bound of the component's value range.</td>
</tr>
</tbody>
</table>

Returns

bool Returns true if the given component has a range, false otherwise.

8.134.3.4 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.

8.134.3.5 getDefaultRenderingIntent()

virtual eRenderingIntent IDOMColorSpace::getDefaultRenderingIntent ( ) [pure virtual]

Retrieves the default rendering intent for this color space.

Returns

eRenderingIntent The default rendering intent.

8.134.3.6 getNumComponents()

virtual int IDOMColorSpace::getNumComponents ( ) [pure virtual]

Retrieves the number of components that are in colors in this color space.

Returns

int The number of components for colors in this color space.

8.134.3.7 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>colorSpace</td>
<td>The color space to compare with.</td>
</tr>
<tr>
<td>intent</td>
<td>The rendering intent to use.</td>
</tr>
<tr>
<td>bpc</td>
<td>The black point compensation treatment. If in double, use eBPCDefault.</td>
</tr>
</tbody>
</table>

Returns

```cpp
bool True if similar, false otherwise.
```

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

```
- idomcolorspace.h
```

## 8.135 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:

![Inheritance Diagram](attachment://inheritance.png)

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

8.135.1 Detailed Description

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

8.135.2 Member Function Documentation

8.135.2.1 classID()

static const CClassID & IDOMColorSpaceDeviceCMY::classID ( )  [inline],  [static]

Retrieves class id of IDOMColorSpaceDeviceCMY.

Returns

CClassID Class id of the element

8.135.2.2 create()

static EDL_API IDOMColorSpaceDeviceCMYPtr IDOMColorSpaceDeviceCMY::create ( IEDLClassFactory * pFactory )  [static]

Create a DeviceCMY Color Space. Throws an IEDLError on failure.

Parameters

| pFactory | The EDL Class factory to use. |

Returns

IDOMColorSpaceDeviceCMYPtr The new color space.

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

- idomcolorspece.h

8.136 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

8.136.1 Detailed Description

Represents the default CMYK color space.

8.136.2 Member Function Documentation
8.136.2.1 classID()

static const CClassID & IDOMColorSpaceDeviceCMYK::classID() [inline], [static]

Retrieves class id of IDOMColorSpaceDeviceCMYK.

Returns
   CClassID Class id of the element

8.136.2.2 create()

static EDL_API IDOMColorSpaceDeviceCMYKPtr IDOMColorSpaceDeviceCMYK::create (IEDLClassFactory ∗ pFactory) [static]

Create a DeviceCMYK Color Space. Throws an IEDLError on failure.

Parameters
   pFactory The EDL Class factory to use.

Returns
   IDOMColorSpaceDeviceCMYKPtr The new color space.

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

- idomcolorspace.h

8.137 IDOMColorSpaceDeviceGray Class Reference

IDOMColorSpaceDeviceGray interface.

#include <idomcolorspace.h>
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

8.137.1 Detailed Description
IDOMColorSpaceDeviceGray interface.

8.137.2 Member Function Documentation

8.137.2.1 classID()  

static const CClassID & IDOMColorSpaceDeviceGray::classID () [inline], [static]
Retrieves class id of IDOMColorSpaceDeviceGray.
Returns
  CClassID Class id of the element
8.137.2.2 create()

static EDL_API IDOMColorSpaceDeviceGrayPtr IDOMColorSpaceDeviceGray::create( IEDLClassFactory * pFactory ) [static]

Create a DeviceGray Color Space. Throws an IEDLError on failure.

Parameters

| pFactory | The EDL Class factory to use. |

Returns

IDOMColorSpaceDeviceGrayPtr The new color space.

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

- idomcolorspace.h

8.138 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 ()=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 IDOMFunctionPtr getTintTransform ()=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)=0
  
  *Get the colorant information for a component at the given index. An exception of type IEDLError is thrown on failure.*

- virtual IDOMColorSpacePtr getProcessColorSpace ()=0
  
  *Get the Process color space associated with this DeviceN space. (See Table 4.2.2 of the PDF 1.7 Reference). Optional.*

- virtual const CEDLVector<EDLSysString> & getProcessComponentNames ()=0
  
  *Get the vector of Process component names to those in the associated process color space (see Table 4.2.2 of the PDF 1.7 reference). An exception of IEDLError is thrown on failure, which should not happen.*

- virtual const CEDLVector<EDLSysString> & getPrintingOrder ()=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.*

Static Public Member Functions

- static EDL_API IDOMColorSpaceDeviceNPtr create (IEDLClassFactory *pFactory, const CEDLVector<EDLSysString> &colorants, const IDOMColorSpacePtr &alternate, const IDOMFunctionPtr &tintTransform, const IDOMColorSpacePtr &processColorSpace=NULL)
  
  *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=NULL, const CEDLVector<EDLSysString> &processComponents=NULL, const CEDLVector<EDLSysString> &printingOrder=NULL, 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=NULL)
  
  *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

8.138.1 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
8.138 IDOMColorSpaceDeviceN Class Reference

8.138.2 Member Function Documentation

8.138.2.1 classID()

static const CClassID & IDOMColorSpaceDeviceN::classID ( ) [inline], [static]

Retrieves class id of IDOMColorSpaceDeviceN.

Returns

CClassID Class id of the element

8.138.2.2 create() [1/3]

static EDL_API IDOMColorSpaceDeviceNPtr IDOMColorSpaceDeviceN::create (IEDLClassFactory * pFactory,
const CEDLVector< EDLSysString > & colorants,
const IDOMColorSpacePtr & alternate,
const IDOMFunctionPtr & tintTransform,
const IDOMColorSpacePtr & processColorSpace = IDOMColorSpacePtr(NULL) ) [static]

Create a simple DeviceN Color Space. Throws an IEDLError on failure.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pFactory</td>
<td>The EDL Class factory to use.</td>
</tr>
<tr>
<td>colorants</td>
<td>A vector of colorant names for the device inks</td>
</tr>
<tr>
<td>alternate</td>
<td>The alternate color space.</td>
</tr>
<tr>
<td>tintTransform</td>
<td>A function that maps colors in the space to the alternate color space.</td>
</tr>
<tr>
<td>processColorSpace</td>
<td>The PDF process color space associated with this DeviceN color space. Optional.</td>
</tr>
</tbody>
</table>

Returns

IDOMColorSpaceDeviceN The new color space.

8.138.2.3 create() [2/3]

static EDL_API IDOMColorSpaceDeviceNPtr IDOMColorSpaceDeviceN::create (IEDLClassFactory * pFactory,
const CDeviceNColorantVect & colorants,
const IDOMColorSpacePtr & alternate,
const IDOMFunctionPtr & tintTransform,
const IDOMColorSpacePtr & processColorSpace = IDOMColorSpacePtr(NULL),
const CEDLVector< EDLSysString > & processComponents = CEDLVector< EDLSysString >(),
const CEDLVector< EDLSysString > & printingOrder = CEDLVector< EDLSysString >(),
bool isNChannel = false ) [static]

Create a DeviceN Color Space. Throws an IEDLError on failure.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pFactory</td>
<td>The EDL Class factory to use.</td>
</tr>
<tr>
<td>colorants</td>
<td>A vector of smart pointers to IDOMDeviceNColorant objects</td>
</tr>
<tr>
<td>alternate</td>
<td>The alternate color space.</td>
</tr>
<tr>
<td>tintTransform</td>
<td>A function that maps colors in the space to the alternate color space.</td>
</tr>
<tr>
<td>processColorSpace</td>
<td>The PDF process color space associated with this DeviceN color space. Optional.</td>
</tr>
<tr>
<td>processComponents</td>
<td>A vector containing a list of process components. Optional</td>
</tr>
<tr>
<td>printingOrder</td>
<td>A vector containing a list defining the printing order. Optional</td>
</tr>
<tr>
<td>isNChannel</td>
<td>Specifies if the new color space is NChannel. Optional</td>
</tr>
</tbody>
</table>

Returns

IDOMColorSpaceDeviceN The new color space.

8.138.2.4 create() [3/3]

static EDL_API IDOMColorSpaceDeviceNPtr IDOMColorSpaceDeviceN::create (  
    IEDLClassFactory * pFactory,
    const CColorantInfoVect & colorants,
    const IDOMColorSpacePtr & alternate,
    const IDOMColorSpacePtr & processColorSpace = IDOMColorSpacePtr(NULL) ) [static]

Create a simple DeviceN Color Space, with automatic generation of the tint transform function. Throws an IEDLError on failure.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pFactory</td>
<td>The EDL Class factory to use.</td>
</tr>
<tr>
<td>colorants</td>
<td>A vector of CColorantInfo classes, specifying the name of each colorant, and the color components of that colorant in the alternate color space.</td>
</tr>
<tr>
<td>alternate</td>
<td>The alternate color space.</td>
</tr>
<tr>
<td>processColorSpace</td>
<td>The PDF process color space associated with this DeviceN color space. Optional.</td>
</tr>
</tbody>
</table>

Returns

IDOMColorSpaceDeviceN The new color space.
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.

---

Get the colorant information for a component at the given index. An exception of type `IEDLError` is thrown on failure.

**Parameters**

- `component` The 0 base component index of the desired colorant

**Returns**

- `IDOMDeviceNColorantPtr` The colorant information.

---

Get if the color space represents an NChannel color space.

**Returns**

- `bool` True if an NChannel color space

---

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**

- `CEDLVector<EDLSysString>` The printing order component names vector.
8.138.2.9 getProcessColorSpace()

virtual IDOMColorSpacePtr IDOMColorSpaceDeviceN::getProcessColorSpace ( ) [pure virtual]

Get the Process color space associated with this DeviceN space. (See Table 4.2.2 of the PDF 1.7 Reference). Optional.

Returns

IDOMColorSpacePtr The process color space if present, NULL otherwise.

8.138.2.10 getProcessComponentNames()

virtual const CEDLVector<EDLSysString>& IDOMColorSpaceDeviceN::getProcessComponentNames ( ) [pure virtual]

Get the vector of Process component names to those in the associated process color space (see Table 4.2.2 of the PDF 1.7 reference). An exception of IEDLError is thrown on failure, which should not happen.

Returns

CEDLVector<EDLSysString> The component names vector.

8.138.2.11 getTintTransform()

virtual IDOMFunctionPtr IDOMColorSpaceDeviceN::getTintTransform ( ) [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
IDOMColorSpaceDeviceRGB interface.

#include <idomcolorsapce.h>

Inheritance diagram for IDOMColorSpaceDeviceRGB:

Static Public Member Functions

- static EDL_API IDOMColorSpaceDeviceRGBPtr create (IEDLClassFactory *pFactory)
  Create a DeviceGray Color Space. Throws an IEDLError on failure.
- static const CClassID & classID ()
  Retrieves class id of IDOMColorSpaceDeviceRGB.

Additional Inherited Members

8.139.1 Detailed Description

IDOMColorSpaceDeviceRGB interface.

8.139.2 Member Function Documentation
8.139.2.1  classID()

static const CClassID & IDOMColorSpaceDeviceRGB::classID () [inline], [static]

Retrieves class id of IDOMColorSpaceDeviceRGB.

Returns

CClassID Class id of the element

8.139.2.2  create()

static EDL_API IDOMColorSpaceDeviceRGBPtr IDOMColorSpaceDeviceRGB::create (IEDLClassFactory * pFactory) [static]

Create a DeviceGray Color Space. Throws an IEDLError on failure.

Parameters

| pFactory | The EDL Class factory to use. |

Returns

IDOMColorSpaceDeviceRGB The new color space.

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

- idomcolorsparse.h

8.140  IDOMColorSpaceICCBased Class Reference

Represents a color space described by an ICC profile.

#include <idomcolorsparse.h>
Inheritance diagram for IDOMColorSpaceICCBased:

![Inheritance Diagram](image)

### 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.*

### Static Public Member Functions

- **static EDL_API IDOMColorSpaceICCBasedPtr create (IEDLClassFactory *pFactory, const IDOMICCProfilePtr &profile, const IDOMColorSpacePtr &alternate=IDOMColorSpacePtr(NULL))**
  
  *Creation function for an IDOMColorSpaceICCBased color space. Throws an IEDLError exception on failure.*

- **static const CClassID & classID ()**
  
  *Retrieves class id of IDOMColorSpaceICCBased.*
8.140.1 Detailed Description

Represents a color space described by an ICC profile.

8.140.2 Member Function Documentation

8.140.2.1 classID()

static const CClassID & IDOMColorSpaceICCBased::classID ( ) [inline], [static]

Retrieves class id of IDOMColorSpaceICCBased.

Returns

    CClassID Class id of the element

8.140.2.2 create()

static EDL_API IDOMColorSpaceICCBasedPtr IDOMColorSpaceICCBased::create ( 
    IEDLClassFactory * pFactory,  
    const IDOMICCProfilePtr & profile,  
    const IDOMColorSpacePtr & alternate = IDOMColorSpacePtr(NULL) ) [static]

Creation function for an IDOMColorSpaceICCBased color space. Throws an IEDLError exception on failure.

Parameters

| pFactory | The class factory. |
| profile | The profile to use |
| alternate | The alternate color space, if any. |

Returns

    IDOMColorSpaceICCBasedPtr The new color space.

8.140.2.3 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.

### 8.140.2.4 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 **IEDLErr** is thrown on failure.

Returns

**IDOMICCProfilePtr** The profile.

### 8.140.2.5 setICCProfile()

virtual void IDOMColorSpaceICCBased::setICCProfile(const IDOMICCProfilePtr & profile) [pure virtual]

Set the ICC profile for the color space.

Parameters

<table>
<thead>
<tr>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>profile</td>
<td>A reference to the new ICC profile.</td>
</tr>
</tbody>
</table>

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

- idomcolorspace.h

---

### 8.141 IDOMColorSpaceIndexed Class Reference

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

```c
#include <idomcolorspace.h>
```
Inheritance diagram for IDOMColorSpaceIndexed:

```
IRObject

IDOMHashable  IEDLObject

IDOMColorSpace

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.

Static Public Member Functions

- static EDL_API IDOMColorSpaceIndexedPtr create (IEDLClassFactory ∗pFactory, const IDOMColorSpace←Ptr &underlying, const IDOMFunctionPtr &mappingFunction)
  
  Create an Indexed Color Space. Throws an IEDLError on failure.

- static EDL_API IDOMColorSpaceIndexedPtr create (IEDLClassFactory ∗pFactory, const IDOMColorSpace←Ptr &underlying, const CEDLVector< uint8 > &table)
  
  Create an Indexed Color Space from an 8bpc table. Throws an IEDLError on failure.

- static const CClassID & classID ()
  
  Retrieves class id of IDOMColorSpaceICCBased.
Additional Inherited Members

8.141.1 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

8.141.2 Member Function Documentation

8.141.2.1 classID()

static const CClassID & IDOMColorSpaceIndexed::classID ( ) [inline], [static]

Retrieves class id of IDOMColorSpaceICCBased.

Returns

CClassID Class id of the element

8.141.2.2 create() [1/2]

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pFactory</td>
<td>The EDL Class factory to use.</td>
</tr>
<tr>
<td>underlying</td>
<td>The underlying color space.</td>
</tr>
<tr>
<td>mappingFunction</td>
<td>A function that maps indexes to colors in the underlying color space.</td>
</tr>
</tbody>
</table>
Returns

**IDOMColorSpaceIndexedPtr** The new color space.

### 8.141.2.3 create() [2/2]

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**

<table>
<thead>
<tr>
<th>pFactory</th>
<th>The EDL Class factory to use.</th>
</tr>
</thead>
<tbody>
<tr>
<td>underlying</td>
<td>The underlying color space.</td>
</tr>
<tr>
<td>table</td>
<td>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 [ &lt;R for index 0&gt;, &lt;G for index 0&gt;, &lt;B for index 0&gt;, &lt;R for index 1&gt;, &lt;G for index 1&gt; ... ]</td>
</tr>
</tbody>
</table>

Returns

**IDOMColorSpaceIndexedPtr** The new color space.

### 8.141.2.4 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.

### 8.141.2.5 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
This color space is as described in section 4.5.4 of the PDF 1.7 Reference Manual.

```cpp
#include <idomcolorspace.h>
```

Inheritance diagram for IDOMColorSpaceLAB:

```
IRCOBJECT

IDOMHashable

IEDLObject

IDOMColorSpace

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.*

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

8.142.1 Detailed Description

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

8.142.2 Member Function Documentation

8.142.2.1 classID()

static const CClassID & IDOMColorSpaceLAB::classID ( ) [inline], [static]

Retrieves class id of IDOMColorSpaceLAB.

Returns

CClassID Class id of the element

8.142.2.2 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

<table>
<thead>
<tr>
<th>pFactory</th>
<th>The EDL Class factory to use.</th>
</tr>
</thead>
<tbody>
<tr>
<td>whitePointX</td>
<td>The CIE X value for the white point.</td>
</tr>
<tr>
<td>whitePointY</td>
<td>The CIE Y value for the white point.</td>
</tr>
<tr>
<td>whitePointZ</td>
<td>The CIE V value for the white point.</td>
</tr>
<tr>
<td>blackPointX</td>
<td>The CIE X value for the black point.</td>
</tr>
<tr>
<td>blackPointY</td>
<td>The CIE Y value for the black point.</td>
</tr>
<tr>
<td>blackPointZ</td>
<td>The CIE V value for the black point.</td>
</tr>
<tr>
<td>rangeALo</td>
<td>The low bound of the A parameter</td>
</tr>
<tr>
<td>rangeAHi</td>
<td>The high bound of the A parameter</td>
</tr>
<tr>
<td>rangeBLo</td>
<td>The low bound of the B parameter</td>
</tr>
<tr>
<td>rangeBHi</td>
<td>The high bound of the B parameter</td>
</tr>
</tbody>
</table>
8.142 IDOMColorSpaceLAB Class Reference

Returns

IDOMColorSpaceLABPtr The new color space.

8.142.2.3 getBlackPoint()

virtual void IDOMColorSpaceLAB::getBlackPoint (  
    float & x,  
    float & y,  
    float & z )  [pure virtual]

Get the Black Point in CIE 1931 XYZ space.

Parameters

<table>
<thead>
<tr>
<th>x</th>
<th>Reference to receive the black point X value</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>Reference to receive the black point Y value</td>
</tr>
<tr>
<td>z</td>
<td>Reference to receive the black point Z value</td>
</tr>
</tbody>
</table>

8.142.2.4 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

<table>
<thead>
<tr>
<th>lowA</th>
<th>Reference to receive the low bound of the A parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>highA</td>
<td>Reference to receive the high bound of the A parameter</td>
</tr>
<tr>
<td>lowB</td>
<td>Reference to receive the low bound of the B parameter</td>
</tr>
<tr>
<td>highB</td>
<td>Reference to receive the high bound of the B parameter</td>
</tr>
</tbody>
</table>

8.142.2.5 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

8.143 IDOMColorSpacescRGB Class Reference

Represents the scRGB color space.

```cpp
#include <idomcolorspace.h>
```

Inheritance diagram for IDOMColorSpacescRGB:

```
  IRObject
    IDOMHashable
      IDOMColorSpace
        IDOMColorSpacescRGB
      IEDLObject
```

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

8.143.1 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.

8.143.2 Member Function Documentation

8.143.2.1 classID()

static const CClassID & IDOMColorSpacescRGB::classID ( ) [inline], [static]

Retrieves the class id of IDOMColorSpacescRGB.

Returns

CClassID Class id of the element

8.143.2.2 create()

static EDL_API IDOMColorSpacescRGBPtr IDOMColorSpacescRGB::create (  
IEDLClassFactory * pFactory ) [static]

Create an scRGB Color Space. Throws an IEDLError on failure.

Parameters

| pFactory | The EDL Class factory to use |

Returns

IDOMColorSpacescRGBPtr The new color space

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

- idomcolorspace.h
8.144  IDOMColorSpacesGray Class Reference

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

```cpp
#include <idomcolorspace.h>
```

Inheritance diagram for IDOMColorSpacesGray:

![Inheritance Diagram]

### 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

8.144.1  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.

8.144.2  Member Function Documentation
8.144.2.1 classID()

```cpp
static const CClassID& IDOMColorSpacesGray::classID ( ) [inline], [static]
```

Retrieves class id of IDOMColorSpacesGray.

**Returns**

- **CClassID** Class id of the element

8.144.2.2 create()

```cpp
static EDL_API IDOMColorSpacesGrayPtr IDOMColorSpacesGray::create ( IEDLClassFactory * pFactory ) [static]
```

Create an sGray Color Space. Throws an IEDLError on failure.

**Parameters**

- **pFactory** The EDL Class factory to use.

**Returns**

- **IDOMColorSpacesGrayPtr** The new color space.

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

- idomcolorsapce.h

### 8.145 IDOMColorSpacesRGB Class Reference

Represents the RGB color space.

```cpp
#include <idomcolorsapce.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

8.145.1 Detailed Description

Represents the RGB color space.

8.145.2 Member Function Documentation

8.145.2.1 classID()

static const CClassID & IDOMColorSpacesRGB::classID () [inline], [static]

Retrieves the class id of IDOMColorSpacesRGB.

Returns

   CClassID Class id of the element
### 8.145.2.2 create()

```
static EDL_API IDOMColorSpacesRGBPtr IDOMColorSpacesRGB::create (IEDLClassFactory * pFactory) [static]
```

Create an sRGB Color Space. Throws an IEDLError on failure.

**Parameters**

- `pFactory` The EDL Class factory to use.

**Returns**

- `IDOMColorSpacesRGBPtr` The new color space.

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

- `idomcolorspace.h`

## 8.146 IDOMCompositeImage Class Reference

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

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

Inheritance diagram for IDOMCompositeImage:
Classes

- class Data
  
  *Initialization data.*

Public Member Functions

- virtual bool getStream (IInputStreamPtr &stream) const
  
  *This image type does not allow direct access to the underlying streams.*

- virtual bool setStream (const IInputStreamPtr &stream)
  
  *This image type does not allow direct access to the underlying streams.*

Static Public Member Functions

- static EDL_API IDOMCompositeImagePtr create (IEDLClassFactory ∗pFactory, const CEDLVector< IDOMImagePtr > componentImages)
  
  *Simplified creator for a IDOMCompositeImage. Throws an IEDLError on failure.*

- static const CClassID & classID ()
  
  *Retrieves class id of IDOMRecombineImage.*

Additional Inherited Members

8.146.1 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.

8.146.2 Member Function Documentation

8.146.2.1 classID()

static const CClassID& IDOMCompositeImage::classID () [inline], [static]

*Retrieves class id of IDOMRecombineImage.*

Returns

  CClassID Class id of the element

8.146.2.2 create()

static EDL_API IDOMCompositeImagePtr IDOMCompositeImage::create ( 
  IEDLClassFactory ∗pFactory,
  const CEDLVector< IDOMImagePtr > componentImages ) [static]

*Simplified creator for a IDOMCompositeImage. Throws an IEDLError on failure.*
Parameters

<table>
<thead>
<tr>
<th>pFactory</th>
<th>The EDL Class Factory</th>
</tr>
</thead>
<tbody>
<tr>
<td>componentImages</td>
<td>The images to be concatenated</td>
</tr>
</tbody>
</table>

Returns

IDOMCompositeImagePtr The combined result.

8.146.2.3 getStream()

virtual bool IDOMCompositeImage::getStream ( 
    IInputStreamPtr & stream ) const [inline], [virtual]

This image type does not allow direct access to the underlying streams.

Parameters

| stream | A smart pointer to the stream |

Returns

bool Always false

Implements IDOMResource.

8.146.2.4 setStream()

virtual bool IDOMCompositeImage::setStream ( 
    const IInputStreamPtr & stream ) [inline], [virtual]

This image type does not allow direct access to the underlying streams.

Parameters

| stream | A smart pointer to the stream |

Returns

bool Always false

Implements IDOMResource.

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

- idomimageresource.h

Generated by Doxygen
8.147  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>

8.147.1  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

8.148  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 EDLSysString & 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.

Static Public Member Functions

• static const CClassID & classID ()
  Retrieves class id for IDOMDeviceNColorant.

Additional Inherited Members

8.148.1 Detailed Description

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

8.148.2 Member Function Documentation

8.148.2.1 classID()

static const CClassID & IDOMDeviceNColorant::classID () [inline], [static]
Retrieves class id for IDOMDeviceNColorant.

Returns

  CClassID Class id of the element

8.148.2.2 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.
8.148.2.3 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.

8.148.2.4 getName()

virtual const EDLSysString& 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

EDLSysString The name of the colorant.

8.148.2.5 getSolidity()

virtual bool IDOMDeviceNColorant::getSolidity ( float & solidity ) [pure virtual]

Get the solidity of the colorant.

Parameters

| solidity | Reference to receive the solidity |

Returns

bool Returns true on success, false if the colorant has no solidity specified.

8.148.2.6 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

### 8.149 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.

```cpp
#include <idomfunction.h>
```

Inheritance diagram for IDOMExponentialFunction:

```
<table>
<thead>
<tr>
<th>IRObject</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDOMHashable</td>
</tr>
<tr>
<td>IEDLObject</td>
</tr>
<tr>
<td>IDOMFunction</td>
</tr>
<tr>
<td>IDOMExponentialFunction</td>
</tr>
</tbody>
</table>
```

### Classes

- class **Data**
  
  *Initialization data.*
Public Member Functions

- virtual bool getOutputC0 (int outputNum, float &c0)=0
  Get the output C0 value for a given output to the function.
- virtual bool getOutputC1 (int outputNum, float &c1)=0
  Get the C1 value for a given output to the function.
- virtual bool getExponent (float &n)=0
  Get the exponent (N) value for the function.
- virtual bool evaluate (float *inputValues, float *outputValues)=0
  Evaluate the input through the function and return the result.
- virtual bool evaluate (int *inputValues, float *outputValues)=0
  Evaluate the input through the function and return the result.

Static Public Member Functions

- static const CClassID & classID ()
  Retrieves class id of IDOM.

Additional Inherited Members

8.149.1 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.

8.149.2 Member Function Documentation

8.149.2.1 classID()

static const CClassID & IDOMExponentialFunction::classID () [inline], [static]

Retrieves class id of IDOM.

Returns

CClassID Class id of the element

8.149.2.2 evaluate() [1/2]

virtual bool IDOMExponentialFunction::evaluate ( 
  float * inputValues,
  float * outputValues ) [pure virtual]

Evaluate the input through the function and return the result.
Parameters

<table>
<thead>
<tr>
<th>inputValues</th>
<th>An array of floats that are input into the function. The size of the array must be the same as the required number of inputValues</th>
</tr>
</thead>
<tbody>
<tr>
<td>outputValues</td>
<td>An array of floats that are the result of evaluating the input through the function. The size of the array must be the same as the required number of inputValues</td>
</tr>
</tbody>
</table>

Returns

bool Returns true on success. False if there is an error evaluating the input.

Implements IDOMFunction.

8.149.2.3 evaluate()

virtual bool IDOMExponentialFunction::evaluate ( int * inputValues, float * outputValues ) [pure virtual]

Evaluate the input through the function and return the result.

Parameters

<table>
<thead>
<tr>
<th>inputValues</th>
<th>An array of integers that are input into the function. The size of the array must be the same as the required number of inputValues</th>
</tr>
</thead>
<tbody>
<tr>
<td>outputValues</td>
<td>An array of floats that are the result of evaluating the input through the function. The size of the array must be the same as the required number of inputValues</td>
</tr>
</tbody>
</table>

Returns

bool Returns true on success. False if there is an error evaluating the input.

Implements IDOMFunction.

8.149.2.4 getExponent()

virtual bool IDOMExponentialFunction::getExponent ( float & n ) [pure virtual]

Get the exponent (N) value for the function.

Parameters

n A reference to receive the exponent value
Returns

bool Returns true on success.

8.149.2.5 getOutputC0()

virtual bool IDOMExponentialFunction::getOutputC0 (  
    int outputNum,  
    float & c0          ) [pure virtual]

Get the output C0 value for a given output to the function.

Parameters

<table>
<thead>
<tr>
<th>outputNum</th>
<th>The 0-indexed output number</th>
</tr>
</thead>
<tbody>
<tr>
<td>c0</td>
<td>A reference to receive c0 value for the given outputNum</td>
</tr>
</tbody>
</table>

Returns

bool Returns true on success

8.149.2.6 getOutputC1()

virtual bool IDOMExponentialFunction::getOutputC1 (  
    int outputNum,  
    float & c1          ) [pure virtual]

Get the C1 value for a given output to the function.

Parameters

<table>
<thead>
<tr>
<th>outputNum</th>
<th>The 0-indexed input number</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>A reference to receive c1 value for the given outputNum</td>
</tr>
</tbody>
</table>

Returns

bool Returns true on success

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

- idomfunction.h
IDOMExternalTarget interface.

#include <idomtarget.h>

Inheritance diagram for IDOMExternalTarget:

![Inheritance Diagram](image)

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.

Additional Inherited Members

8.150.1 Detailed Description

IDOMExternalTarget interface.

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

8.150.2.1  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.

8.150.2.2  getTargetType()

virtual eTargetType IDOMExternalTarget::getType ( ) const [inline], [virtual]

Retrieves the target URI type.

Returns

  eTargetType. The target type specified by the eTargetType enumeration.

Implements IDOMTarget.

8.150.2.3  getTargetUri()

virtual EDLSSysString IDOMExternalTarget::getTargetUri ( ) const [pure virtual]

Retrieves the target URI.

Returns

  EDLSSysString. The function retrieves the target URI.

8.150.2.4  setIsMap()

virtual void IDOMExternalTarget::setIsMap ( bool isMap ) [pure virtual]

Sets the value of the IsMap flag.
Parameters

| isMap | The new value of IsMap flag. |

8.150.2.5 setTargetUri()

```cpp
toal void IDOMExternalTarget::setTargetUri {
    const EDLSysString & uri } [pure virtual]
```

Sets the target URI.

Parameters

| uri | The new target URI. |

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

- idomtarget.h

8.151 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.

```cpp
#include <idomimageresource.h>
```
Inheritance diagram for IDOMFilteredImage:

Classes

• class Data
  
  Initialization data.

Public Member Functions

• virtual bool pushFilter (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 bool getFilterAtIndex (uint32 index, IDOMImageFilterPtr &filter) 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 bool setStream (const IInputStreamPtr &stream)
  
  This image type does not allow direct access to the underlying streams.
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. Throws an IEDLError on failure.
- static EDL_API IDOMFilteredImagePtr create (IEDLClassFactory *pFactory, const IDOMImagePtr &sourceImage, const CEDLVector<IDOMImageFilterPtr>& filters)
  Simplified creator for a filtered image with a vector of filters. Throws an IEDLError on failure.
- static const CClassID & classID ()
  Retrieves class id of IDOMFilteredImage.

Additional Inherited Members

8.151.1 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.

8.151.2 Member Function Documentation

8.151.2.1 classID()

static const CClassID& IDOMFilteredImage::classID () [inline], [static]

Retrieves class id of IDOMFilteredImage.

Returns

CClassID Class id of the element

8.151.2.2 create() [1/2]

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. Throws an IEDLError on failure.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pFactory</td>
<td>The EDL Class Factory</td>
</tr>
<tr>
<td>sourceImage</td>
<td>The source image.</td>
</tr>
<tr>
<td>filter</td>
<td>The filter to use.</td>
</tr>
</tbody>
</table>
Returns

IDLFilteredImagePtr The filtered image.

8.151.2.3 create() [2/2]

static EDL_API IDOMFilteredImagePtr IDOMFilteredImage::create ( 
    IEDLClassFactory * pFactory, 
    const IDOMImagePtr & sourceImage, 
    const CEDLVector< IDOMImageFilterPtr > & filters ) [static]

Simplified creator for a filtered image with a vector of filters. Throws an IEDLError on failure.

Parameters

<table>
<thead>
<tr>
<th>pFactory</th>
<th>The EDL Class Factory</th>
</tr>
</thead>
<tbody>
<tr>
<td>sourceImage</td>
<td>The source image.</td>
</tr>
<tr>
<td>filters</td>
<td>The vector of filters to instantiate with.</td>
</tr>
</tbody>
</table>

Returns

IDLFilteredImagePtr The filtered image.

8.151.2.4 getFilterAtIndex()

virtual bool IDOMFilteredImage::getFilterAtIndex ( 
    uint32 index, 
    IDOMImageFilterPtr & filter ) const [pure virtual]

Get a filter from the filter chain at the given index.

Parameters

<table>
<thead>
<tr>
<th>index</th>
<th>Index of the desired filter in the chain (beginning at 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>Smart pointer to receive the filter.</td>
</tr>
</tbody>
</table>

Returns

bool True on success, false otherwise.

8.151.2.5 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.

8.151.2.6 getSourceImage()

virtual IDOMImagePtr IDOMFilteredImage::getSourceImage ( ) const [pure virtual]

Get the source image.

Returns

IDOMImagePtr Smart pointer to the source image.

8.151.2.7 getStream()

virtual bool IDOMFilteredImage::getStream ( IInputStreamPtr & stream ) const [inline], [virtual]

This image type does not allow direct access to the underlying streams.

Parameters

| stream | A smart pointer to the stream |

Returns

bool Always false

Implements IDOMResource.

8.151.2.8 pushFilter()

virtual bool IDOMFilteredImage::pushFilter ( IDOMImageFilterPtr & filter ) [pure virtual]

Push a filter onto the end of the internal filter chain.

Parameters

| filter | A pointer to the image filter to be added. |
Returns

`bool` True on success.

8.151.9 `setStream()`

```
virtual bool IDOMFilteredImage::setStream ( 
    const IInputStreamPtr & stream ) [inline], [virtual]
```

This image type does not allow direct access to the underlying streams.

Parameters

| stream | A smart pointer to the stream |

Returns

`bool` Always false

Implements `IDOMResource`.

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

- `idomimageresource.h`

8.152 `IDOMFixedPage` Class Reference

Represents `<FixedPage>` element.

```
#include <idompage.h>
```
Inheritance diagram for IDOMFixedPage:

```
IRCOBJECT
  IEDLOBJECT
    IDOMNODE
      IDOMJOBTOWNER
        IDOMFIXEDPAGE
```

**Classes**

- class **Data**  
  *Initialization data.*

**Public Member Functions**

- virtual bool **setWidth** (double width)=0  
  *Sets the fixed page width.*
- virtual double **getWidth** () const =0  
  *Retrieves the fixed page width.*
- virtual bool **setHeight** (double height)=0  
  *Sets the fixed page height.*
- virtual double **getHeight** () const =0  
  *Retrieves the fixed page height.*
- virtual bool **getContentBox** (FRect &bb) 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 bool **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 bool getBleedBox (FRect &bb) 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 bool 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 bool getCropBox (FRect &cropBox) 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 bool setCropBox (const FRect &cropBox)=0

Sets the fixed page's crop box.

virtual bool getTrimBox (FRect &trimBox) const =0

Retrieves the fixed page's trim box. The trim box defines the intended dimensions of the finished page after trimming.

virtual bool 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 bool getLanguage (EDLString &lang) 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 bool setLanguage (const EDLString &lang)=0

Sets the language of the fixed page and any of its children.

virtual bool getResourceDictionary (IDOMResourceDictionaryPtr &ptrResources) 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 bool setResourceDictionary (const IDOMResourceDictionaryPtr &ptrResources)=0

Sets the resource dictionary.

virtual bool getThumbnail (IDOMImagePtr &ptrThumbnail) const =0

Retrieves a smart pointer to the fixed document thumbnail image.

virtual bool 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.

virtual IDOMPageLinkManagerPtr getLinkManager ()=0

Retrieves the page's links manager.

virtual IDOMPageAnnotationManagerPtr getAnnotationManager ()=0

Retrieves the page's annotations manager.

virtual bool isDegenerated () const =0

Retrieves the page's degenerated flag value.

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.
Additional Inherited Members

8.152.1 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.

8.152.2 Member Function Documentation

8.152.2.1 classID()

    static const CClassID & IDOMFixedPage::classID ( ) [inline], [static]

Retrieves the class id of IDOMFixedPage.

Returns

    CClassID Class id of the IDOMFixedPage

8.152.2.2 create()

    static EDL_API IDOMFixedPagePtr IDOMFixedPage::create ( 
        IEDLClassFactory * pFactory, 
        double width = 793.7, 
        double height = 1122.5 ) [static]

Simplified creation function for IDOMFixedPage Throws an IEDLError exception on failure.

Parameters

    | pFactory | The EDL class factory |
    |----------|-----------------------|
    | width    | Description for width (defaults to 210mm, width of A4) |
    | height   | Description for height (defaults to 297mm, height of A4) |

Returns

    IDOMFixedPagePtr A smart pointer to the fixed page
8.152.2.3 getAnnotationManager()

virtual IDOMPageAnnotationManagerPtr IDOMFixedPage::getAnnotationManager () [pure virtual]

Retrieves the page's annotations manager.

Returns

IDOMPageAnnotationManagerPtr A smart pointer to the page's annotations manager.

8.152.2.4 getBleedBox()

virtual bool IDOMFixedPage::getBleedBox ( FRect & bb ) 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.

Parameters

bb A reference parameter to receive the fixed page's bleed box.

Returns

bool True on success, false if the call fails.

8.152.2.5 getContentBox()

virtual bool IDOMFixedPage::getContentBox ( FRect & bb ) 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.

Parameters

bb A reference parameter to receive the fixed page's content box

Returns

bool True on success, false if the call fails.
8.152.2.6  getCropBox()

virtual bool IDOMFixedPage::getCropBox (  
    FRect & cropBox ) 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.

Parameters

    cropBox  A reference parameter to receive the fixed page's crop box.

Returns

    bool  True on success, false if the call fails.

8.152.2.7  getHeight()

virtual double IDOMFixedPage::getHeight ( ) const  [pure virtual]

Retrieves the fixed page height.

Returns

    double  Returns the fixed page height

8.152.2.8  getLanguage()

virtual bool IDOMFixedPage::getLanguage (  
    EDLString & lang ) 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/

Parameters

    lang  A reference parameter which will receive the language of the fixed page.
Returns

**bool** True on success, false if the call fails.

---

8.152.2.9 getLinkManager()

virtual IDOMPageLinkManagerPtr IDOMFixedPage::getLinkManager() const [pure virtual]

Retrieves the page’s links manager.

Returns

**IDOMPageLinkManagerPtr** The page’s link manager

---

8.152.2.10 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.

---

8.152.2.11 getResourceDictionary()

virtual bool IDOMFixedPage::getResourceDictionary(
    IDOMResourceDictionaryPtr & ptrResources
) 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.

Parameters

| **ptrResources** | Reference parameter to be populated with a smart pointer to the resource dictionary |

Returns

**bool** True on success, false if the call fails.
8.152.2.12 getThumbnail()

virtual bool IDOMFixedPage::getThumbnail ( 
    IDOMImagePtr & ptrThumbnail ) const [pure virtual]

Retrieves a smart pointer to the fixed document thumbnail image.

Parameters

| ptrThumbnail | Smart pointer to recieve a reference to the thumbnail image. |

Returns

bool True on success, false if the call fails.

8.152.2.13 getTrimBox()

virtual bool IDOMFixedPage::getTrimBox ( 
    FRect & trimBox ) 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.

Parameters

| trimBox | A reference parameter to receive the fixed page's trim box. |

Returns

bool True on success, false if the call fails.

8.152.2.14 getWidth()

virtual double IDOMFixedPage::getWidth ( ) const [pure virtual]

Retrieves the fixed page width.

Returns

double Returns the fixed page width.
8.152.15  isDegenerated()

virtual bool IDOMFixedPage::isDegenerated( ) const [pure virtual]

Retrieves the page's degenerated flag value.

Returns
   bool True if page is degenerated

8.152.16  setBleedBox()

virtual bool IDOMFixedPage::setBleedBox( const FRect & bb ) [pure virtual]

Sets the fixed page's bleed box. The bleed box is the area, inclusive of crop marks, that extends outside of the
physical page. See Figure 8.11.

These values are specified in units of 1/96 inch.

Parameters
   bb  The new bleed box settings.

Returns
   bool True on success, false if the call fails.

8.152.17  setContentBox()

virtual bool 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
   bb  The new content box settings.
Returns

bool True on success, false if the call fails.

8.152.18 setCropBox()

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

Sets the fixed page's crop box.

Parameters

cropBox The new crop box settings.

Returns

bool True on success, false if the call fails.

8.152.19 setHeight()

virtual bool IDOMFixedPage::setHeight ( double height ) [pure virtual]

Sets the fixed page height.

The height of the page is expressed as a real number in units of 1/96th of an inch. If either the width or height of
the page (or both) is set to greater than the original page width or height, a larger page will result. This would have
more white space toward the right and/or bottom of the page when the document is viewed in an XPS viewer, since
the document origin is in the top-left corner. Changing the page dimensions has no effect on the coordinates of the
page content.

Parameters

height New fixed page height.

Returns

bool True on success, false if the call fails.

8.152.20 setLanguage()

virtual bool 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 \url{http://www.w3.org/International/articles/language-tags/}

**Parameters**

| lang | The new language setting. |

**Returns**

bool True on success, false if the call fails.

---

### 8.152.21 setPageGroup()  

```cpp
def setPageGroup(const IDOMTransparencyGroupPtr &pageGroup)
```

Set the page transparency group.

**Parameters**

| pageGroup | A reference to the desired transparency group, or NULL. |

---

### 8.152.22 setResourceDictionary()  

```cpp
def setResourceDictionary(const IDOMResourceDictionaryPtr &ptrResources)
```

Sets the resource dictionary.

**Parameters**

| ptrResources | Smart pointer to the new resource dictionary The resource dictionary contains the IDs of resources stored at this level of the document. |

**Returns**

bool True on success, false if the call fails.
8.152.2.23 setThumbnail()

virtual bool IDOMFixedPage::setThumbnail (  
   const IDOMImagePtr & ptrThumbnail ) [pure virtual]

Sets a new thumbnail image for the document sequence. The image must be in either JPEG or PNG format. Setting thumbnail to NULL deletes the document sequence thumbnail.

Parameters

| ptrThumbnail | Smart pointer to new thumbnail image. |

Returns

- **bool** True on success, false if the call fails.

8.152.2.24 setTrimBox()

virtual bool IDOMFixedPage::setTrimBox (  
   const FRect & trimBox ) [pure virtual]

Sets the fixed page's trim box. The trim box defines the intended dimensions of the finished page after trimming. These values are specified in units of 1/96 inch.

Parameters

| trimBox | The new trim box settings. |

Returns

- **bool** True on success, false if the call fails.

8.152.2.25 setWidth()

virtual bool IDOMFixedPage::setWidth (  
   double width ) [pure virtual]

Sets the fixed page width.

The width of the page is expressed as a real number in units of 1/96th of an inch. If either the width or height of the page (or both) is set to greater than the original page width or height, a larger page will result. This would have more white space toward the right and/or bottom of the page when the document is viewed in an XPS viewer, since the document origin is in the top-left corner. Changing the page dimensions has no effect on the coordinates of the page content.
Parameters

| width | New fixed page width |

Returns

**bool** True on success

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

- `idompage.h`

### 8.153 IDOMFont Class Reference

**IDOMFont** Base Class.

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

Inheritance diagram for IDOMFont:

![Inheritance diagram for IDOMFont](image)

**Classes**

- **class Data**
  
  *Initialization data.*
8.153 IDOMFont Class Reference

Public Types

- enum eFontType
type used to uniquely identify the type of font
- typedef uint32 CharCode
type used to uniquely identify a character code

Public Member Functions

- virtual eFontType getFontType () const =0
  Gets the font type. See eFontType for more information on font types.
- virtual bool getFontSource (IDOMFontSourcePtr &fontSource) const =0
  Get the font source of this font.
- virtual bool setFontSource (IDOMFontSourcePtr &fontSource)=0
  Sets the font source for this font.
- virtual bool getFontBaseStream (IInputStreamPtr &stream) const =0
  Return the base stream for this font, obtaining it from the font source.

Static Public Member Functions

- static const CClassID & classID ()
  Retrieves class id of IDOMFont.

Additional Inherited Members

8.153.1 Detailed Description

IDOMFont Base Class.

Requesting the stream from this resource will give you a useable font file related to the font type.

8.153.2 Member Function Documentation

8.153.2.1 classID()

static const CClassID & IDOMFont::classID ( ) [inline], [static]
Retrieves class id of IDOMFont.
Returns
  CClassID class id of the element

8.153.2.2 getFontBaseStream()

virtual bool IDOMFont::getFontBaseStream ( IInputStreamPtr &stream ) const [pure virtual]
Return the base stream for this font, obtaining it from the font source.
Parameters

* stream | Reference parameter to be populated with the retrieved font base stream.

Returns

  **bool** Function True on success, false if the call fails.

8.153.2.3 getFontSource()

virtual bool IDOMFont::setFontSource (IDOMFontSourcePtr & fontSource) const [pure virtual]

Get the font source of this font.

Parameters

* fontSource | Reference parameter to be populated with the retrieved font source.

Returns

  **bool** Function True on success, false if the call fails.

8.153.2.4 getFontType()

virtual eFontType IDOMFont::setFontType () const [pure virtual]

Gets the font type. See eFontType for more information on font types.

Returns

  eFontType. Returns the font type.

8.153.2.5 setFontSource()

virtual bool IDOMFont::setFontSource (IDOMFontSourcePtr & fontSource) [pure virtual]

Sets the font source for this font.
Parameters

| fontSource | The new font source |

Returns

bool Function True on success, false if the call fails.

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

- idomfont.h

8.154 IDOMFontOpenType Class Reference

IDOMFontOpenType interface.

#include <idomfont.h>

Inheritance diagram for IDOMFontOpenType:

![Inheritance Diagram]

Classes

- class Data
  
  Initialization data.

Generated by Doxygen
Class Documentation

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

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 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 bool getPostScriptName (IEDLClassFactory ∗pFactory, int32 fontIndex, EDLSysString &postScriptName)=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 bool getFullName (IEDLClassFactory ∗pFactory, int32 fontIndex, EDLSysString &fullName)=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 bool getFontOpenTypeTableAccessor (IFontOpenTypeTableAccessorPtr &accessor, uint32 fontIndex=0)=0  
  Creates an OpenType font table accessor.

- virtual bool getFontTrueTypeGlyphAccessor (IFontTrueTypeGlyphAccessorPtr &accessor, bool stripInstructions=false)=0  
  Creates a TrueType glyph accessor.

- virtual uint16 getFontLicenseFromOS2Table ()=0  
  Gets the IsType field (embedded licensing information) from the OS/2 table.

- virtual bool createTrueTypeOnlyFontVersion (const ISessionPtr &ptrSession, IDOMFontOpenTypeTTPtr &fontOpenTypeTT, bool regenerateStream)=0  
  Create a TrueType only font from this OpenType font that may contain CFF fonts.

- virtual bool createSubsetFont (const ISessionPtr &ptrSession, CEDLVector< uint16 > &usedGlyphs, CEDLVector< uint32 > &usedUnicode, IDOMFontOpenTypePtr &subsetFont)=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.
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.

Additional Inherited Members

8.154.1 Detailed Description

IDOMFontOpenType interface.

8.154.2 Member Enumeration Documentation

8.154.2.1 eOpenTypeFontType

enum IDOMFontOpenType::eOpenTypeFontType

Type used to uniquely identify the type of OpenType font.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eOpenTypeFontTypeUnknown</td>
<td>Unknown font type.</td>
</tr>
<tr>
<td>eOpenTypeFontTypeTTF</td>
<td>True type font.</td>
</tr>
<tr>
<td>eOpenTypeFontTypeCFF</td>
<td>Compact font format.</td>
</tr>
<tr>
<td>eOpenTypeFontTypeTTC</td>
<td>True type collections.</td>
</tr>
</tbody>
</table>

8.154.2.2 eOriginalFontType

enum IDOMFontOpenType::eOriginalFontType

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

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eOriginalTypeOpenType</td>
<td>Originally OpenType format.</td>
</tr>
<tr>
<td>eOriginalType1</td>
<td>Type 1 font.</td>
</tr>
<tr>
<td>eOriginalType2</td>
<td>Type 2 font - Usually a Type 1 encoded in a PDF in CFF form.</td>
</tr>
<tr>
<td>eOriginalType42</td>
<td>Type42/TrueType from PS/PDF.</td>
</tr>
<tr>
<td>eOriginalType9</td>
<td>CIDFontType0 (CIDFont based on Type 1 CharStrings)</td>
</tr>
<tr>
<td>eOriginalType11</td>
<td>CIDFontType2 (CIDFont based on TrueType glyph descriptions)</td>
</tr>
</tbody>
</table>
8.154.3 Member Function Documentation

8.154.3.1 classID()

static const CClassID & IDOMFontOpenType::classID ( ) [inline], [static]

Retrieves the class id of IDOMFontOpenType.

Returns

    CClassID Class id of the element

8.154.3.2 create()

static EDL_API IDOMFontOpenTypePtr IDOMFontOpenType::create (  
    IEDLClassFactory * pFactory,  
    const IInputStreamPtr & stream ) [static]

Simplified creator of a font from a stream. Throws an IEDLError exception on failure.

Parameters

| pFactory | EDL class factory to use. |
| stream   | The font stream. |

Returns

    IDOMFontOpenTypePtr the new font

8.154.3.3 createRenamedFont()

virtual IDOMFontOpenTypePtr IDOMFontOpenType::createRenamedFont (  
    const ISessionPtr & ptrSession,  
    const EDLSysString & fontName,  
    uint32 fontIndex = 0 ) [pure virtual]

Get a renamed version of this font. An exception of type IEDLError is thrown on failure.

Parameters

| ptrSession | A pointer to the EDL session. |
| fontName   | The desired font name |
| fontIndex  | The index of the font, if the font being renamed is a TrueType collection. The default is zero. |
8.154 IDOMFontOpenType Class Reference

Returns

IDOMFontOpenTypePtr The renamed font.

8.154.3.4 createSubsetFont()

virtual bool IDOMFontOpenType::createSubsetFont ( const ISessionPtr & ptrSession,
CEDLVector< uint16 > & usedGlyphs,
CEDLVector< uint32 > & usedUnicode,
IDOMFontOpenTypePtr & subsetFont ) [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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ptrSession</td>
<td>A pointer to the EDL session.</td>
</tr>
<tr>
<td>usedGlyphs</td>
<td>A vector of glyph IDs that should be retained in the subset font.</td>
</tr>
<tr>
<td>usedUnicode</td>
<td>A vector of unicode code points whose underlying glyphs should be retained in the subset font.</td>
</tr>
<tr>
<td>subsetFont</td>
<td>A reference to receive the new subset font</td>
</tr>
</tbody>
</table>

Returns

bool True on success, false if the call fails.

8.154.3.5 createTrueTypeOnlyFontVersion()

virtual bool IDOMFontOpenType::createTrueTypeOnlyFontVersion ( const ISessionPtr & ptrSession,
IDOMFontOpenTypeTTPtr & fontOpenTypeTT,
bool regenerateStream ) [pure virtual]

Create a TrueType only font from this OpenType font that may contain CFF fonts.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ptrSession</td>
<td>A pointer to the EDL session.</td>
</tr>
<tr>
<td>fontOpenTypeTT</td>
<td>The returned IDOMFontOpenTypeTT True-Type only Opentype font</td>
</tr>
<tr>
<td>regenerateStream</td>
<td>Determines whether the stream should be regenerated</td>
</tr>
</tbody>
</table>
8.154.3.6 getEmbedded()

virtual bool IDOMFontOpenType::getEmbedded ( ) const [pure virtual]

Establishes whether the font is flagged for embedding.

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

8.154.3.7 getFontLicenseFromOS2Table()

virtual uint16 IDOMFontOpenType::getFontLicenseFromOS2Table ( ) [pure virtual]

Gets the fsType field (embedded licensing information) from the OS/2 table.

Returns

uint16 The function returns the fsType field from the OS/2 table, or zero if the table does not exist in the font.

8.154.3.8 getFontOpenTypeTableAccessor()

virtual bool IDOMFontOpenType::getFontOpenTypeTableAccessor ( 
  IFontOpenTypeTableAccessorPtr & accessor,
  uint32 fontIndex = 0 ) [pure virtual]

Creates an OpenType font table accessor.

Parameters

<table>
<thead>
<tr>
<th>accessor</th>
<th>A reference parameter to receive the font table accessor.</th>
</tr>
</thead>
<tbody>
<tr>
<td>fontIndex</td>
<td>The index of the font - only used for TTC fonts.</td>
</tr>
</tbody>
</table>
Returns

bool True on success, false if the call fails.

8.154.3.9  getFontTrueTypeGlyphAccessor()

virtual bool IDOMFontOpenType::setFontTrueTypeGlyphAccessor (  
    IFontTrueTypeGlyphAccessorPtr & accessor,  
    bool stripInstructions = false ) [pure virtual]

Creates a TrueType glyph accessor.

Parameters

<table>
<thead>
<tr>
<th>accessor</th>
<th>A reference parameter to receive the glyph accessor</th>
</tr>
</thead>
<tbody>
<tr>
<td>stripInstructions</td>
<td>Determines whether instructions are stripped. Default is false</td>
</tr>
</tbody>
</table>

Returns

bool True on success, false if the call fails.

8.154.3.10  getFullName()

virtual bool IDOMFontOpenType::getFullName (  
    IEDLClassFactory * pFactory,  
    int32 fontIndex,  
    EDLSysString & fullName ) [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

<table>
<thead>
<tr>
<th>fontIndex</th>
<th>The index of the desired font (if a TrueType collection)</th>
</tr>
</thead>
<tbody>
<tr>
<td>pFactory</td>
<td>A pointer to an EDL Class factory.</td>
</tr>
<tr>
<td>fullName</td>
<td>A reference parameter to receive the font name.</td>
</tr>
</tbody>
</table>

Returns

bool True on success, false if the call fails.
8.154.3.11 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

8.154.3.12 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

8.154.3.13 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

8.154.3.14 getOpenTypeFontType()

virtual eOpenTypeFontType IDOMFontOpenType::getOpenTypeFontType ( ) [pure virtual]

Returns the sub font type for this opentype font.

Returns

eOpenTypeFontType The opentype font type
8.154.3.15  getOriginalFontType()

virtual eOriginalFontType IDOMFontOpenType::getOriginalFontType ( ) [pure virtual]

Returns the original font type for this opentype font.

Returns

  eOriginalFontType The original font type.

8.154.3.16  getPostScriptName()

virtual bool IDOMFontOpenType::getPostScriptName ( 
  IEDLClassFactory ∗ pFactory,
  int32 fontIndex,
  EDLSysString & postScriptName ) [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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pFactory</td>
<td>A pointer to an EDL Class factory.</td>
</tr>
<tr>
<td>fontIndex</td>
<td>The index of the desired font (if a TrueType collection)</td>
</tr>
<tr>
<td>postScriptName</td>
<td>A reference parameter to receive the font name.</td>
</tr>
</tbody>
</table>

Returns

  bool True on success, false if the call fails.

8.154.3.17  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.
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

| embedded | Set to true to flag the font for embedding |

Returns

`bool` True if font data is permitted to be embedded in the output

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

- `idomfont.h`

## 8.155 IDOMFontOTFTrueType Class Reference

Opentype Font.

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

### 8.155.1 Detailed Description

Opentype Font.

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

- `idomfont.h`

## 8.156 IDOMFontPCL5 Class Reference

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

```cpp
#include <idomfont.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.

8.156.1 Detailed Description

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

8.156.2 Member Function Documentation

8.156.2.1 classID()

static const CClassID & IDOMFontPCL5::classID () [inline], [static]
Retrieves class id of IDOMFontPCL5.

Returns

CClassID class id of the element

8.156.2.2 getFontName()

virtual bool IDOMFontPCL5::getName (EDLSysString &fontName) [pure virtual]

Returns the XL font name.
Parameters

<table>
<thead>
<tr>
<th>fontName</th>
<th>The font name returned</th>
</tr>
</thead>
</table>

Returns

**bool** True on success

### 8.156.2.3 getFontTrueTypeGlyphAccessor()

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

Creates a TrueType glyph accessor.

Parameters

<table>
<thead>
<tr>
<th>accessor</th>
<th>Returns the accessor</th>
</tr>
</thead>
</table>

Returns

**bool** True on success.

### 8.156.2.4 getFormat15FontBlockEnumerator()

```cpp
template bool IDOMFontPCL5::getFormat15FontBlockEnumerator (
    IFontPCL5WriteSegmentBlockEnumeratorPtr & enumerator
) [pure virtual]
```

Creates a Format 15 font block enumerator.

Parameters

<table>
<thead>
<tr>
<th>enumerator</th>
<th>Returns the enumerator</th>
</tr>
</thead>
</table>

Returns

**bool** True on success

### 8.156.2.5 getFormat16FontBlockEnumerator()

```cpp
template bool IDOMFontPCL5::getFormat16FontBlockEnumerator (
    IFontPCL5WriteSegmentBlockEnumeratorPtr & enumerator
) [pure virtual]
```

Creates a Format 16 font block enumerator.
Parameters

enumerator

Returns the enumerator

Returns

bool True on success

8.156.2.6 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.

8.156.2.7 getSymbolSetID()

virtual bool IDOMFontPCL5::getSymbolSetID ( EDLSysString & symbolSetID ) const [pure virtual]

Returns the font symbol set ID.

Parameters

symbolSetID The font symbol set ID

Returns

bool True on success

8.156.2.8 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
8.156.2.9  setSymbolSetIDCode()

virtual bool IDOMFontPCL5::setSymbolSetIDCode (IDOMFontPCL5::SymbolSetIDCode symbolSet) [pure virtual]

Set the font symbol set to be used in the generated font.

Parameters

symbolSet  The font symbol set

Returns

bool  True on success

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

- idomfont.h

8.157  IDOMFontPCLXL Class Reference

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

#include <idomfont.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.

8.157.1 Detailed Description

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

8.157.2 Member Enumeration Documentation

8.157.2.1 ePCLXLFontType

enum IDOMFontPCLXL::ePCLXLFontType

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

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ePCLXLFontTypeTTF</td>
<td>TrueType based font.</td>
</tr>
<tr>
<td>ePCLXLFontTypeBitmap</td>
<td>Bitmap font.</td>
</tr>
</tbody>
</table>

8.157.3 Member Function Documentation

8.157.3.1 classID()

static const CClassID & IDOMFontPCLXL::classID () [inline], [static]

Retrieves the class id of IDOMFontPCLXL.

Returns

  CClassID The class id of the element

8.157.3.2 getFontHeaderSegmentBlockEnumerator()

virtual bool IDOMFontPCLXL::getFontHeaderSegmentBlockEnumerator ( IFontHeaderWriteSegmentBlockEnumeratorPtr & enumerator ) [pure virtual]

Creates a PCL XL FontHeader enumerator of the font for the PCL XL ReadFontHeader operator.
Parameters

| enumerator | Reference parameter to receive a smart pointer to the enumerator. |

Returns

bool True on success, false if the call fails.

8.157.3.3 getFontName()

virtual bool IDOMFontPCLXL::setFontName (EDLSysString & fontName) [pure virtual]

Returns the XL font name. For example, "Times New Roman".

Parameters

| fontName | Reference parameter to receive a pointer to the font name. |

Returns

bool True on success, false if the call fails.

8.157.3.4 getFontTrueTypeGlyphAccessor()

virtual bool IDOMFontPCLXL::setFontTrueTypeGlyphAccessor (IFontPCLXLTrueTypeGlyphAccessorPtr & accessor) [pure virtual]

Creates a TrueType glyph accessor.

Parameters

| accessor | A reference parameter to receive the glyph accessor. |

Returns

bool True on success, false if the call fails.

8.157.3.5 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:

- idomfont.h

8.158 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:

![Inheritance Diagram](image)

Classes

- class Data
  
  *Initialization 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)*

- virtual bool determineUri (EDLSysString &uri) const
  
  *Determines the URI based on the font source (backwards compatible version)*
Static Public Member Functions

- static const CClassID & classID ()
  Retrieves the class id of IDOMFontSource.

Additional Inherited Members

8.158.1 Detailed Description

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

8.158.2 Member Enumeration Documentation

8.158.2.1 eFontSourceType


type used to uniquely identify the source type of a font

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eFontSourceTypeNone</td>
<td>Not specified.</td>
</tr>
<tr>
<td>eFontSourceTypeStreamFilter</td>
<td>Font filter (converter)</td>
</tr>
<tr>
<td>eFontSourceTypeStream</td>
<td>Font data stream.</td>
</tr>
<tr>
<td>eFontSourceTypeFont</td>
<td>Derived from a font.</td>
</tr>
</tbody>
</table>

8.158.3 Member Function Documentation

8.158.3.1 classID()

static const CClassID & IDOMFontSource::classID () [inline], [static]

Retrieves the class id of IDOMFontSource.

Returns

  CClassID The class id of the element
8.158.3.2 determineUri() [1/2]

virtual const EDLSysString& IDOMFontSource::determineUri ( ) const [pure virtual]

Determines the URI based on the font source (underlying font sources may be searched)

Returns

**EDLSysString** The uri, which will have a size of zero on failure.

Implemented in IDOMFontSourceObfuscationConverter, IDOMFontSourceFromStream, and IDOMFontSource<--StreamFilter.

8.158.3.3 determineUri() [2/2]

virtual bool IDOMFontSource::determineUri ( EDLSysString & uri ) const [inline], [virtual]

Determines the URI based on the font source (backwards compatible version)

Note: This is an alternate version of determineUri() given above. This version has been kept in source for backward compatibility.

Parameters

- **uri** Reference parameter to receive the determined URI.

Returns

**bool** True on success, or false if the call fails.

Reimplemented in IDOMFontSourceObfuscationConverter, IDOMFontSourceFromStream, and IDOMFontSource<--StreamFilter.

8.158.3.4 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
The source for IDOMFont when sourced from an existing stream.

#include <domfont.h>

Inheritance diagram for IDOMFontSourceFromStream:

Classes

- class Data
  
  Initialization data.

Public Member Functions

- virtual bool getStream (IInputStreamPtr &stream) const =0
  
  Return the actual stream.
- virtual bool getStreamLength (uint64 &length) 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 determineUri (EDLSysString &uri) const
  
  Determines the URI based on the font source (backwards compatible version)

Static Public Member Functions

- static const CClassID & classID ()
  
  Retrieves the class id of IDOMFontSourceFromStream.
Additional Inherited Members

8.159.1 Detailed Description

The source for IDOMFont when sourced from an existing stream.

8.159.2 Member Function Documentation

8.159.2.1 classID()

static const CClassID & IDOMFontSourceFromStream::classID ( ) [inline], [static]

Retrieves the class id of IDOMFontSourceFromStream.

Returns

CClassID Returns the class id of the element.

8.159.2.2 determineUri() [1/2]

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, which will have a size of zero on failure.

Implements IDOMFontSource.

8.159.2.3 determineUri() [2/2]

virtual bool IDOMFontSourceFromStream::determineUri ( EDLSysString & uri ) const [inline], [virtual]

Determines the URI based on the font source (backwards compatible version)

Note: This is an alternate version of determineUri() given above. This version has been kept in source for backward compatibility.
Parameters

| uri  | Reference parameter to receive the determined URI. |

Returns

`bool` True on success, or false if the call fails.

Reimplemented from `IDOMFontSource`.

### 8.159.2.4 `getStream()`

```cpp
virtual bool IDOMFontSourceFromStream::getStream ( IInputStreamPtr & stream ) const [pure virtual]
```

Return the actual stream.

Parameters

| stream | Reference parameter to receive a smart pointer to the actual stream. |

Returns

`bool` True on success, or false if the call fails.

### 8.159.2.5 `getStreamLength()`

```cpp
virtual bool IDOMFontSourceFromStream::getStreamLength ( uint64 & length ) const [pure virtual]
```

Retrieves the stream length, if possible.

Parameters

| length | Reference parameter to receive the stream length. |

Returns

`bool` True on success, or false if the call fails.

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

- `idomfont.h`
8.160  IDOMFontSourceObfuscationConverter Class Reference

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

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

Inheritance diagram for IDOMFontSourceObfuscationConverter:

```
 IRObject

 IDOMHashable  IEDLObject

 IDOMFontSource

 IDOMFontSourceStreamFilter

 IDOMFontSourceObfuscationConverter
```

Classes

- **class** Data
  
  *Initialization data.*

Public Types

- **enum** eOperation { eObfuscate, eDeobfuscate }
  
  *type used to uniquely identify the conversion direction*
Public Member Functions

- virtual bool getStream (IInputStreamPtr &stream) 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 bool determineUri (EDLSysString &uri) const
  
  Determines the URI based on the font source (backwards compatible version)

- virtual bool getInputFontSource (IDOMFontSourcePtr &fontSource) const =0
  
  Get the underlying font source of this font stream processor.

Static Public Member Functions

- static const CClassID & classID ()
  
  Retrieves the class id of IDOMFontSourceObfuscationConverter.

Additional Inherited Members

8.160.1 Detailed Description

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

8.160.2 Member Enumeration Documentation

8.160.2.1 eOperation

enum IDOMFontSourceObfuscationConverter::eOperation

  type used to uniquely identify the conversion direction

  Enumerator

<table>
<thead>
<tr>
<th>eObfuscate</th>
<th>Obfuscation.</th>
</tr>
</thead>
<tbody>
<tr>
<td>eDeobfuscate</td>
<td>Deobfuscation.</td>
</tr>
</tbody>
</table>

8.160.3 Member Function Documentation

8.160.3.1 classID()

static const CClassID & IDOMFontSourceObfuscationConverter::classID () [inline], [static]

  Retrieves the class id of IDOMFontSourceObfuscationConverter.
Returns

CClassId The class id of the element

8.160.3.2 determineUri() [1/2]

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, which will have a size of zero on failure.

Implements IDOMFontSourceStreamFilter.

8.160.3.3 determineUri() [2/2]

virtual bool IDOMFontSourceObfuscationConverter::determineUri ( EDLSysString & uri ) const [inline], [virtual]

Determines the URI based on the font source (backwards compatible version)

Note: This is an alternate version of determineUri() given above. This version has been kept in source for backward compatibility.

Parameters

uri Reference parameter to receive the determined URI.

Returns

bool True on success, or false if the call fails.

Reimplemented from IDOMFontSourceStreamFilter.

8.160.3.4 getInputFontSource()

virtual bool IDOMFontSourceObfuscationConverter::getInputFontSource ( IDOMFontSourcePtr & fontSource ) const [pure virtual]

Get the underlying font source of this font stream processor.
Parameters

| fontSource | A reference parameter to receive a smart pointer to the font source. |

Returns

bool True on success, or false if the call fails.

Implements IDOMFontSourceStreamFilter.

8.160.3.5 getStream()

virtual bool IDOMFontSourceObfuscationConverter::getStream ( 
    IInputStreamPtr & stream ) const [pure virtual]

Get the output stream of the processor.

Parameters

| stream | A reference parameter to receive a smart pointer to the output stream. |

Returns

bool True on success, or false if the call fails.

Implements IDOMFontSourceStreamFilter.

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

- idomfont.h

8.161 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 Member Functions

• virtual eFontStreamFilterType getFontStreamFilterType () const =0

   Get the font stream filter type.

• virtual bool getStream (IInputStreamPtr &stream) 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 bool determineUri (EDLSysString &uri) const

   Determines the URI based on the font source (backwards compatible version)

• virtual bool getInputFontSource (IDOMFontSourcePtr &fontSource) const =0

   Get the input font source of this font stream processor.
Additional Inherited Members

8.161.1 Detailed Description

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

8.161.2 Member Enumeration Documentation

8.161.2.1 eFontStreamFilterType

enum IDOMFontSourceStreamFilter::eFontStreamFilterType

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

Enumerator

<table>
<thead>
<tr>
<th>eFontStreamFilterTypeNone</th>
<th>Not specified.</th>
</tr>
</thead>
<tbody>
<tr>
<td>eFontStreamFilterTypeObfuscation</td>
<td>Obfuscation filter.</td>
</tr>
</tbody>
</table>

8.161.3 Member Function Documentation

8.161.3.1 determineUri() [1/2]

virtual const EDLSysString& IDOMFontSourceStreamFilter::determineUri ( ) const [pure virtual]

Determines the URI based on the font source (underlying font sources may be searched)

Returns

EDLSysString The uri, which will have a size of zero on failure.

Implements IDOMFontSource.

Implemented in IDOMFontSourceObfuscationConverter.

8.161.3.2 determineUri() [2/2]

virtual bool IDOMFontSourceStreamFilter::determineUri ( 
    EDLSysString & uri ) const [inline], [virtual]

Determines the URI based on the font source (backwards compatible version)

Note: This is an alternate version of determineUri() given above. This version has been kept in source for backward compatibility.
Parameters

**uri** Reference parameter to receive the determined URI.

Returns

**bool** True on success, or false if the call fails.

Reimplemented from **IDOMFontSource**.

Reimplemented in **IDOMFontSourceObfuscationConverter**.

### 8.161.3.3 getFontStreamFilterType()

```
virtual eFontStreamFilterType IDOMFontSourceStreamFilter::getFontStreamFilterType () const
[pure virtual]
```

Get the font stream filter type.

Returns

**eFontStreamFilterType** The font stream filter type

### 8.161.3.4 getInputFontSource()

```
virtual bool IDOMFontSourceStreamFilter::getInputFontSource ( 
    IDOMFontSourcePtr & fontSource ) const [pure virtual]
```

Get the input font source of this font stream processor.

Parameters

**fontSource** The font Source to retrieved

Returns

**bool** True on success

Implemented in **IDOMFontSourceObfuscationConverter**.
8.161.3.5 getStream()

virtual bool IDOMFontSourceStreamFilter::getStream (  
    IInputStreamPtr & stream ) const [pure virtual]

Retrieves the output stream of the processor.
Parameters

| stream | Reference parameter to receive a smart pointer to the output stream. |

Returns

**bool** True on success, false if the function fails.

Implemented in **IDOMFontSourceObfuscationConverter**.

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

- idomfont.h

### 8.162 IDOMForm Class Reference

**IDOMForm** interface. 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**.

```
#include <idomform.h>
```

Inheritance diagram for IDOMForm:
Public Member Functions

- virtual DOMid getFormId () const =0
  Obtain the unique DOMid of this form.
- virtual bool getMatrix (FMatrix &matrix) const =0
  Retrieves the Matrix of the form contents (equivalent to the /Matrix entry in a PDF/PS form dictionary).
- virtual bool setMatrix (const FMatrix &matrix)=0
  Sets the form matrix.
- virtual bool 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 &elementRef)=0
  Set the JawsMako logical structure element reference to be associated with this form, or NULL to clear.

Static Public Member Functions

- static EDL_API IDOMFormPtr create (IEDLClassFactory *pFactory, const FMatrix &matrix= FMatrix (), const FRect &bounds=FRect())
  Simplified creation function for IDOMForm. Throws an IEDLError exception on failure.
- static const CClassID & classID ()
  Retrieves class id of IDOMForm.

Additional Inherited Members

8.162.1 Detailed Description

IDOMForm interface. 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.

8.162.2 Member Function Documentation

8.162.2.1 classID()

static const CClassID & IDOMForm::classID () [inline], [static]
Retrieves class id of IDOMForm.

Returns

CClassID class id of the element

8.162.2.2 create()

static EDL_API IDOMFormPtr IDOMForm::create ( IEDLClassFactory *pFactory,
  const FMatrix & matrix = FMatrix(),
  const FRect & bounds = FRect() ) [static]
Simplified creation function for IDOMForm. Throws an IEDLError exception on failure.

Generated by Doxygen
Parameters

<table>
<thead>
<tr>
<th>pFactory</th>
<th>The EDL class factory</th>
</tr>
</thead>
<tbody>
<tr>
<td>matrix</td>
<td>The desired form matrix</td>
</tr>
<tr>
<td>bounds</td>
<td>The desired form bounds</td>
</tr>
</tbody>
</table>

Returns

IDOMFormPtr A smart pointer to the IDOMForm

8.162.2.3  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

8.162.2.4  getMatrix()

virtual bool IDOMForm::getMatrix ( 
       FMatrix & matrix ) const [pure virtual]

Retrieves the Matrix of the form contents (equivalent to the /Matrix entry in a PDF/PS form dictionary).

Parameters

| matrix | Reference to receive the matrix |

Returns

bool True on success
8.162.2.5  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

8.162.2.6  setBounds()

virtual bool IDOMForm::setBounds ( const FRect & bounds ) [pure virtual]

Sets the form bounding box.

Parameters

| bounds | Reference parameter to receive the bounding box. |

Returns

bool True on success, false if the call fails.

8.162.2.7  setMatrix()

virtual bool IDOMForm::setMatrix ( const FMatrix & matrix ) [pure virtual]

Sets the Matrix of the form contents (equivalent to the /Matrix entry in a PDF/PS form dictionary).

Parameters

| matrix | Reference to receive the matrix |

Returns

bool True on success
8.162.8.2 setStructureElementReference()

virtual void IDOMForm::setStructureElementReference ( const JawsMako::IStructureElementReferencePtr & elementRef ) [pure virtual]

Set the JawsMako logical structure element reference to be associated with this form, or NULL to clear.

Parameters

| elementRef | The structure element reference to set |

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

- idomform.h

8.163 IDOMFormInstance Class Reference

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

#include <idomform.h>

Inheritance diagram for IDOMFormInstance:

```
IRCOObject
  `-- IEDLObject
      `-- IDOMNode
          `-- IDOMFormInstance
```

Classes

- class Data
  
  Initialization data.
Public Member Functions

- virtual float getOpacity () const =0
  Get the form alpha/opacity.
- virtual bool setOpacity (float opacity)=0
  Set the form alpha/opacity.
- virtual bool getRenderTransform (FMatrix &renderTransform) const =0
  Retrieves render transform applied to this instance of the form.
- virtual bool setRenderTransform (FMatrix &renderTransform)=0
  Retrieves render transform applied to this instance of the form.
- virtual bool getForm (IDOMFormPtr &form) const =0
  Retrieves the Form associated with this instance.
- virtual bool setForm (IDOMFormPtr &form)=0
  Sets the Form associated with this instance.
- virtual bool getBlendMode (eBlendMode &blendMode) const =0
  Get the blend mode to be used for compositing this form.
- virtual bool setBlendMode (eBlendMode blendMode)=0
  Set the blend mode to be used for compositing this form.
- virtual bool getOpacityMask (IDOMBrushPtr &ptrOpacityMask) const =0
  Retrieves smart pointer to opacity mask.
- virtual bool setOpacityMask (IDOMBrushPtr &ptrOpacityMask)=0
  Sets the opacity mask.

Static Public Member Functions

- static const CClassID & classID () [inline], [static]
  Retrieves class id of IDOMForm.

Additional Inherited Members

8.163.1 Detailed Description

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

8.163.2 Member Function Documentation

8.163.2.1 classID()

static const CClassID& IDOMFormInstance::classID () [inline], [static]
Retrieves class id of IDOMForm.
Returns
  CClassID class id of the element

8.163.2.2 getBlendMode()

virtual bool IDOMFormInstance::getBlendMode (eBlendMode & blendMode) const [pure virtual]
Get the blend mode to be used for compositing this form.
Parameters

| blendMode | Reference to receive the blend mode |

Returns

| bool | True on success |

8.163.2.3 getForm()

virtual bool IDOMFormInstance::getForm (IDOMFormPtr & form) const [pure virtual]

Retrieves the Form associated with this instance.

Parameters

| form | Reference to receive the form. |

Returns

| bool | True on success. False if there are no contents. |

8.163.2.4 getOpacity()

virtual float IDOMFormInstance::getOpacity () const [pure virtual]

Get the form alpha/opacity.

Returns

| float | The opacity. |

8.163.2.5 getOpacityMask()

virtual bool IDOMFormInstance::getOpacityMask (IDOMBrushPtr & ptrOpacityMask) const [pure virtual]

Retrieves smart pointer to opacity mask.
Parameters

| ptrOpacityMask | Smart pointer to opacity mask |

Returns

bool True on success

8.163.2.6 getRenderTransform()

virtual bool IDOMFormInstance::getRenderTransform ( 
    FMatrix & renderTransform ) const [pure virtual]

Retrieves render transform applied to this instance of the form.

Parameters

| renderTransform | Reference to receive the render transform |

Returns

bool True on success

8.163.2.7 setBlendMode()

virtual bool IDOMFormInstance::setBlendMode ( 
    eBlendMode blendMode ) [pure virtual]

Set the blend mode to be used for compositing this form.

Parameters

| blendMode | The desired blend mode |

Returns

bool True on success

8.163.2.8 setForm()

virtual bool IDOMFormInstance::setForm ( 
    IDOMFormPtr & form ) [pure virtual]
Sets the Form associated with this instance.
Parameters

| form | Reference to the desired form |

Returns

**bool** True on success.

---

### 8.163.2.9 setOpacity()

virtual bool IDOMFormInstance::setOpacity ( 
    float opacity ) [pure virtual]

Set the form alpha/opacity.

Parameters

| opacity | The desired opacity, a value between 0.0 and 1.0 |

Returns

**bool** True on success

---

### 8.163.2.10 setOpacityMask()

virtual bool IDOMFormInstance::setOpacityMask ( 
    IDOMBrushPtr & ptrOpacityMask ) [pure virtual]

Sets the opacity mask.

Parameters

| ptrOpacityMask | Smart pointer to desired opacity mask |

Returns

**bool** True on success

---

### 8.163.2.11 setRenderTransform()

virtual bool IDOMFormInstance::setRenderTransform ( 
    FMatrix & renderTransform ) [pure virtual]
Retrieves render transform applied to this instance of the form.
Parameters

renderTransform  Reference to the desired transform

Returns

bool  True on success

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

• idomform.h

8.164  IDOMFunction Class Reference

Base class for PDF/PS Style functions.

#include <idomfunction.h>

Inheritance diagram for IDOMFunction:

Public Types

• enum eFunctionType

Public Member Functions

• virtual eFunctionType getFunctionType ()=0
  Retrieves function type.
• virtual int getNumInputValues ()=0
  Get the number of input values that this function will operate on.
• virtual int getNumOutputValues ()=0
  Get the number of output values that this function will produce.
• virtual bool getInputDomain (int inputNum, float &low, float &high)=0
  Get the input domain for a given input to the function.
• virtual bool getOutputRange (int outputNum, float &low, float &high)=0
  Get the output range for a given input to the function.
• virtual bool evaluate (float *inputValues, float *outputValues)=0
  Evaluate the input through the function and return the result.
• virtual bool evaluate (int *inputValues, float *outputValues)=0
  Evaluate the input through the function and return the result.
Additional Inherited Members

8.164.1 Detailed Description

Base class for PDF/PS Style functions.

8.164.2 Member Enumeration Documentation

8.164.2.1 eFunctionType

def IDOMFunction::eFunctionType

An enum for the various types of functions.

8.164.3 Member Function Documentation

8.164.3.1 evaluate() [1/2]

virtual bool IDOMFunction::evaluate(
    float * inputValues,
    float * outputValues) [pure virtual]

Evaluate the input through the function and return the result.

Parameters

| inputValues | An array of floats that are input into the function. The size of the array must be the same as the required number of inputValues |
| outputValues | An array of floats that are the result of evaluating the input through the function. The size of the array must be the same as the required number of inputValues |

Returns

bool Returns true on success. False if there is an error evaluating the input.

Implemented in IDOMPostScriptCalculatorFunction, IDOMGroupingFunction, IDOMStitchingFunction, IDOMExponentialFunction, and IDOMSampledFunction.
8.164.3.2 evaluate() [2/2]

```cpp
virtual bool IDOMFunction::evaluate ( 
    int * inputValues, 
    float * outputValues ) [pure virtual]
```

Evaluate the input through the function and return the result.

### Parameters

| inputValues | An array of integers that are input into the function. The size of the array must be the same as the required number of inputValues |
| outputValues | An array of floats that are the result of evaluating the input through the function. The size of the array must be the same as the required number of inputValues |

### Returns

bool Returns true on success. False if there is an error evaluating the input.

Implemented in IDOMPostScriptCalculatorFunction, IDOMGroupingFunction, IDOMStitchingFunction, IDOMExponentialFunction, and IDOMSampledFunction.

8.164.3.3 getFunctionType()

```cpp
virtual eFunctionType IDOMFunction::getFunctionType ( ) [pure virtual]
```

Retrieves function type.

### Returns

eFunctionType The function type

8.164.3.4 getInputDomain()

```cpp
virtual bool IDOMFunction::getInputDomain ( 
    int inputNum, 
    float & low, 
    float & high ) [pure virtual]
```

Get the input domain for a given input to the function.

### Parameters

| inputNum | The 0-indexed input number |
| low | A reference to receive the low domain bound for the given inputNum |
| high | A reference to receive the high domain bound for the given inputNum |
Returns

bool Returns true on success

8.164.3.5  getNumInputValues()

virtual int IDOMFunction::getNumInputValues ( ) [pure virtual]

Get the number of input values that this function will operate on.

Returns

int The number of input values.

8.164.3.6  getNumOutputValues()

virtual int IDOMFunction::getNumOutputValues ( ) [pure virtual]

Get the number of output values that this function will produce.

Returns

int The number of output values.

8.164.3.7  getOutputRange()

virtual bool IDOMFunction::getOutputRange (  
    int outputNum,  
    float & low,  
    float & high ) [pure virtual]

Get the output range for a given input to the function.

Parameters

<table>
<thead>
<tr>
<th>outputNum</th>
<th>The 0-indexed output number</th>
</tr>
</thead>
<tbody>
<tr>
<td>low</td>
<td>A reference to receive the low range bound for the given outputNum</td>
</tr>
<tr>
<td>high</td>
<td>A reference to receive the high range bound for the given outputNum</td>
</tr>
</tbody>
</table>

Returns

bool Returns true on success. False if there is no output range (which is not required for all function types) or other error.
The documentation for this class was generated from the following file:

- idomfunction.h

8.165  IDOMGlyph Class Reference

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

```cpp
#include <idomglyph.h>
```

Inheritance diagram for IDOMGlyph:

```
IRXObject
  ↓
IEDLObject
  ↓
IDOMNode
  ↓
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.
- 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.

Static Public Member Functions

- static const CClassID & classID ()
  
  Retrieves the class id for IDOMGlyph.

Additional Inherited Members

8.165.1 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

8.165.2 Member Typedef Documentation

8.165.2.1 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.

8.165.3 Member Enumeration Documentation

8.165.3.1 eGlyphIDSpecial

enum IDOMGlyph::eGlyphIDSpecial

Enumeration type used to define known, special Glyph IDs.

Enumerator

| eGlyphIDNotdef | The glyph ID to substitute for font undefined glyphs. |

8.165.4 Member Function Documentation
8.165.4.1 classID()

```cpp
static const CClassID & IDOMGlyph::classID ( ) [inline], [static]
```

Retrieves the class id for IDOMGlyph.

Returns

**CClassID** The class id of IDOMGlyph.

8.165.4.2 getAdvanceX()

```cpp
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

8.165.4.3 getAdvanceY()

```cpp
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.
8.165.4.4 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

8.165.4.5 getColor()

virtual bool IDOMGlyph::getColor () 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 setchar-width in the PS proc.

Returns

   bool Colored flag

8.165.4.6 getGlyphID()

virtual GlyphID IDOMGlyph::getGlyphID () const [pure virtual]

Retrieves the glyph id for this glyph.

Returns

   GlyphID GlyphID value

8.165.4.7 getGlyphName() [1/2]

virtual bool IDOMGlyph::getGlyphName ( EDLSysString & glyphName ) [pure virtual]

Retrieves the glyph name for this glyph, if present. An empty string is still a valid glyph name.
Parameters

| glyphName | Reference to receive the glyph name |

Returns

bool True on success, false if the call fails in any way.

8.165.4.8 getGlyphName()

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

8.165.4.9 getGlyphUnicode()

virtual bool IDOMGlyph::getGlyphUnicode ( 
    uint32 & codePoint ) const [pure virtual]

Retrieves the Unicode code point for the glyph.

Parameters

| codePoint | A reference to receive the code point |

Returns

bool True on success.

8.165.4.10 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.

8.165.4.11  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.

8.165.4.12  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.
8.165.4.13  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.

8.165.4.14  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 setchar-width in the PS proc

Parameters

| bColored | Set to true if the glyph specifies its own colors |

8.165.4.15  setHasCustomAdvance()

virtual void IDOMGlyph::setHasCustomAdvance ( bool bHasCustomAdvance ) [pure virtual]
Informs whether a custom value has been set for the glyph's advance.

The advance of a glyph is the distance from the glyph's origin to the origin of the next glyph along the baseline, which can be either vertical or horizontal.

A "custom" advance is one that has not been extracted from the font. A glyph has a custom advance if the advance information has been determined by the IndicesString or has been adjusted manually in some way.

Parameters

| bHasCustomAdvance | Set to true if the glyph has a custom advance |

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

- idomglyph.h

### 8.166 IDOMGlyphIDEEnumerator Class Reference

DOM GlyphID Enumerator.

#include <idomglyph.h>

Inheritance diagram for IDOMGlyphIDEEnumerator:

```
IRCOObject

IEDLObject

IDOMGlyphIDEEnumerator
```

Public Member Functions

- virtual void \texttt{add (IDOMGlyph::GlyphID\ glyphID)}=0
  
  \textit{Adds a glyph to the end of the list.}

- virtual IDOMGlyph::GlyphID \texttt{getGlyphID} ()=0
  
  \textit{Returns the ID of the glyph at the current point in the list.}

- virtual bool \texttt{beginEnumeration} ()=0
  
  \textit{Moves the current list point to the start of the list.}

- virtual bool \texttt{endEnumeration} ()=0
  
  \textit{Indicates whether or not the end of the list has been reached.}

- virtual bool \texttt{nextEnumerationItem} ()=0
  
  \textit{Moves the current list position on to the next item in the list, unless the end of the list has been reached.}
Static Public Member Functions

- static const CClassID & classID ()
  Retrieves the class id of IDOMGlyphIDEnumerator.

Additional Inherited Members

8.166.1 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.

8.166.2 Member Function Documentation

8.166.2.1 add()

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

Adds a glyph to the end of the list.

Parameters

<table>
<thead>
<tr>
<th>glyphID</th>
<th>The glyph ID to add to the end of the list.</th>
</tr>
</thead>
</table>

8.166.2.2 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.
8.166.2.3  classID()

static const CClassID& IDOMGlyphIDEnumerator::classID ( ) [inline], [static]

Retrieves the class id of IDOMGlyphIDEnumerator.

Returns

  CClassID  Class id of IDOMGlyphIDEnumerator.

8.166.2.4  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.

8.166.2.5  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.

8.166.2.6  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
8.167  IDOMGlyphName Class Reference

#include <idomglyph.h>

Inheritance diagram for IDOMGlyphName:

![Inheritance Diagram]

Additional Inherited Members

8.167.1  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

8.168  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:

```
IRCOBject

IEDLObject

IDOMNode

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 bool **setBidiLevel** (uint8 bidiLevel)=0
  
  *Sets the bidiLevel for this run of text.*

- virtual bool **getCaretStops** (EDLSysString &careStops) const =0
  
  *Retrieves the caret stop specification for a run of glyphs.*

- virtual bool **setCaretStops** (const EDLSysString &careStops)=0
  
  *Sets the caret stop specification for a run of glyphs.*

- virtual bool **getDeviceFontName** (EDLString &name) const =0
  
  *Retrieves the 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 bool **setDeviceFontName** (const EDLString &name)=0
Sets the 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 bool 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 bool setFont (IDOMFontPtr &font)=0

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

- virtual uint32 getFontIndex () const =0

  Retrieves the font index.

- virtual bool 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 bool 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 bool 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 bool setIsSideways (bool isSideways)=0

  Sets the IsSideways value. See getIsSideways for a description of the effects of IsSideways.

- virtual bool getIndices (EDLSysString &indices) const =0

  Retrieves the indices for a rendering a run of glyphs. The indices may be used specify a series of glyphs, complex character to glyph mappings, or a combination of both.

- virtual EDLSysString getIndices () const =0

  Retrieves a Unicode string that represents the text of the run of glyphs. Throws an IEDLError exception on failure. Here an empty indices is not considered a failure.

- virtual bool setIndices (const EDLSysString &indices)=0

  Sets the glyphs indices.

- virtual bool getUnicodeString (EDLString &unicodeString) const =0

  Retrieves a Unicode string that represents the text of the run of glyphs.

- virtual EDLString getUnicodeString () const =0

  Retrieves a Unicode string that represents the text of the run of glyphs. Throws an IEDLError exception on failure. Here an empty string is not considered a failure.

- virtual bool setUnicodeString (const EDLString &unicodeString)=0

  Sets the text value of the run of glyphs. The text is specified as Unicode code points.

- virtual eStyleSimulations getStyleSimulations () const =0

  Retrieves the style simulation value.

- virtual bool setStyleSimulations (eStyleSimulations styleSimulations)=0
Retrieves the style simulation value.

- virtual void getRenderTransform (FMatrix &matrix) const =0
  Retrieves the render transform matrix of the Glyphs node and its children.

- virtual void setRenderTransform (const FMatrix &matrix)=0
  Sets the render transform matrix of the Glyphs node and its children.

- 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 bool 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.

- virtual eBlendMode getBlendMode () const =0
  Gets the blend mode to be used for rendering this node.

- virtual bool 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 bool getNavigateLink (IDOMTargetPtr &target) const =0
  Retrieves a hyperlink URI associated with the node. The URI may be a relative reference or a URI that addresses a resource that is internal to or external to the package.

- virtual bool setNavigateLink (const IDOMTargetPtr &target)=0
  Associates a hyperlink URI with the node. The URI may be a relative reference or URI that addresses a resource that is internal to or external to the package.

- virtual bool getLanguage (EDLString &lang) const =0
  Retrieves the default language of the Glyphs element and any of its children.

- virtual bool setLanguage (const EDLString &lang)=0
  Sets the default language of the Glyphs element and any of its children.

- virtual bool getOpacityMask (IDOMBrushPtr &ptrOpacityMask) 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 bool 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 bool getClip (IDOMPathGeometryPtr &ptrClip) const =0
  Retrieves the clip for the Glyphs node. The clip limits the rendered region of the glyph.

- virtual bool setClip (const IDOMPathGeometryPtr &ptrClip)=0
  Sets the clip for the Glyphs node. The clip limits the rendered region of the glyph.

- virtual bool getFill (IDOMBrushPtr &ptrFill) const =0
  Retrieves the fill brush used to fill the shape of the rendered glyphs.

- virtual bool setFill (const IDOMBrushPtr &ptrFill)=0
  Sets the fill brush used to fill the shape of the rendered glyphs.

- virtual IDOMGlyphCollectionEnumPtr getGlyphInfoCollectionEnum ()=0
  Retrieves the enumerator of the glyph information collection.

- virtual void getFlattenedGlyphInfo (IDOMGlyph::CDOMGlyphDataVect &flattenedGlyphs) const =0
  Retrieves the flattened glyph information, similarly to getGlyphInfoCollectionEnum(). However, the results are provided as a simple vector of IDOMGlyph::Data objects, which is faster. Each glyph is laid out according to the glyphs node. Throws IEDLError on failure.

- virtual bool getIsCharPath ()=0
  Retrieves a Boolean value which indicates if this Glyphs node is intended to draw a path.

- virtual bool setIsCharPath (bool isCharPath)=0
  Sets the flag to indicate if this glyph is intended to draw a path.

- virtual bool getEquivalentPath (IDOMPathNodePtr &path)=0
  Retrieves the equivalent path for the Glyphs node.
Get a graphically equivalent path node for the glyphs node, with all glyphs converted to path geometry.

- virtual IDOMNodePtr split () const =0
  Split the glyphs node into a series of glyph nodes each consisting of the smallest possible unit.

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, e←StyleSimulations styleSimulations=eStyleSimulationsNone, 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.

Additional Inherited Members

8.168.1 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 and child nodes.

8.168.2 Member Function Documentation

8.168.2.1 classID()

static const CClassID & IDOMGlyphs::classID ( ) [inline], [static]

Retrieves class id of IDOM.

Returns

  CClassID Class id of the element
8.168.2.2 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 = eStyleSimulationsNone,  
    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

<table>
<thead>
<tr>
<th>pFactory</th>
<th>The factory to use.</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>The desired position for the glyphs.</td>
</tr>
<tr>
<td>unicodeString</td>
<td>The text to be used.</td>
</tr>
<tr>
<td>size</td>
<td>The font size to use, in ems.</td>
</tr>
<tr>
<td>brush</td>
<td>The fill brush to use.</td>
</tr>
<tr>
<td>font</td>
<td>The font to use.</td>
</tr>
<tr>
<td>fontIndex</td>
<td>The index of the font to use, if the font is within a truetype collection.</td>
</tr>
<tr>
<td>styleSimulations</td>
<td>The desired style simulations effect.</td>
</tr>
<tr>
<td>renderTransform</td>
<td>The render transform to use.</td>
</tr>
<tr>
<td>clip</td>
<td>The geometry to use for clipping. NULL if no clip.</td>
</tr>
<tr>
<td>opacity</td>
<td>The opacity to use.</td>
</tr>
</tbody>
</table>

Returns

IDOMPathNodePtr The new path.

8.168.2.3 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.
8.168.2.4  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.

8.168.2.5  getCaretStops()

virtual bool IDOMGlyphs::getCaretStops ( 
  EDLSysString & caretStops ) const [pure virtual]

Retrieves the caret stop specification for a run of glyphs.

Caret stops identify the positions within the sequence of Unicode characters at which a text-selection tool may place a text-editing caret. Potential caret-stop positions are identified by their indices into the UTF-16 code units represented by the unicodeString attribute value. When this value is missing, the test in the UnicodeString attribute must be interpreted as having a caret stop between every Unicode UTF-16 code unit and at the beginning and end of the text.

Because it is legal to place the caret before any of the UTF-16 code units in the Unicode string if the caretStops attribute is omitted, omitting the caretStops attribute is equivalent to specifying a string that has all the bits set to 1.

If there are insufficient flags in the caretStops string to correspond to all the UTF-16 code units in the Unicode string, all remaining UTF-16 code units in the Unicode string MUST be considered valid caret stops.

Parameters

| caretStops | A reference parameter to recieve the caret stops specification string. See setCaretStops() for a description of the format of the specification string. |

Returns
   bool  True on success

8.168.2.6  getClip()

virtual bool IDOMGlyphs::getClip ( 
  IDOMPathGeometryPtr & ptrClip ) const [pure virtual]

Retrieves the clip for the Glyphs node. The clip limits the rendered region of the glyph.
Parameters

| ptrClip | Smart pointer to receive the clip information. |

Returns

`bool` True on success, false if the call fails.

8.168.2.7 `getDeviceFontName()`

```cpp
virtual bool IDOMGlyphs::getDeviceFontName (  
    EDLString & name ) const [pure virtual]
```

Retrieves the device font name. The device font name, which uniquely identifies a specific device font. The identifier is typically defined by a hardware vendor or font vendor.

Parameters

| name | The device font name |

Returns

`bool` True on success, false if the call fails.

8.168.2.8 `getEquivalentPath()`

```cpp
virtual bool IDOMGlyphs::getEquivalentPath (  
    IDOMPathNodePtr & path ) [pure virtual]
```

Get a graphically equivalent path node for the glyphs node, with all glyphs converted to path geometry.

Parameters

| path | A reference to receive the path. |

Returns

`bool` True on success, false if the call fails.

8.168.2.9 `getFill()`

```cpp
virtual bool IDOMGlyphs::getFill (  
    IDOMBrushPtr & ptrFill ) const [pure virtual]
```
Retrieves the fill brush used to fill the shape of the rendered glyphs.

### Parameters

- `ptrFill` Smart pointer to receive a reference to the fill brush.

### Returns

- **bool** True on success, false if the call fails.

8.168.2.10  **getFlattenedGlyphInfo()**

```cpp
virtual void IDOMGlyphs::getFlattenedGlyphInfo ( 
    IDOMGlyph::CDOMGlyphDataVect & flattenedGlyphs ) const [pure virtual]
```

Retrieves the flattened glyph information, similarly to `getGlyphInfoCollectionEnum()`. However, the results are provided as a simple vector of `IDOMGlyph::Data` objects, which is faster. Each glyph is laid out according to the glyphs node. Throws `IEDLError` on failure.

### Parameters

- `flattenedGlyphs` A reference to receive the data

8.168.2.11  **getFont()**

```cpp
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.

8.168.2.12  **getFontIndex()**

```cpp
virtual uint32 IDOMGlyphs::getFontIndex ( ) const [pure virtual]
```

Retrieves the font index.

### Returns

- **uint32** The font index.
8.168.2.13  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.

8.168.2.14  getGlyphInfoCollectionEnum()

virtual IDOMGlyphCollectionEnumPtr IDOMGlyphs::getGlyphInfoCollectionEnum ( )  [pure virtual]

Retrieves the enumerator of the glyph information collection.

Returns

  IDOMGradientStopCollectionEnumPtr  The glyph information collection enumerator

8.168.2.15  getIndices()  [1/2]

virtual bool IDOMGlyphs::getIndices ( 
  EDLSysString &  indices  ) const  [pure virtual]

Retrieves the indices for a rendering a run of glyphs. The indices may be used specify a series of glyphs, complex character to glyph mappings, or a combination of both.

Parameters

  indices  A string specifying the glyph indices—that is, metrics and clustering information.

Returns

  bool  True on success, false if the call fails.

8.168.2.16  getIndices()  [2/2]

virtual EDLSysString IDOMGlyphs::getIndices ( ) const  [pure virtual]

Retrieves a Unicode string that represents the text of the run of glyphs. Throws an IEDLError exception on failure. Here an empty indices is not considered a failure.
Returns

**EDLSysString** The indices.

8.168.2.17  getIsCharPath()

virtual bool IDOMGlyphs::getIsCharPath ( ) [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.

8.168.2.18  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

8.168.2.19  getLanguage()

virtual bool IDOMGlyphs::getLanguage ( 
    EDLSysString & lang ) 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/](http://www.w3.org/International/articles/language-tags/)
Parameters

| lang | Smart pointer to receive the default language string. |

Returns

**bool** True on success, false if the call fails.

8.168.2.20 getNavigateLink()

```cpp
virtual bool IDOMGlyphs::getNavigateLink ( 
    IDOMTargetPtr & target ) const [pure virtual]
```

Retrieves a hyperlink URI associated with the node. The URI may be a relative reference or a URI that addresses a resource that is internal to or external to the package.

Parameters

| target | A smart pointer to receive the URI. |

Returns

**bool** True on success, false if the call fails.

8.168.2.21 getOpacity()

```cpp
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

**double** Returns the opacity value.

8.168.2.22 getOpacityMask()

```cpp
virtual bool IDOMGlyphs::getOpacityMask ( 
    IDOMBrushPtr & ptrOpacityMask ) 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.
Parameters

- **ptrOpacityMask**: Smart pointer to receive the opacity mask.

Returns

- **bool**: True on success, false if the call fails.

---

8.168.2.23  getOriginX()

```cpp
template 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.

---

8.168.2.24  getOriginY()

```cpp
template 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.

---

8.168.2.25  getRenderTransform()

```cpp
template virtual void IDOMGlyphs::getRenderTransform(
    FMatrix & matrix ) 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.
8.168 IDOMGlyphs Class Reference

Parameters

| matrix | A smart pointer to receive the render transform matrix. |

8.168.2.26 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 eStyleSimulationsNone, that is, the shapes of the glyphs are not modified. When eStyleSimulationsBold 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 eStyleSimulationsItalic 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 eStyleSimulationsBoldItalic.

Returns

eStyleSimulations The current style simulation setting.

8.168.2.27 getUnicodeString() [1/2]

virtual bool IDOMGlyphs::getUnicodeString ( EDLString & unicodeString ) const [pure virtual]

Retrieves a Unicode string that represents the text of the run of glyphs.

Parameters

| unicodeString | A reference to receive the string. |

Returns

bool True on success, false if the call fails.
8.168.2.28  getUnicodeString() [2/2]

virtual EDLString IDOMGlyphs::getUnicodeString ( ) const [pure virtual]

Retrieves a Unicode string that represents the text of the run of glyphs. Throws an IEDLError exception on failure. Here an empty string is not considered a failure.

Returns

EDLString The unicode string.

8.168.2.29  setBidiLevel()

virtual bool IDOMGlyphs::setBidiLevel ( 
    uint8 bidiLevel ) [pure virtual]

Sets the bidiLevel for this run of text.

The bidi (bi-directional) level specifies the Unicode bi-directional nesting level. A value of 0 implies left-to-right text, while a value of 1 implies right-to-left text. Right-to-left text places the run origin at the right side of the first glyph, with positive advance widths representing advances to the left, so that subsequent glyphs are placed to the left of the previous glyph.

Parameters

| bidiLevel | The new bidiLevel value. |

Returns

bool True on success, false if the call fails.

8.168.2.30  setBlendMode()

virtual bool IDOMGlyphs::setBlendMode ( 
    eBlendMode blendMode ) [pure virtual]

Sets the blend mode to be used for rendering this node. Note: modes other than Normal are not directly representable in XPS.

Parameters

| blendMode | The new blend mode. |
8.168.2.31  setCaretStops()

virtual bool IDOMGlyphs::setCaretStops (  
   const EDLSysString & caretStops ) [pure virtual]

Sets the caret stop specification for a run of glyphs.

Caret stops identify the positions within the sequence of Unicode characters at which a text-selection tool may place a text-editing caret. Potential caret-stop positions are identified by their indices into the UTF-16 code units represented by the unicodeString attribute value. When this value is missing, the test in the UnicodeString attribute must be interpreted as having a caret stop between every Unicode UTF-16 code unit and at the beginning and end of the text.

Because it is legal to place the caret before any of the UTF-16 code units in the Unicode string if the caretStops attribute is omitted, omitting the caretStops attribute is equivalent to specifying a string that has all the bits set to 1.

If there are insufficient flags in the caretStops string to correspond to all the UTF-16 code units in the Unicode string, all remaining UTF-16 code units in the Unicode string MUST be considered valid caret stops.

Parameters

| caretStops | The caret stops specification expressed as a string. |

The caretStops attribute contains an array of boolean bit-flags, which is represented as a string of hexadecimal characters. The caretStops attribute includes a final flag for placement of the caret following the final UTF-16 code unit in the Unicode string. Each hexadecimal character in the caretStops value represents the flags for four UTF-16 code units in the Unicode string, with the highest-order bit representing the first UTF-16 code unit. Any unused bits in the last UTF-16 code unit must be 0. For example if caretStops is set to "f7" (that is, 1111 0111), in reference to the run of glyphs, "Not here.", it means that it is valid to place the caret stop in any position in the run of text except for directly before the "h" of "here".

Returns

   bool True on success, false if the call fails.

8.168.2.32  setClip()

virtual bool IDOMGlyphs::setClip (  
   const IDOMPathGeometryPtr & ptrClip ) [pure virtual]

Sets the clip for the Glyphs node. The clip limits the rendered region of the glyph.
### Parameters

| `ptrClip` | Smart pointer to the new clip information. |

### Returns

- **bool** True on success, false if the call fails.

#### 8.168.2.33 `setDeviceFontName()`

```cpp
template virtual bool IDOMGlyphs::setDeviceFontName (const EDLString & name) [pure virtual]
```

Sets the device font name. The device font name, which uniquely identifies a specific device font. The identifier is typically defined by a hardware vendor or font vendor.

**Parameters**

| `name` | The new device font name. |

**Returns**

- **bool** True on success, false if the call fails.

#### 8.168.2.34 `setFill()`

```cpp
template virtual bool IDOMGlyphs::setFill (const IDOMBrushPtr & ptrFill) [pure virtual]
```

Sets the fill brush used to fill the shape of the rendered glyphs.

**Parameters**

| `ptrFill` | Smart pointer to the new fill brush. |

**Returns**

- **bool** True on success, false if the call fails.

#### 8.168.2.35 `setFont()`

```cpp
template virtual bool IDOMGlyphs::setFont (IDOMFontPtr & font) [pure virtual]
```

Generated by Doxygen
Sets the font object to be used to display the text.

**Parameters**

| font | Smart pointer to the new font object to use. |

**Returns**

**bool** True on success, false if the call fails.

---

### 8.168.2.36 setFontIndex()

```cpp
virtual bool IDOMGlyphs::setFontIndex ( 
        uint32 index ) [pure virtual]
```

Sets the font index.

**Parameters**

| index | The new font index value. |

**Returns**

**bool** True on success, false if the call fails.

---

### 8.168.2.37 setFontRenderingEmSize()

```cpp
virtual bool IDOMGlyphs::setFontRenderingEmSize ( 
        double emSize ) [pure virtual]
```

Sets the font rendering em size in drawing surface units. The font rendering em size is expressed as a floating point value in units of the effective coordinate space. A value of 0 results in no visible text.

**Parameters**

| emSize | The new font rendering em size value. |

**Returns**

**bool** True on success, false if the call fails.
8.168.2.38  setIndices()

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

Sets the glyphs indices.

Parameters

| indices | The string specifying the indices. Within the string, each glyph specification is separated by a semicolon. |

Returns

bool True on success, false if the call fails.

8.168.2.39  setIsCharPath()

virtual bool IDOMGlyphs::setIsCharPath ( bool isCharPath ) [pure virtual]

Sets the flag to indicate if this glyph is intended to draw a path.

Parameters

| isCharPath | True if this node is a charpath, false otherwise. |

Returns

bool True on success, false if the call fails.

8.168.2.40  setIsSideways()

virtual bool IDOMGlyphs::setIsSideways ( bool isSideways ) [pure virtual]

Sets the IsSideways value. See getIsSideways for a description of the effects of IsSideways.

Parameters

| isSideways | The new IsSideways value. |

Returns

bool True on success, false if the call fails.
8.168.2.41  setLanguage()

virtual bool 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/

Parameters

| lang | The new default language string. |

Returns

    bool True on success, false if the call fails.

8.168.2.42  setNavigateLink()

virtual bool IDOMGlyphs::setNavigateLink (  
    const IDOMTargetPtr &  target  )  [pure virtual]

Associates a hyperlink URI with the node. The URI may be a relative reference or URI that addresses a resource that is internal to or external to the package.

Parameters

| target | The URI to associate with the node. |

Returns

    bool True on success, false if the call fails.

8.168.2.43  setOpacity()

virtual bool 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.
Parameters

opc  The new opacity value.

Returns

bool  True on success, false if the call fails.

8.168.2.44  setOpacityMask()

virtual bool IDOMGlyphs::setOpacityMask (const IDOMBrushPtr & ptrOpacityMask) [pure virtual]

Sets the opacity mask for the node. The opacity mask specifies a mask of alpha values that is applied to the glyphs in the same fashion as the opacity setting, but allowing different alpha values for different areas of the canvas.

Parameters

ptrOpacityMask  Smart pointer to a brush representing the new opacity mask for the node.

Returns

bool  True on success, false if the call fails.

8.168.2.45  setOriginX()

virtual bool IDOMGlyphs::setOriginX (double originX) [pure virtual]

Sets the x-coordinate of the first glyph in the run, in units of the effective coordinate space. The first glyph is placed so that the leading edge of its advance vector and its baseline intersect with the point defined by the originX and originY coordinates.

Parameters

originX  The new x-coordinate for the first glyph in the text run.

Returns

bool  True on success, false if the call fails.
8.168.2.46 setOriginY()

virtual bool IDOMGlyphs::setOriginY (double originY) [pure virtual]

Sets the y-coordinate of the first glyph in the run, in units of the effective coordinate space. The first glyph is placed so that the leading edge of its advance vector and its baseline intersect with the point defined by the originX and originY coordinates.

Parameters

originY The new y-coordinate for the first glyph in the text run.

Returns

bool True on success, false if the call fails.

8.168.2.47 setRenderTransform()

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

Sets the render transform matrix of the Glyphs node and its children.

The render transform establishes a new coordinate frame for the child and descendants of the Glyphs node. The render transform also affects clip and opacity masks, fill, x-origin, y-origin, the actual shape of the individual glyphs and the advance widths. The render transform also affects the font size and values specified by the indices setting.

Parameters

matrix the new render transform matrix.

8.168.2.48 setStyleSimulations()

virtual bool 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 eStyleSimulationsBold 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 eStyleSimulationItalic 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 eStyleSimulationsBoldItalic.

Parameters

| styleSimulations | The new style simulation setting. |

Returns

| bool | True on success, false if the call fails. |

8.168.2.49 setUnicodeString()

virtual bool IDOMGlyphs::setUnicodeString ( const EDLString & unicodeString ) [pure virtual]

Sets the text value of the run of glyphs. The text is specified as Unicode code points.

Parameters

| unicodeString | A string of UTF-8 characters representing the text to be rendered by this Glyphs node. |

Returns

| bool | True on success, false if the call fails. |

8.168.2.50 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.

An exception of type IEDLObject will be thrown on error.
Returns

**IDOMNodePtr** The result of splitting the glyphs node.

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

- idomglyphs.h

## 8.169 IDOMGradientBrush Class Reference

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

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

Inheritance diagram for IDOMGradientBrush:

![Inheritance diagram for IDOMGradientBrush](image-url)
Public Member Functions

- virtual eColorInterpolationMode getColorInterpolationMode () const =0
  Retrieves the color interpolation mode value of the radial gradient brush.
- virtual bool 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 bool setSpreadMethod (eSpreadMethod sm)=0
  Sets spread method value of the RadialGradientBrush element.
- virtual IDOMGradientStopCollectionEnumPtr getGradientStopCollectionEnum ()=0
  Retrieves an enumerator for the list of the gradient stops for the brush.
- virtual uint32 getGradientStopsCount ()=0
  Retrieves the number of gradient stops in the gradient stop list.
- virtual void clearGradientStopCollection ()=0
  Clears the gradient stop list.
- virtual bool addGradientStop (IDOMGradientStopPtr &ptrGradientStop)=0
  Append a gradient stop to the collection of gradient stops.

Additional Inherited Members

8.169.1 Detailed Description

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

8.169.2 Member Function Documentation

8.169.2.1 addGradientStop()

virtual bool IDOMGradientBrush::addGradientStop (IDOMGradientStopPtr &ptrGradientStop)  [pure virtual]

Append a gradient stop to the collection of gradient stops.

Parameters

| ptrGradientStop | The smart pointer to the gradient stop interface |

See also

IDOMGradientStop
8.169.2.2  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. Returns the color interpolation mode.

8.169.2.3  getGradientStopCollectionEnum()

virtual IDOMGradientStopCollectionEnumPtr IDOMGradientBrush::getGradientStopCollectionEnum ( ) [pure virtual]

Retrieves an enumerator for the list of the gradient stops for the brush.

See also

  IDOMGradientStop

Returns

  IDOMGradientStopCollectionEnumPtr. Returns an enumerator for the gradient stop list.

8.169.2.4  getGradientStopsCount()

virtual uint32 IDOMGradientBrush::getGradientStopsCount ( ) [pure virtual]

Retrieves the number of gradient stops in the gradient stop list.

Returns

  uint32. Returns the number of defined gradient stops.
8.169.2.5 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
eSpreadMethod. Returns the spread method value.

8.169.2.6 setColorInterpolationMode()

virtual bool IDOMGradientBrush::setColorInterpolationMode ( 
    eColorInterpolationMode cim ) [pure virtual]

Sets the color interpolation mode value of the radial gradient brush.

This is the gamma function for color interpolation for sRGB colors. The gamma adjustment should not be applied to the alpha component, if specified.

Parameters
cim The color interpolation mode value

See also
eColorInterpolationMode

Returns
bool. Returns true on success, false if the call fails.

8.169.2.7 setSpreadMethod()

virtual bool IDOMGradientBrush::setSpreadMethod ( 
    eSpreadMethod sm ) [pure virtual]

Sets spread method value of the RadialGradientBrush element.

The spread method describes how the brush should fill the content area outside of the primary gradient area.
Parameters

\( sm \)  The new spread method value.

Returns

bool. Returns true on success, false if the call fails.

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

- idombrush.h

### 8.170 IDOMGradientStop Class Reference

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

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

Inheritance diagram for IDOMGradientStop:

```
IRCOBJECT

IEDLOBJECT

IDOMGRADIENTSTOP
```

#### Classes

- class **Data**
  
  *Initialization data.*

#### Public Member Functions

- virtual bool **getColor** (IDOMColorPtr &color) const =0
  
  Retrieves the color value of the gradient stop.

- virtual bool **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 bool **setOffset** (double offset)=0
  
  Sets the offset value of the gradient stop.
Static Public Member Functions

- static const CClassID & classID()
  Retrieves class id of IDOMGradientStop.

Additional Inherited Members

8.170.1 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.

8.170.2 Member Function Documentation

8.170.2.1 classID()

static const CClassID & IDOMGradientStop::classID ( ) [inline], [static]

Retrieves class id of IDOMGradientStop.

Returns

CClassID. Class id of the element

8.170.2.2 getColor()

virtual bool IDOMGradientStop::getColor ( 
    IDOMColorPtr & color ) const [pure virtual]

Retrieves the color value of the gradient stop.

Parameters

color Reference parameter to receive the color value.
Returns

bool. Returns true on success, false if the call fails.

8.170.2.3 getOffset()

virtual double IDOMGradientStop::getOffset ( ) const [pure virtual]

Retrieves the offset value of the gradient stop.

Returns

double. Returns the offset value.

8.170.2.4 setColor()

virtual bool IDOMGradientStop::setColor ( const IDOMColorPtr & color ) [pure virtual]

Sets the color value of the gradient stop.

Parameters

| color | The new color value. |

Returns

bool. Returns true on success, false if the call fails.

8.170.2.5 setOffset()

virtual bool IDOMGradientStop::setOffset ( double offset ) [pure virtual]

Sets the offset value of the gradient stop.

Parameters

| offset | The new offset value. |
Returns

bool. Returns true on success, false if the call fails.

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

- idombrush.h

## 8.171 IDOMGroup Class Reference

IDOMGroup interface.

```cpp
#include <idomgroup.h>
```

Inheritance diagram for IDOMGroup:

![Inheritance diagram](image)

### Classes

- class **Data**

  * Initialization data.*

---

Generated by Doxygen
Public Member Functions

- virtual bool getRenderTransform (FMatrix &matrix) const =0
  Retrieves render transform matrix of the Group and its children.
- virtual bool setRenderTransform (const FMatrix &matrix)=0
  Sets render transform matrix of the Group and its children.
- virtual bool getClip (IDOMPathGeometryPtr &ptrClip) const =0
  Retrieves smart pointer to clip.
- virtual bool 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.

Static Public Member Functions

- static EDL_API IDOMGroupPtr create (IEDLClassFactory ∗pFactory, const FMatrix &transform=FMatrix(), const IDOMPathGeometryPtr &clip=IDOMPathGeometryPtr(NULL))
  Simplified creation function for IDOMGroup. Throws an IEDLError exception on failure.
- static const CClassID & classID ()
  Retrieves class id of IDOMGroup.

Additional Inherited Members

8.171.1 Detailed Description

IDOMGroup interface.

8.171.2 Member Function Documentation

8.171.2.1 classID()

static const CClassID & IDOMGroup::classID () [inline], [static]

Retrieves class id of IDOMGroup.

Returns

CClassID class id of the element

Generated by Doxygen
8.171.2.2 create()

```cpp
static EDL_API IDOMGroupPtr IDOMGroup::create (  
    IEDLClassFactory ∗ pFactory,  
    const FMatrix & transform = FMatrix (),  
    const IDOMPathGeometryPtr & clip = IDOMPathGeometryPtr(NULL) ) [static]
```

Simplified creation function for IDOMGroup. Throws an IEDLError exception on failure.

**Parameters**

<table>
<thead>
<tr>
<th>pFactory</th>
<th>The EDL class factory</th>
</tr>
</thead>
<tbody>
<tr>
<td>transform</td>
<td>The desired render transform</td>
</tr>
<tr>
<td>clip</td>
<td>The desired clip</td>
</tr>
</tbody>
</table>

**Returns**

IDOMGroupPtr The IDOMGroup

8.171.2.3 getClip()

```cpp
virtual bool IDOMGroup::getClip (  
    IDOMPathGeometryPtr & ptrClip ) const [pure virtual]
```

Retrieves smart pointer to clip.

**Parameters**

| ptrClip       | Smart pointer to clip |

**Returns**

bool True on success

8.171.2.4 getMarkedContentDetails()

```cpp
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
8.171.2.5 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

8.171.2.6 getRenderTransform()

virtual bool IDOMGroup::getRenderTransform(FMatrix & matrix) const [pure virtual]

Retrieves render transform matrix of the Group and its children.

Parameters

| matrix | Render transform matrix |

Returns

bool True on success

8.171.2.7 setClip()

virtual bool IDOMGroup::setClip(const IDOMPathGeometryPtr & ptrClip) [pure virtual]

Sets clip.

Parameters

| ptrClip | Smart pointer to clip |

Returns

bool True on success
8.171.2.8 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

<table>
<thead>
<tr>
<th>details</th>
<th>The Marked Content details</th>
</tr>
</thead>
</table>

8.171.2.9 setOptionalContentDetails()

virtual void IDOMGroup::setOptionalContentDetails (  
    const JawsMako::IOptionalContentDetailsPtr & details ) [pure virtual]

Set the JawsMako Optional Content details for the group, or NULL to remove. Note that this is only allowed for bare 
IDOMGroup objects and *not* for subclasses. Any attempt to set optional content details on any object that is not an 
IDOMGroup (getNodeType() is eDOMGroupNode) will result in an exception. Note that an IDOMGroup may have 
optional content details or marked content details, but not both.

Parameters

<table>
<thead>
<tr>
<th>details</th>
<th>The Optional Content details</th>
</tr>
</thead>
</table>

8.171.2.10 setRenderTransform()

virtual bool IDOMGroup::setRenderTransform (  
    const FMatrix & matrix ) [pure virtual]

Sets render transform matrix of the Group and its children.

Parameters

<table>
<thead>
<tr>
<th>matrix</th>
<th>Render transform matrix</th>
</tr>
</thead>
</table>

Returns

    bool True on success

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

- idomgroup.h
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 bool getNumFunctions (uint32 &numFunctions)=0
  
  *Get the number of functions in the group.*

- virtual bool getFunctionAtIndex (uint32 index, IDOMFunctionPtr &function)=0
  
  *Get the function for a given function index.*

- virtual bool evaluate (float *inputValues, float *outputValues)=0
  
  *Evaluate the input through the function and return the result.*

- virtual bool evaluate (int *inputValues, float *outputValues)=0
  
  *Evaluate the input through the function and return the result.*

Static Public Member Functions

- static const CClassID & classID ()
  
  *Retrieves class id of IDOM.*
Additional Inherited Members

8.172.1 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.

8.172.2 Member Function Documentation

8.172.2.1 classID()

static const CClassID & IDOMGroupingFunction::classID ( ) [inline], [static]

Retrieves class id of IDOM.

Returns

\[
\text{CClassID class id of the element}
\]

8.172.2.2 evaluate() [1/2]

virtual bool IDOMGroupingFunction::evaluate (  
  float * inputValues,  
  float * outputValues ) [pure virtual]

Evaluate the input through the function and return the result.

Parameters

| inputValues | A pointer to a single float value that is the input to this grouped function. |
| outputValues | An array of floats that are the result of evaluating the input through the function. The size of the array must be the same as the required number of output values |

Returns

\[
\text{bool Returns true on success. False if there is an error evaluating the input.}
\]

Implements IDOMFunction.
virtual bool IDOMGroupingFunction::evaluate (
    int * inputValues,
    float * outputValues ) [pure virtual]

Evaluate the input through the function and return the result.

Parameters

| inputValues | A pointer to a single float value that is the input to this grouped function. |
| outputValues | An array of floats that are the result of evaluating the input through the function. The size of the array must be the same as the required number of output values |

Returns

bool Returns true on success. False if there is an error evaluating the input.

Implements IDOMFunction.

8.172.2.4 getFunctionAtIndex()

virtual bool IDOMGroupingFunction::getFunctionAtIndex (
    uint32 index,
    IDOMFunctionPtr & function ) [pure virtual]

Get the function for a given function index.

Parameters

| index | A zero based index for the desired function. |
| function | A reference to receive the function pointer. |

Returns

bool Returns true on success

8.172.2.5 getNumFunctions()

virtual bool IDOMGroupingFunction::getNumFunctions (
    uint32 & numFunctions ) [pure virtual]

Get the number of functions in the group.
Parameters

| numFunctions | A reference to receive the number of functions. |

Returns

- bool Returns true on success.

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

- idomfunction.h

8.173 IDOMHashable Class Reference

Abstract interface for EDL objects that may be hashed.

#include <idomhashable.h>
Inheritance diagram for IDOMHashable:

Public Member Functions

- virtual bool hash (uint64 &hash)=0

Retrieve a hash for this object.
8.173.1 Detailed Description

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.

8.173.2 Member Function Documentation

8.173.2.1 hash()

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

Retrieve a hash for this object.

Parameters

| hash | A reference to receive the hash. |

Returns

**bool** True on success, false if the call fails.

Implemented in `IInputStream`.

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

- `idomhashable.h`

8.174 IDOMICCProfile Class Reference

`IDOMICCProfile` interface.

```cpp
#include <idomresources.h>
```
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.*

Static Public Member Functions

• static EDL_API IDOMICCProfilePtr create (IEDLClassFactory *pFactory, const IInputStreamPtr &stream, uint32 length=0, const EDLSysString &uri=EDLSysString())
  *Creation function for an IDOMICCProfile Throws an IEDLError exception on failure.*

• static const CClassID & classID ()
  *Retrieves class id of IDOM.*

Additional Inherited Members

8.174.1 Detailed Description

IDOMICCProfile interface.
8.174.2 Member Function Documentation

8.174.2.1 classID()

static const CClassID & IDOMICCProfile::classID ( ) [inline], [static]

Retrieves class id of IDOM.

Returns

  CClassID Class id of the element

8.174.2.2 create()

static EDL_API IDOMICCProfilePtr IDOMICCProfile::create (IEDLClassFactory * pFactory,
     const IInputStreamPtr & stream,
     uint32 length = 0,
     const EDLSysString & uri = EDLSysString() ) [static]

Creation function for an IDOMICCProfile Throws an IEDLError exception on failure.

Parameters

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

Returns

  IDOMICCProfilePtr The new profile.

8.174.2.3 getProfileVersion()

virtual void IDOMICCProfile::getProfileVersion ( uint8 & majorVersion,
     uint8 & minorVersion ) [pure virtual]

Get the ICC profile version. Throws an exception of type IEDLError on failure.
Parameters

- **majorVersion** Reference to receive the major version.
- **minorVersion** Reference to receive the minor version.

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

- `idomresources.h`

### 8.175 IDOMImage Class Reference

The base class describing an image. This class is subclassed to create a number of more specific image types.

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

Inheritance diagram for IDOMImage:

![Inheritance diagram for IDOMImage](image)

#### Classes

- **class Data**
  
  *Initialization data.*
Public Member Functions

- virtual bool createImageDecoder (IEDLClassFactory *factory, IImageDecoderPtr &imageDecoder, IDOMImagePropertiesPtr &imageProperties)=0
  *Creates a properly initialized image decoder object that reads from an inputstream that is specific to that image format.*

- virtual bool getFirstImageFrame (IEDLClassFactory *factory, IImageFramePtr &frame)
  *Convenience routine to retrieve the first frame in an image.*

- virtual bool createImageEncoder (ISessionPtr &session, IOutputStreamPtr &imageDest, IImageEncoderPtr &imageEncoder, IDOMImagePropertiesPtr &imageProperties)=0
  *Creates a properly initialized image encoder object that writes to an outputstream that is specific to that image format.*

- virtual bool getImageProperties (IDOMImagePropertiesPtr &imageProperties)=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.*

Additional Inherited Members

8.175.1 Detailed Description

The base class describing an image. This class is subclassed to create a number of more specific image types.

8.175.2 Member Function Documentation

8.175.2.1 createImageDecoder()

virtual bool IDOMImage::createImageDecoder (IEDLClassFactory *factory, IImageDecoderPtr &imageDecoder, IDOMImagePropertiesPtr &imageProperties) [pure virtual]

Creates a properly initialized image decoder object that reads from an inputstream that is specific to that image format.

Parameters

<table>
<thead>
<tr>
<th>parameter</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>factory</td>
<td>Pointer to the EDL class factory</td>
</tr>
<tr>
<td>imageDecoder</td>
<td>Smart pointer to the image decoder interface</td>
</tr>
<tr>
<td>imageProperties</td>
<td>Smart pointer to the DOM image properties interface</td>
</tr>
</tbody>
</table>

Returns

**bool** True on success, false if the call fails.
8.175.2.2  createImageEncoder()

virtual bool IDOMImage::createImageEncoder (  
    ISessionPtr & session,  
    IOutputStreamPtr & imageDest,  
    IImageEncoderPtr & imageEncoder,  
    IDOMImagePropertiesPtr & imageProperties ) [pure virtual]

Creates a properly initialized image encoder object that writes to an output stream that is specific to that image format.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>Pointer to the EDL session</td>
</tr>
<tr>
<td>imageDest</td>
<td>Smart pointer to the output stream</td>
</tr>
<tr>
<td>imageEncoder</td>
<td>Smart pointer to the image encoder interface</td>
</tr>
<tr>
<td>imageProperties</td>
<td>Smart pointer to the DOM image properties interface</td>
</tr>
</tbody>
</table>

Returns

bool True on success, false if the call fails.

8.175.2.3  getFirstImageFrame()

virtual bool IDOMImage::getFirstImageFrame (  
    IEDLClassFactory ∗ factory,  
    IImageFramePtr & frame ) [virtual]

Convenience routine to retrieve the first frame in an image.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>factory</td>
<td>Pointer to the EDL class factory</td>
</tr>
<tr>
<td>frame</td>
<td>Smart pointer to receive the image frame.</td>
</tr>
</tbody>
</table>

Returns

bool True on success, false if the call fails.

8.175.2.4  getImageProperties()

virtual bool IDOMImage::getImageProperties (  
    IDOMImagePropertiesPtr & imageProperties ) [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.
Parameters

| imageProperties | Smart pointer to receive the DOM image properties interface. Image properties can be a boolean, an integer or a string. At present it is used to store information such as whether the image has an alpha value. |

Returns

**bool** True on success, false if the call fails.

### 8.175.2.5 getImageType()

virtual eDOMImageType IDOMImage::getImageType() [pure virtual]

Retrieves the image type.

Returns

**eDOMImageType** The image type

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

- idomimageresource.h

### 8.176 IDOMImageBitScalerFilter Class Reference

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

```cpp
#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. Throws an IEDLError on failure.*

- static const CClassID & classID ()
  
  *Retrieves class id of IDOMImageBitScalerFilterClassID.*
8.176.1 Detailed Description

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

8.176.2 Member Function Documentation

8.176.2.1 classID()

static const CClassID& IDOMImageBitScalerFilter::classID() [inline], [static]

Retrieves class id of IDOMImageBitScalerFilterClassID.

Returns

CClassID Class id of the element

8.176.2.2 create()

static EDL_API IDOMImageBitScalerFilterPtr IDOMImageBitScalerFilter::create (IEDLClassFactory * pFactory,
uint8 bps ) [static]

Simplified creator for a bit scaling filter. Throws an IEDLError on failure.

Parameters

<table>
<thead>
<tr>
<th>pFactory</th>
<th>The EDL Class Factory</th>
</tr>
</thead>
<tbody>
<tr>
<td>bps</td>
<td>The desired bits per sample (1, 3, 4, 8, 12 or 16 bit)</td>
</tr>
</tbody>
</table>

Returns

IDOMBitScalerFilterPtr The new filter.

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

- idomimageresource.h

8.177 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. Throws an IEDLError on failure.

• static const CClassID & classID ()
  
  Retrieves class id of IDOMImageBleederFilter.

8.177.1 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.

8.177.2 Member Function Documentation

8.177.2.1 classID()

static const CClassID& IDOMImageBleederFilter::classID () [inline], [static]

Retrieves class id of IDOMImageBleederFilter.

Returns

  CClassID Class id of the element

8.177.2.2 create()

static EDL_API IDOMImageBleederFilterPtr IDOMImageBleederFilter::create {
  IEDLClassFactory * pFactory,
  uint16 bleedPixels = 1 } [static]

Simplified creator for an image bleeder. Throws an IEDLError on failure.

Parameters

<table>
<thead>
<tr>
<th>pFactory</th>
<th>The EDL Class Factory</th>
</tr>
</thead>
<tbody>
<tr>
<td>bleedPixels</td>
<td>The number of pixels to bleed the image.</td>
</tr>
</tbody>
</table>
Returns

IDOMImageBleederFilterPtr The new filter.

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

- idomimageresource.h

8.178  IDOMImageBrush Class Reference

Provides an interface to a DOM image brush object.

#include <idombrush.h>

Inheritance diagram for IDOMImageBrush:

```
+--- IRCObject
 |    +--- IEDLObject
 |         +--- IDOMBrush
 |                     +--- IDOMHashable
 |                     +--- IDOMTransformableBrush
 +--- IDOMImageBrush
     +--- IDOMMaskedBrush
```

Classes

- class Data

  Initialization data.

Generated by Doxygen
Public Member Functions

- virtual eTilingMode getTileMode () const =0
  Retrieves the tiling mode value of the image brush.
- virtual bool setTileMode (eTilingMode tm)=0
  Sets the tiling mode of the image brush.
- virtual eViewUnitsgetViewBoxUnits () const =0
  Retrieves the viewbox units used by the image brush. Currently, only absolute units are supported.
- virtual bool 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 bool setViewPortUnits (eViewUnits vu)=0
  Sets the viewport units used for the image brush. Currently, only absolute units are supported.
- virtual bool getTileBox (FRect &vb) const =0
  Retrieves the viewbox rectangle.
- virtual bool setTileBox (const FRect &vb)=0
  Sets viewbox rectangle.
- virtual bool getViewport (FRect &vp) const =0
  Retrieves the viewport rectangle.
- virtual bool setViewport (const FRect &vp)=0
  Sets the viewport rectangle.
- virtual bool getICCProfile (IDOMICCProfilePtr &icc) const =0
  Retrieves the ICC profile of the brush.
- virtual bool setICCProfile (const IDOMICCProfilePtr &icc)=0
  Sets ICC profile.
- virtual bool getImageSource (IDOMImagePtr &ptrImageSource) const =0
  Retrieves a smart pointer to the image resource.
- virtual bool setImageSource (const IDOMImagePtr &ptrImageSource)=0
  Sets the image resource for the brush.
- virtual bool getEquivalentTilingBrush (IEDLClassFactory ∗pFactory, IDOMTilingPatternBrushPtr &tiling)=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. Exceptions of type IEDLError are thrown on failure. Not available for masked brushes as tiling is not supported for those brushes.

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)
  Simplified creator for an image brush. Throws an IEDLError on failure.
- static const CClassID & classID ()
  Retrieves class id of IDOM.

Additional Inherited Members

8.178.1 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.
8.178.2 Member Function Documentation

8.178.2.1 classID()

static const CClassID & IDOMImageBrush::classID () [inline], [static]

Retrieves class id of IDOM.

Returns

CClassID Class id of the element

8.178.2.2 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 ) [static]

Simplified creator for an image brush. Throws an IEDLError on failure.

Parameters

| pFactory | The factory to use. |
| image    | The image to use.   |
| viewBox  | The desired view box. If empty, the viewPort will be set to the full area of the image. |
| viewPort | The desired view port. |
| renderTransform | The desired render transform. |
| opacity | The desired brush opacity. |
| tileMode | The desired tile mode. |

Returns

IDOMImageBrushPtr The new brush.

8.178.2.3 getEquivalentTilingBrush()

virtual bool IDOMImageBrush::getEquivalentTilingBrush ( IEDLClassFactory * pFactory, IDOMTilingPatternBrushPtr & tiling ) [pure virtual]
Gets an equivalent IDOMTilingPattern brush. If the receiver has a tile mode of eNoTile, this call will fail.
Parameters

<table>
<thead>
<tr>
<th>pFactory</th>
<th>A pointer to an EDL class factory</th>
</tr>
</thead>
<tbody>
<tr>
<td>tiling</td>
<td>A reference to receive the equivalent tiling brush</td>
</tr>
</tbody>
</table>

Returns

bool Returns true on success.

8.178.2.4 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. Exceptions of type IEDLError are thrown on failure. Not available for masked brushes as tiling is not supported for those brushes.

Parameters

| pFactory | A pointer to an EDL class factory |

Returns

IDOMVisualBrushPtr The created visual brush.

8.178.2.5 getICCProfile()

virtual bool IDOMImageBrush::getICCProfile ( IDOMICCProfilePtr & icc ) const [pure virtual]

Retrieves the ICC profile of the brush.

Parameters

| icc | Reference parameter to receive the ICC profile |

Returns

bool. Returns true on success, false if the call fails.
getimageSource()

virtual bool IDOMImageBrush::getImageSource (  
    IDOMImagePtr & ptrImageSource ) const [pure virtual]

Retrieves a smart pointer to the image resource.

Parameters

| ptrImageSource | Smart pointer to receive the image resource. |

Returns

bool. Returns true on success, false if the call fails.

getTileMode()

virtual eTilingMode IDOMImageBrush::getTileMode ( ) const [pure virtual]

Retrieves the tiling mode value of the image brush.

See also

eTilingMode

Returns

eTilingMode. Returns the image tiling mode.

giveViewBox()

virtual bool IDOMImageBrush::getViewBox (  
    FRect & vb ) 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.

Parameters

| vb | The viewbox rectangle |
8.178 IDOMImageBrush Class Reference

Returns

bool Returns true on success

8.178.2.9 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. Returns the viewbox units.

8.178.2.10 getViewPort()

virtual bool IDOMImageBrush::getViewPort ( 
    FRect & vp ) 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.

Parameters

| vp | Reference parameter to receive the viewport rectangle. |

Returns

bool. Returns true on success, false if the call fails.

8.178.2.11 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. Returns the viewport units.

8.178.2.12 setICCProfile()

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

Sets ICC profile.

Parameters

<table>
<thead>
<tr>
<th>icc</th>
<th>The ICC profile</th>
</tr>
</thead>
</table>

Returns

bool. Returns true on success, false if the call fails.

8.178.2.13 setImageSource()

virtual bool IDOMImageBrush::setImageSource (  
    const IDOMImagePtr & ptrImageSource ) [pure virtual]

Sets the image resource for the brush.

Parameters

<table>
<thead>
<tr>
<th>ptrImageSource</th>
<th>Smart pointer to the new image resource.</th>
</tr>
</thead>
</table>

Returns

bool. Returns true on success, false if the call fails.

8.178.2.14 setTileMode()

virtual bool IDOMImageBrush::setTileMode (  
    eTilingMode tm ) [pure virtual]
Sets the tiling mode of the image brush.
Parameters

| tm  | The tiling mode value |

Returns

bool. Returns true on success, false if the call fails.

8.178.2.15 setViewBox()

virtual bool IDOMImageBrush::setViewBox (const FRect & vb) [pure virtual]

Sets viewbox rectangle.

The viewbox specifies the portion of a source image or visual to be rendered to the page as a tile. The tile is then used to fill the geometry specified by the parent element according to the TileMode() function. The viewbox can specify a region larger than the image itself, including negative values. The viewbox specifies the position and dimension of the brush's source content. It is specified by four comma-separated real numbers (x, y, width, height) where width and height are non-negative.

Parameters

| vb   | The viewbox rectangle |

Returns

bool Returns true on success

8.178.2.16 setViewBoxUnits()

virtual bool IDOMImageBrush::setViewBoxUnits (eViewUnits vu) [pure virtual]

Sets the viewbox units value of the image brush. Currently, only absolute units are supported.

See also

getViewBox()
setAddressBox()
Returns

bool. Returns true on success, false if the call fails.

8.178.2.17 setViewport()

virtual bool IDOMImageBrush::setViewport ( const FRect & vp ) [pure virtual]

Sets the viewport rectangle.

The viewport specifies the dimensions and location of the initial tile that will be filled with the specified image or visual fragment. It is defined in the current effective coordinate space. It is specified by four comma separated real numbers (x, y, width, height) where width and height are non-negative.

Parameters

vp The viewport rectangle

Returns

bool. Returns true on success, false if the call fails.

8.178.2.18 setViewportUnits()

virtual bool IDOMImageBrush::setViewportUnits ( eViewUnits vu ) [pure virtual]

Sets the viewport units used for the image brush. Currently, only absolute units are supported.

See also

getViewPort()
setViewport()

Parameters

vu The new viewport units value

Returns

bool. Returns true on success, false if the call fails.

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

- idombrush.h
8.179 IDOMImageChannelSelectorFilter Class Reference

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

```c
#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)
  
  *Simplified creator for a color converter filter. Throws an IEDLError on failure.*

- static const CClassID & classID ()
  
  *Retrieves class id of IDOMImageChannelSelectorFilter.*

8.179.1 Detailed Description

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

8.179.2 Member Function Documentation

8.179.2.1 classID()

```cpp
static const CClassID& IDOMImageChannelSelectorFilter::classID () [inline], [static]
```

Retrieves class id of IDOMImageChannelSelectorFilter.

Returns

- **CClassID** Class id of the element

8.179.2.2 create()

```cpp
static EDL_API IDOMImageChannelSelectorFilterPtr IDOMImageChannelSelectorFilter::create ( 
    IEDLClassFactory * pFactory, 
    eSelection selection ) [static]
```

Simplified creator for a color converter filter. Throws an IEDLError on failure.
Parameters

<table>
<thead>
<tr>
<th>pFactory</th>
<th>The EDL Class Factory</th>
</tr>
</thead>
<tbody>
<tr>
<td>selection</td>
<td>Which channel(s) to retain during filtering.</td>
</tr>
</tbody>
</table>

Returns

IDOMImageChannelSelectorFilterPtr The new filter.

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

- idomimageresource.h

8.180 IDOMImageColorConverterFilter Class Reference

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

#include <idomimageresource.h>

Inherits IDOMImageFilter.

Classes

- class Data
  
  *Initialization data.*

Static Public Member Functions

- static EDL_API IDOMImageColorConverterFilterPtr create (IEDLClassFactory *pFactory, const IDOMColor ← SpacePtr &space, eRenderingIntent intent, eBlackPointCompensation bpc)
  
  *Simplified creator for a color converter filter. Throws an IEDLError on failure.*

- static const CClassID & classID ()
  
  *Retrieves class id of IDOMImageColorConverterFilter.*

8.180.1 Detailed Description

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

8.180.2 Member Function Documentation
8.180.2.1 classID()

```
static const CClassID & IDOMImageColorConverterFilter::classID ( ) [inline], [static]
```

Retrieves class id of IDOMImageColorConverterFilter.

Returns

**CClassID** Class id of the element

8.180.2.2 create()

```
static EDL_API IDOMImageColorConverterFilterPtr IDOMImageColorConverterFilter::create (  
    IEDLClassFactory * pFactory,  
    const IDOMColorSpacePtr & space,  
    eRenderingIntent intent,  
    eBlackPointCompensation bpc ) [static]
```

Simplified creator for a color converter filter. Throws an **IEDLError** on failure.

Parameters

<table>
<thead>
<tr>
<th>pFactory</th>
<th>The EDL Class Factory</th>
</tr>
</thead>
<tbody>
<tr>
<td>space</td>
<td>The target color space. Must be a base space; ie no DeviceN or Indexed spaces.</td>
</tr>
<tr>
<td>intent</td>
<td>The rendering intent to use</td>
</tr>
<tr>
<td>bpc</td>
<td>The desired black point compensation treatment. If in doubt, use eBPCDefault.</td>
</tr>
</tbody>
</table>

Returns

**IDOMImageColorConverterFilterPtr** The new filter.

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

- idomimageresource.h

8.181 IDOMImageColorKeyFilter Class Reference

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

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

Inherits IDOMImageFilter.

Classes

- **class Data**

  * Initialization data.*
8.181 IDOMImageColorKeyFilter Class Reference

Static Public Member Functions

- static EDL_API IDOMImageColorKeyFilterPtr create (IEDLClassFactory ∗pFactory, const CEDLVector< uint16 > &key)
  
  Simplified creator for a color key scaling filter. Throws an IEDLError on failure.
- static const CClassID & classID ()
  
  Retrieves class id of IDOMImageColorKeyFilter.

8.181.1 Detailed Description

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

8.181.2 Member Function Documentation

8.181.2.1 classID()

static const CClassID & IDOMImageColorKeyFilter::classID ( ) [inline], [static]

Retrieves class id of IDOMImageColorKeyFilter.

Returns

  CClassID Class id of the element

8.181.2.2 create()

static EDL_API IDOMImageColorKeyFilterPtr IDOMImageColorKeyFilter::create ( IEDLClassFactory ∗pFactory,
  const CEDLVector< uint16 > & key ) [static]

Simplified creator for a color key scaling filter. Throws an IEDLError on failure.

Parameters

<table>
<thead>
<tr>
<th>pFactory</th>
<th>The EDL Class Factory</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>A vector that describes the range of colours to be masked out. In the form low1, hi1, low2, hi2, ... lowN, hiN where N is the number of color components in the source image. The values are expressed in the range of the samples in the image. For example, if the image is 4 bits per component then the values in this vector need to be between 0 and 15 inclusive.</td>
</tr>
</tbody>
</table>
Returns

`IDOMImageColorKeyFilterPtr` The new filter.

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

- `idomimageresource.h`

### 8.182 IDOMImageColorSpaceSubstitutionFilter Class Reference

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

```cpp
#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. Throws an IEDLError on failure.*

- **static const CClassID & classID ()**
  
  *Retrieves class id of IDOMImageColorSpaceSubstitutionFilterClassID.*

### 8.182.1 Detailed Description

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

### 8.182.2 Member Function Documentation

#### 8.182.2.1 classID()  

```cpp
static const CClassID & IDOMImageColorSpaceSubstitutionFilter::classID () [inline], [static]
```

*Retrieves class id of IDOMImageColorSpaceSubstitutionFilterClassID.*

*Returns*

- **CClassID** Class id of the element

#### 8.182.2.2 create()  

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

*Simplified creator for a color space substitutor filter. Throws an IEDLError on failure.*
Parameters

<table>
<thead>
<tr>
<th>pFactory</th>
<th>The EDL Class Factory</th>
</tr>
</thead>
<tbody>
<tr>
<td>colorSpace</td>
<td>The color space to substitute.</td>
</tr>
</tbody>
</table>

Returns

IDOMImageColorSpaceSubstitutionFilterPtr The new filter.

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

- idomimageresource.h

8.183 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. Throws an IEDLError on failure.*

- static const CClassID & classID ()
  
  *Retrieves class id of IDOMImageDecodeFilter.*

8.183.1 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.

8.183.2 Member Function Documentation
8.183.2.1 classID()

static const CClassID & IDOMImageDecodeFilter::classID ( ) [inline], [static]

Retrieves class id of IDOMImageDecodeFilter.

Returns

  CClassID Class id of the element

8.183.2.2 create()

static EDL_API IDOMImageDecodeFilterPtr IDOMImageDecodeFilter::create ( 
  IEDLClassFactory * pFactory,
  const CEDLVector< float > & decode ) [static]

Simplified creator for a decode filter. Throws an IEDLError on failure.

Parameters

<table>
<thead>
<tr>
<th>pFactory</th>
<th>The EDL Class Factory</th>
</tr>
</thead>
<tbody>
<tr>
<td>decode</td>
<td>The decode vector.</td>
</tr>
</tbody>
</table>

Returns

  IDOMImageDecodeFilterPtr The new filter.

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

• idomimageresource.h

8.184 IDOMImageDeindexerFilter Class Reference

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

#include <idomimageresource.h>

Inherits IDOMImageFilter.

Classes

• class Data

  Initialization data.
Static Public Member Functions

- static EDL_API IDOMImageDeindexerFilterPtr create (IEDLClassFactory *pFactory)
  Simplified creator for a deindexer filter. Throws an IEDLError on failure.
- static const CClassID & classID ()
  Retrieves class id of IDOMImageDeindexerFilter.

8.184.1 Detailed Description

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

8.184.2 Member Function Documentation

8.184.2.1 classID()

static const CClassID& IDOMImageDeindexerFilter::classID ( ) [inline], [static]

Retrieves class id of IDOMImageDeindexerFilter.

Returns

  CClassID Class id of the element

8.184.2.2 create()

static EDL_API IDOMImageDeindexerFilterPtr IDOMImageDeindexerFilter::create ( IEDLClassFactory * pFactory ) [static]

Simplified creator for a deindexer filter. Throws an IEDLError on failure.

Parameters

  pFactory  The EDL Class Factory

Returns

  IDOMImageDeindexerFilterPtr The new filter.

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

- idomimageresource.h
8.185  IDOMImageDeviceNToBaseFilter Class Reference

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

#include <idomimageresource.h>

Inherits IDOMImageFilter.

Classes

• class Data
  
  Initialization data.

Static Public Member Functions

• static EDL_API IDOMImageDeviceNToBaseFilterPtr create (IEDLClassFactory *pFactory)
  
  Simplified creator for this filter Throws an IEDLError on failure.

• static const CClassID & classID ()
  
  Retrieves class id of IDOMImageDeindexerFilter.

8.185.1 Detailed Description

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

8.185.2 Member Function Documentation

8.185.2.1 classID()

static const CClassID & IDOMImageDeviceNToBaseFilter::classID () [inline], [static]

Retrieves class id of IDOMImageDeindexerFilter.

Returns

  CClassID Class id of the element

8.185.2.2 create()

static EDL_API IDOMImageDeviceNToBaseFilterPtr IDOMImageDeviceNToBaseFilter::create (IEDLClassFactory * pFactory ) [static]

Simplified creator for this filter Throws an IEDLError on failure.
Parameters

\textbf{\texttt{pFactory}} | The EDL Class Factory

Returns

\textbf{IDOMImageDeviceNToBaseFilterPtr} The new filter.

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

- \texttt{idomimageresource.h}

### 8.186 \ IDOMImageDownsamplerFilter Class Reference

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

#include <idomimageresource.h>

Inherits IDOMImageFilter.

#### Classes

- class \textbf{Data}
  
  \emph{Initialization data.}

#### Public Types

- enum \textbf{eDownsamplingMethod}
  
  \emph{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)
  
  \emph{Simplified creator for an image downsampler filter. Throws an \texttt{IEDLError} on failure.}

- static const CClassID & classID ()
  
  \emph{Retrieves class id of IDOMImageDownsamplerFilter.}

### 8.186.1 Detailed Description

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

### 8.186.2 Member Function Documentation

Generated by Doxygen
8.186.2.1 classID()

static const CClassID & IDOMImageDownsamplerFilter::classID() [inline], [static]

Retrieves class id of IDOMImageDownsamplerFilter.

Returns

CClassID Class id of the element

8.186.2.2 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. Throws an IEDLError on failure.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pFactory</td>
<td>The EDL Class Factory</td>
</tr>
<tr>
<td>xRes</td>
<td>The desired x resolution of the result.</td>
</tr>
<tr>
<td>yRes</td>
<td>The desired y resolution of the result. If 0, the x resolution will be used.</td>
</tr>
<tr>
<td>method</td>
<td>The desired downsampling method to use.</td>
</tr>
</tbody>
</table>

Returns

IDOMImageDownsamplerFilterPtr The new filter.

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

* idomimageresource.h

8.187 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 Throws an IEDLError on failure.*

- static const CClassID & classID ()
  
  *Retrieves class id of IDOMImageInverterFilter.*

8.187.1 Detailed Description

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

8.187.2 Member Function Documentation

8.187.2.1 classID()

static const CClassID & IDOMImageInverterFilter::classID () [inline], [static]

*Retrieves class id of IDOMImageInverterFilter.*

Returns

  **CClassID** Class id of the element

8.187.2.2 create()

static EDL_API IDOMImageInverterFilterPtr IDOMImageInverterFilter::create (IEDLClassFactory * pFactory ) [static]

*Simplified creator for this filter Throws an IEDLError on failure.*

Parameters

| **pFactory** | The EDL Class Factory |

Returns

  **IDOMDePremultiplierFilterPtr** The new filter.

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

- idomimageresource.h

---

Generated by Doxygen
8.188  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 a IDOMMaskedBrush where the brush masked by the image is a solid color.

```cpp
#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. Throws an IEDLError on failure.

- static const CClassID & classID ()
  
  Retrieves class id of IDOMImageMaskExpanderFilter.

8.188.1 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 a IDOMMaskedBrush where the brush masked by the image is a solid color.

8.188.2 Member Function Documentation

8.188.2.1 classID()

```cpp
static const CClassID & IDOMImageMaskExpanderFilter::classID ( ) [inline], [static]
```

Retrieves class id of IDOMImageMaskExpanderFilter.

Returns

**CClassID** Class id of the element

8.188.2.2 create()

```cpp
static EDL_API IDOMImageMaskExpanderFilterPtr IDOMImageMaskExpanderFilter::create ( 
  IEDLClassFactory * pFactory,
  const IDOMColorPtr & color ) [static]
```

Simplified creator for a mask expander filter. Throws an IEDLError on failure.
Parameters

<table>
<thead>
<tr>
<th>pFactory</th>
<th>The EDL Class Factory</th>
</tr>
</thead>
<tbody>
<tr>
<td>color</td>
<td>The color to use.</td>
</tr>
</tbody>
</table>

Returns

IDOMImageMaskExpanderFilterPtr The new filter.

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

- idomimageresource.h

8.189 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:

```
IRCOObject
    ↓
IEDLObject
    ↓
IDOMImageProperties
```

Public Member Functions

- virtual bool setProperty (const EDLString &name, const EDLString &value)=0
  
  Registers the name of a property and a string value for the property.
- virtual bool getBoolProperty (const EDLString &name, bool &value)=0
  
  Retrieves the value of a named property as a string value.
- virtual bool 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 bool 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 bool 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.

Static Public Member Functions

- static const CClassID & classID ()
  Retrieves class id of IDOMImageProperties.

Additional Inherited Members

8.189.1 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.

8.189.2 Member Function Documentation

8.189.2.1 classID()

static const CClassID & IDOMImageProperties::classID ( ) [inline], [static]
Retrieves class id of IDOMImageProperties.

Returns

CClassID Class id of the element

8.189.2.2 getBoolProperty()

virtual bool IDOMImageProperties::getBoolProperty ( const EDLString & name, bool & value ) [pure virtual]
Retrieves the value of a named property as a Boolean value.
### Parameters

<table>
<thead>
<tr>
<th>name</th>
<th>Property name.</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Reference parameter to receive the property value.</td>
</tr>
</tbody>
</table>

**Returns**

**bool** True on success, false if the call fails. The value parameter will contain the property value.

#### 8.189.2.3 getIntProperty()

```cpp
virtual bool IDOMImageProperties::getIntProperty ( 
    const EDLString & name, 
    int & value ) [pure virtual]
```

Retrieves the value of a named property as an integer value.

**Parameters**

<table>
<thead>
<tr>
<th>name</th>
<th>Property name.</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Reference parameter to receive the property value.</td>
</tr>
</tbody>
</table>

**Returns**

**bool** True on success, false if the call fails. The value parameter will contain the property value.

#### 8.189.2.4 getObjectProperty()

```cpp
virtual bool IDOMImageProperties::getObjectProperty ( 
    const EDLString & name, 
    IRCObjectPtr & value ) [pure virtual]
```

Retrieves the value of a named property as an **IRCOBJ**ect value.

**Parameters**

<table>
<thead>
<tr>
<th>name</th>
<th>Property name.</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>An <strong>IRCOBJ</strong>ect reference to receive the property value.</td>
</tr>
</tbody>
</table>

**Returns**

**bool** True on success, false if the call fails. The value parameter will contain the property value.
8.189.2.5 getProperty()

virtual bool IDOMImageProperties::getProperty ( const EDLString & name, EDLString & value ) [pure virtual]

Retrieves the value of a named property as a string value.

Parameters

<table>
<thead>
<tr>
<th>name</th>
<th>Property name.</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Reference parameter to receive the property value.</td>
</tr>
</tbody>
</table>

Returns

bool True on success, false if the call fails. The value parameter will contain the property value.

8.189.2.6 setBoolProperty()

virtual bool IDOMImageProperties::setBoolProperty ( const EDLString & name, bool value ) [pure virtual]

Registers the name of a property and a Boolean value for the property.

Parameters

<table>
<thead>
<tr>
<th>name</th>
<th>Property name.</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>The Boolean value for the property.</td>
</tr>
</tbody>
</table>

Returns

bool True on success, false if the call fails.

8.189.2.7 setIntProperty()

virtual bool IDOMImageProperties::setIntProperty ( const EDLString & name, int value ) [pure virtual]

Registers the name of a property and an integer value for the property.

Parameters

<table>
<thead>
<tr>
<th>name</th>
<th>Property name.</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>The integer value for the property.</td>
</tr>
</tbody>
</table>
Returns

**bool** True on success, false if the call fails.

### 8.189.2.8 setObjectProperty()

```cpp
virtual bool IDOMImageProperties::setObjectProperty (
    const EDLString & name,
    const IRCObjectPtr & object ) [pure virtual]
```

Registers the name of a property and an **IRCObject** value for the property.

**Parameters**

<table>
<thead>
<tr>
<th>name</th>
<th>Property name.</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>An <strong>IRCObject</strong> reference to the value for the property.</td>
</tr>
</tbody>
</table>

Returns

**bool** True on success, false if the call fails.

### 8.189.2.9 setProperty()

```cpp
virtual bool IDOMImageProperties::setProperty (
    const EDLString & name,
    const EDLString & value ) [pure virtual]
```

Registers the name of a property and a string value for the property.

**Parameters**

<table>
<thead>
<tr>
<th>name</th>
<th>Property name.</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>The string value for the property.</td>
</tr>
</tbody>
</table>

Returns

**bool** True on success, false if the call fails.

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

- `idomimageresource.h`

---

Generated by Doxygen
The IDOMInternalTarget interface describes the targets of hyperlinks that are in the same document but not on the current page.

```cpp
#include <idomtarget.h>
```

Inheritance diagram for IDOMInternalTarget:

```
IRCOBJECT

IEDLOBJECT

IDOMTARGET

IDOMInternalTarget
```

### Public Member Functions

- virtual `DOMid getTargetId () const =0`
  
  Retrieves the internal target id.

- virtual void `setTargetId (DOMid id)=0`
  
  Sets the internal target id.

- virtual `eTargetType getTargetType () const`

  Implementation of getTargetType for IDOMInternalTarget.

### Additional Inherited Members

#### 8.190.1 Detailed Description

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

#### 8.190.2 Member Function Documentation
8.190.2.1 getTargetId()

virtual DOMid IDOMInternalTarget::getTargetId ( ) const [pure virtual]

Retrieves the internal target id.

Returns
DOMid. Returns the internal target id

8.190.2.2 getTargetType()

virtual eTargetType IDOMInternalTarget::getWidthType ( ) const [inline], [virtual]

Implementation of getTargetType for IDOMInternalTarget.

Returns
eTargetType Returns eInternal;

Implements IDOMTarget.

8.190.2.3 setTargetId()

virtual void IDOMInternalTarget::setWidthType ( ) const [pure virtual]

Sets the internal target id.

Parameters

| id. | The internal target id. |

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

- idomtarget.h

8.191 IDOMJobTk Class Reference

Represents an EDL JobTicket.

#include <idomjobtk.h>
Inheritance diagram for IDOMJobTk:

```
IRCOBJECT
  ＼
  ＼
  ＼
IEDEOBJECT
  ＼
  ＼
  ＼
IDOMJobTk
```

Public Member Functions

- virtual bool 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 bool setContent (const IDOMJobTkContentPtr &ptrContent)=0
  
  Sets the JobTicket content node.

- virtual IDOMJobTkContentPtr getContent () const =0
  
  Retrieves the JobTicket content node.

- virtual IDOMJobTkNodePtr findChild (IDOMJobTkNode::eDOMJobTkNodeType nodeType, const EDLString &name, const EDLSysString &nmspace)=0
  
  Searches through the node's direct child set for the node type, name and namespace matching the information provided.

- virtual IDOMJobTkNodePtr findChild (IDOMJobTkNode::eDOMJobTkNodeType nodeType, const EDLQName &qname)=0
  
  Searches through the node's direct child set for the node type and qname matching the information provided.

- virtual bool getCombinedContent (IDOMJobTkContentPtr &ptrCombinedContent, bool addDefaultJob=false)=0
  
  Generate combined JobTicket content. Combined content is generated by combining the current JobTicket with all upper level JobTickets.

Static Public Member Functions

- static const CClassID & classID ()
  
  Retrieves the class id of IDOMJobTk.
Additional Inherited Members

8.191.1 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.

8.191.2 Member Function Documentation

8.191.2.1 `classID()`

```cpp
static const CClassID & IDOMJobTk::classID ( ) [inline], [static]
```

Retrieves the class id of IDOMJobTk.

Returns

CClassID. Class id of the element.

8.191.2.2 `findChild()` [1/2]

```cpp
virtual IDOMJobTkNodePtr IDOMJobTk::findChild ( 
    IDOMJobTkNode::eDOMJobTkNodeType nodeType,
    const EDLString & name,
    const EDLSysString & nmspace ) [pure virtual]
```

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

Parameters

| `nodeType` | Type of the node’s child to search for. |
| `name`     | Name of the node’s child to search for. |
| `nmspace`  | Namespace of the node’s child to search for. |

Returns

IDOMJobTkNodePtr Smart pointer to the found child node.
virtual IDOMJobTkNodePtr IDOMJobTk::findChild ( 
    IDOMJobTkNode::eDOMJobTkNodeType nodeType, 
    const EDLMName & qname ) [pure virtual]

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

Parameters

<table>
<thead>
<tr>
<th>nodeType</th>
<th>Node type of the the node's child to search for.</th>
</tr>
</thead>
<tbody>
<tr>
<td>qname</td>
<td>Qualified name of the the node's child to search for.</td>
</tr>
</tbody>
</table>

Returns

IDOMJobTkNodePtr Smart pointer to the found child node.

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

<table>
<thead>
<tr>
<th>ptrCombinedContent</th>
<th>A smart pointer to receive the resulting combined content.</th>
</tr>
</thead>
<tbody>
<tr>
<td>addDefaultJobTk</td>
<td>If this parameter is set to &quot;true&quot; then combine the default level of JobTicket as well as document, job and page levels.</td>
</tr>
</tbody>
</table>

Returns

bool. Returns true on success, false if the call fails.

virtual IDOMJobTkContentPtr IDOMJobTk::getContent ( ) const [pure virtual]

Retrieves the JobTicket content node.

Returns

IDOMJobTkContentPtr Returns smart pointer to the JobTicket content node
8.191.2.6 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.

8.191.2.7 setContent()

virtual bool IDOMJobTk::setContent ( const IDOMJobTkContentPtr & ptrContent ) [pure virtual]

Sets the JobTicket content node.

Parameters
   \textbf{ptrOwner} | Smart pointer to the new JobTicket content node.

Returns
   bool. Returns true on success, false if the call fails.

8.191.2.8 setOwner()

virtual bool IDOMJobTk::setOwner ( IDOMJobTkOwnerPtr & ptrOwner ) [pure virtual]

Sets the owner node, that is, the level of the DOMtree with which the JobTicket is associated.

Parameters
   \textbf{ptrOwner} | Smart pointer to the new owner node.

Returns
   bool. Returns true on success, false if the call fails.

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

- idomjobtk.h

---

Generated by Doxygen
8.192  IDOMJobTkContent Class Reference

Represents the content element of the JobTicket.

```cpp
#include <idomjobtk.h>
```

Inheritance diagram for IDOMJobTkContent:

![Inheritance Diagram](image)

**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 bool **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 bool **setVersion** (double version)=0
  
  *Sets the version of the JobTicket content.*

- virtual double **getVersion** () const =0

Generated by Doxygen
Retrieves the version of the JobTicket content.

- virtual bool setModified (bool modified)=0
  Sets the "modified" flag.

- virtual bool getModified () const =0
  Retrieves the value of the "modified" flag.

- virtual bool getRootNode (IDOMJobTkNodePtr &rootNode)=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 bool addNamespace (const IEDLNamespacePtr &ptrNamespace)=0
  Appends a namespace to the collection of namespaces.

- virtual IEDLNamespacePtr addNamespace (const EDLSysString &prefix, const EDLSysString &name)=0
  Appends a namespace to the collection of namespaces.

- virtual bool findNamespaceByName (const EDLSysString &nmspace, IEDLNamespacePtr &ptrNamespace)=0
  Finds a namespace in the namespace collection by name.

- virtual bool findNamespaceByPrefix (const EDLSysString &prefix, IEDLNamespacePtr &ptrNamespace)=0
  Finds a namespace in the namespace collection by prefix.

- virtual void addStandardNamespaces ()=0

- virtual bool convertToQName (const EDLString &name, const EDLSysString &nmspace, EDLQName &qname)=0
  Constructs a qualified name based on a name and namespace provided. The namespace must be present in the namespace collection.

- virtual IDOMJobTkNodePtr findChild (IDOMJobTkNode::eDOMJobTkNodeType nodeType, const EDLString &name, const EDLSysString &nmspace)=0
  Searches through the node's direct child set for a node matching the provided node type, name and namespace.

- virtual IDOMJobTkNodePtr findChild (IDOMJobTkNode::eDOMJobTkNodeType nodeType, const EDLQName &qname)=0
  Searches through the node's direct child set for a node matching provided the node type and qualified name.

- virtual bool loadFromFile (const EDLSysString &filename)=0
  Loads a JobTicket from a named XML file.

- virtual bool loadFromStream (const IRAInputStreamPtr &xmlStream)=0
  Loads a JobTicket from an XML stream.

- virtual bool loadFromInitString (const EDLSysString &sInit)=0
  Fills the JobTicket using input from the initialisation string.

- virtual bool addParameterInit (const IEDLNamespacePtr &parameterNamespace, const EDLSysString &parameterName, const PValue &value)=0
  Adds a <psf:ParameterInit> element to the JobTicket.

- virtual bool addFeature (const IEDLNamespacePtr &featureNamespace, const EDLSysString &featureName, const EDLQName &option)=0
  Adds Feature/Option elements to the JobTicket in the format:
Static Public Member Functions

- static const CClassID & classID ()
  Retrieves the class id of IDOMJobTkContent.

- static EDL_API EDLSysString addToInitString (const EDLSysString &paramName, const EDLSysString &initString, const EDLSysString &paramValue="", bool overwrite=true)
  Static helper to add a key/value pair to an initialisation string. Takes care of escaping as required.

Additional Inherited Members

8.192.1 Detailed Description

Represents the content element of the JobTicket.

8.192.2 Member Function Documentation

8.192.2.1 addFeature()

virtual bool 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

<table>
<thead>
<tr>
<th>featureNamespace</th>
<th>Smart pointer to the feature name namespace.</th>
</tr>
</thead>
<tbody>
<tr>
<td>featureName</td>
<td>Name of feature.</td>
</tr>
<tr>
<td>option</td>
<td>Qualified name of feature option.</td>
</tr>
</tbody>
</table>

Returns

bool. Returns true on success, or false if the call fails.
8.192.2.2 addNamespace() [1/2]

virtual bool IDOMJobTkContent::addNamespace (const IEDLNamespacePtr & ptrNamespace) [pure virtual]

Appends a namespace to the collection of namespaces.

Parameters

| ptrNamespace | Smart pointer to the new namespace. |

Returns

bool. Returns true on success, or false if the call fails.

8.192.2.3 addNamespace() [2/2]

virtual IEDLNamespacePtr IDOMJobTkContent::addNamespace (const EDLSysString & prefix, const EDLSysString & name) [pure virtual]

Appends a namespace to the collection of namespaces.

Parameters

| prefix | Value of the new namespace prefix |
| name | Value of the new namespace name |

Returns

IEDLNamespacePtr. Smart pointer to the new namespace which has been added.

8.192.2.4 addParameterInit()

virtual bool IDOMJobTkContent::addParameterInit (const IEDLNamespacePtr & parameterNamespace, const EDLSysString & parameterName, const PValue & value) [pure virtual]

Adds a <psf:ParameterInit> element to the JobTicket.

Parameters

| parameterNamespace | Smart pointer to the parameter namespace. |
| parameterName | Name of parameter. |
| value | Value of parameter. |
Returns

bool. Returns true on success, or false if the call fails.

8.192.2.5 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

<table>
<thead>
<tr>
<th>paramName</th>
<th>The parameter name.</th>
</tr>
</thead>
<tbody>
<tr>
<td>initString</td>
<td>The init string to add to</td>
</tr>
<tr>
<td>paramValue</td>
<td>The optional parameter value.</td>
</tr>
<tr>
<td>overwrite</td>
<td>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.</td>
</tr>
</tbody>
</table>

Returns

EDLSysString Returns the new init string.

8.192.2.6 classID()

static const CClassID & IDOMJobTkContent::classID ( ) [inline], [static]

Retrieves the class id of IDOMJobTkContent.

Returns

CClassID. Returns the class id of the element.

8.192.2.7 convertToQName()

virtual bool IDOMJobTkContent::convertToQName ( const EDLString & name, const EDLSysString & namespace, EDLQName & qname ) [pure virtual]

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

<table>
<thead>
<tr>
<th>name</th>
<th>Name of JobTicket content element</th>
</tr>
</thead>
<tbody>
<tr>
<td>nmspace</td>
<td>Namespace of JobTicket content element</td>
</tr>
<tr>
<td>qname</td>
<td>reference parameter to receive the constructed qualified name.</td>
</tr>
</tbody>
</table>

Returns

bool. Returns true on success, or false if the call fails.

8.192.2.8 findChild() [1/2]

virtual IDOMJobTkNodePtr IDOMJobTkContent::findChild (|
    IDOMJobTkNode::eDOMJobTkNodeType nodeType,|
    const EDLString & name,|
    const EDLSysString & nmspace ) [pure virtual]

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

Parameters

<table>
<thead>
<tr>
<th>nodeType</th>
<th>Job ticket node type.</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of JobTicket node.</td>
</tr>
<tr>
<td>nmspace</td>
<td>Namespace of JobTicket node.</td>
</tr>
</tbody>
</table>

Returns

IDOMJobTkNodePtr. Returns smart pointer to the retrieved JobTicket node.

8.192.2.9 findChild() [2/2]

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

<table>
<thead>
<tr>
<th>nodeType</th>
<th>Job ticket node type.</th>
</tr>
</thead>
<tbody>
<tr>
<td>qname</td>
<td>Qualified name of JobTicket node.</td>
</tr>
</tbody>
</table>
Returns

IDOMJobTkNodePtr. Returns smart pointer to the retrieved JobTicket node.

8.192.2.10 findNamespaceByName()

```cpp
virtual bool IDOMJobTkContent::findNamespaceByName (  
    const EDLSysString & nmspace,  
    IEDLNamespacePtr & ptrNamespace ) [pure virtual]
```

Finds a namespace in the namespace collection by name.

**Parameters**

<table>
<thead>
<tr>
<th>nmspace</th>
<th>Value of namespace name.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ptrNamespace</td>
<td>Smart pointer to receive a reference to the found namespace.</td>
</tr>
</tbody>
</table>

**Returns**

bool. Returns true on success, or false if the call fails.

8.192.2.11 findNamespaceByPrefix()

```cpp
virtual bool IDOMJobTkContent::findNamespaceByPrefix (  
    const EDLSysString & prefix,  
    IEDLNamespacePtr & ptrNamespace ) [pure virtual]
```

Finds a namespace in the namespace collection by prefix.

**Parameters**

<table>
<thead>
<tr>
<th>prefix</th>
<th>Value of namespace prefix.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ptrNamespace</td>
<td>Smart pointer to receive a reference to the found namespace.</td>
</tr>
</tbody>
</table>

**Returns**

bool. Returns true on success, or false if the call fails.

8.192.2.12 getLevel()

```cpp
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.

8.192.2.13 getModified()

virtual bool IDOMJobTkContent::getModified ( ) const [pure virtual]

Retrieves the value of the "modified" flag.

Returns

bool. Value of the "modified" flag.

8.192.2.14 getNamespaceCollectionEnum()

virtual IEDLNamespaceCollectionEnumPtr IDOMJobTkContent::getNamespaceCollectionEnum ( ) [pure virtual]

Retrieves the enumerator of the namespace collection.

Returns

IEDLNamespaceCollectionEnumPtr. Returns a pointer to the enumerator.

8.192.2.15 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.

8.192.2.16 getRootNode()

virtual bool IDOMJobTkContent::getRootNode ( IDOMJobTkNodePtr & rootNode ) [pure virtual]

Retrieves the root node of the JobTicket content.
Parameters

| rootNode | Smart pointer to receive a reference to the root node. |

Returns

bool. Returns true on success, or false if the call fails.

8.192.17 getVersion()

virtual double IDOMJobTkContent::getVersion() const [pure virtual]

Retrieves the version of the JobTicket content.

Returns
double. Returns the version of the JobTicket content.

8.192.18 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.

8.192.19 loadFromFile()

virtual bool IDOMJobTkContent::loadFromFile(
    const EDLSysString & filename) [pure virtual]

Loads a JobTicket from a named XML file.

Parameters

| filename | Path to and name of the JobTicket file to load |
8.192 IDOMJobTkContent Class Reference

Returns

bool. Returns true on success, or false if the call fails.

8.192.2.20 loadFromInitString()

virtual bool IDOMJobTkContent::loadFromInitString (const EDLSysString & sInit) [pure virtual]

Fills the JobTicket using input from the initialisation string.

Parameters

\textit{sInit} | Initialisation string.

Returns

bool. Returns true on success, or false if the call fails.

8.192.2.21 loadFromStream()

virtual bool IDOMJobTkContent::loadFromStream (const IRAInputStreamPtr & xmlStream) [pure virtual]

Loads a JobTicket from an XML stream.

Parameters

\textit{xmlStream} | Stream for xml data

Returns

bool. Returns true on success, or false if the call fails.

8.192.2.22 setLevel()

virtual bool 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

| level | Job ticket level. |

Returns

bool. Returns true on success, or false if the call fails.

8.192.23  setModified()

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

Sets the "modified" flag.

Parameters

| modified | New value of the "modified" flag. |

Returns

bool. Returns true on success, or false if the call fails.

8.192.24  setVersion()

virtual bool IDOMJobTkContent::setVersion ( double version ) [pure virtual]

Sets the version of the JobTicket content.

Parameters

| version | Job ticket version. |

Returns

bool. Returns true on success, or false if the call fails.

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

- idomjobtk.h
8.193 IDOMJobTkGenericCharacterData Class Reference

Interface to the IDOMJobTkGenericCharacterData node.

#include <idomjobtk.h>

Inheritance diagram for IDOMJobTkGenericCharacterData:

```
IRCOBJECT

IEDLOBJECT

IDOMNODE

IDOMJobTkGenericCharacterData
```

Classes

- class Data

  Initialization data.

Public Member Functions

- virtual bool getCharacterData (EDLString &charData) const =0

  Get the node’s CDATA.

- virtual bool setCharacterData (const EDLString &charData)=0

  Set the node’s CDATA.

- virtual bool appendCharacterData (const EDLString &charData)=0

  Append to the node’s CDATA.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves the class id of IDOMJobTkGenericNode.
8.193.1 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.

8.193.2 Member Function Documentation

8.193.2.1 appendCharacterData()

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

Append to the node's CDATA.

Parameters

| charData | The additional CDATA as an EDLString |

Returns

bool. Returns true on success, false if the call fails.

8.193.2.2 classID()

static const CClassID & IDOMJobTkGenericCharacterData::classID ( ) [inline], [static]

Retrieves the class id of IDOMJobTkGenericNode.

Returns

CClassID. Class id of the element.

8.193.2.3 getCharacterData()

virtual bool IDOMJobTkGenericCharacterData::getCharacterData ( EDLString & charData ) const [pure virtual]

Get the node's CDATA.
Parameters

| charData | Reference to receive the CDATA |

Returns

bool. Returns true on success, false if the call fails.

8.193.2.4 setCharacterData()

virtual bool IDOMJobTkGenericCharacterData::setCharacterData (const EDLString & charData) [pure virtual]

Set the node’s CDATA.

Parameters

| charData | The CDATA as an EDLString |

Returns

bool. Returns true on success, false if the call fails.

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

- idomjobtk.h

8.194 IDOMJobTkGenericNode Class Reference

Interface to the IDOMJobTkGenericNode node.

#include <idomjobtk.h>

Generated by Doxygen
Inheritance diagram for IDOMJobTkGenericNode:

```
IRCOBJECT

IEDLObject

IDOMNode

IDOMJobTkGenericNode
```

Classes

- class **Data**
  
  *Initialization data.*

Public Member Functions

- virtual bool **getQName** (EDLQName &qname) const =0
  
  *Retrieves the node qname.*

- virtual bool **addAttribute** (const EDLQName &name, const EDLSysString &value)=0
  
  *Adds an attribute.*

- virtual uint32 **getNumAttributes** () const =0
  
  *Gets the number of attributes.*

- virtual bool **getAttributeAtIndex** (uint32 index, EDLQName &name, EDLSysString &value) const =0
  
  *Get an attribute at the given index.*

Static Public Member Functions

- static const **CClassID & classID** ()
  
  *Retrieves the class id of IDOMJobTkGenericNode.*
Additional Inherited Members

8.194.1 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.

8.194.2 Member Function Documentation

8.194.2.1 addAttribute()

virtual bool IDOMJobTkGenericNode::addAttribute (  
    const EDLQName & name,  
    const EDLSysString & value ) [pure virtual]

Adds an attribute.

Parameters

<table>
<thead>
<tr>
<th>qname</th>
<th>Qualified name of the attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>The value of the attribute as a string</td>
</tr>
</tbody>
</table>

Returns

  bool. Returns true on success, false if the call fails.

8.194.2.2 classID()

static const CClassID & IDOMJobTkGenericNode::classID ( ) [inline], [static]

Retrieves the class id of IDOMJobTkGenericNode.

Returns

  CClassID. Class id of the element.
8.194.2.3  getAttributeAtIndex()

virtual bool IDOMJobTkGenericNode::getAttributeAtIndex (  
    uint32 index,  
    EDLQName & name,  
    EDLSysString & value ) const [pure virtual]

Get an attribute at the given index.

Parameters

<table>
<thead>
<tr>
<th>index</th>
<th>The index of the attribute, with 0 representing the first attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Reference to receive the qualified name of the attribute</td>
</tr>
<tr>
<td>value</td>
<td>Reference to receive the value of the attribute as a string</td>
</tr>
</tbody>
</table>

Returns

bool. Returns true on success, false if the call fails.

8.194.2.4  getNumAttributes()

virtual uint32 IDOMJobTkGenericNode::getNumAttributes () const [pure virtual]

Gets the number of attributes.

Returns

uint32 The number of attributes

8.194.2.5  getQName()

virtual bool IDOMJobTkGenericNode::getQName (  
    EDLQName & qname ) const [pure virtual]

Retrieves the node qname.

Parameters

| qname | Qualified name value |

Returns

bool. Returns true on success, false if the call fails.

The documentation for this class was generated from the following file:
8.195 IDOMJobTkNode Class Reference

Represents a Job Ticket Node.

#include <idomjobtk.h>

Inheritance diagram for IDOMJobTkNode:

```
IRCObject
  |
  |
  IEDLObject
  |
  |
  IDOMNode
  |
  |
  IDOMJobTkNode
```

Classes

- class **Data**
  
  *Initialization data.*

Public Types

- enum **eDOMJobTkNodeType**

  *DOMJobTk node (Property, Feature, Option, InitParam, ScoredProperty, ParamRef)*
Public Member Functions

- virtual bool setQName (const EDLQName &qname)=0
  Sets the qualified name of the node.
- virtual bool getQName (EDLQName &qname) const =0
  Retrieves the node qname.
- virtual bool getQNameAsString (EDLString &strQName) const =0
  Retrieves the full qualified name of the node as a string.
- virtual bool 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.

Static Public Member Functions

- static const CClassID & classID ()
  Retrieves the class id of IDOMJobTk.

Additional Inherited Members

8.195.1 Detailed Description

Represents a Job Ticket Node.

8.195.2 Member Function Documentation

8.195.2.1 classID()

static const CClassID & IDOMJobTkNode::classID () [inline], [static]

Retrieves the class id of IDOMJobTk.

Returns

CClassID. Class id of the element
virtual IDOMJobTkNodePtr IDOMJobTkNode::findChild (eDOMJobTkNodeType nodeType, 
const EDLString & name, 
const EDLSysString & nmspace) [pure virtual]

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

<table>
<thead>
<tr>
<th>nodeType</th>
<th>Job ticket node type.</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of JobTicket node.</td>
</tr>
<tr>
<td>nmspace</td>
<td>Namespace of JobTicket node.</td>
</tr>
</tbody>
</table>

Returns

IDOMJobTkNodePtr. Smart pointer to the found JobTicket node

8.195.2.3 findChild() [2/2]

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

<table>
<thead>
<tr>
<th>nodeType</th>
<th>Job ticket node type.</th>
</tr>
</thead>
<tbody>
<tr>
<td>qname</td>
<td>Qualified name of JobTicket node.</td>
</tr>
</tbody>
</table>

Returns

IDOMJobTkNodePtr. Smart pointer to the found JobTicket node

8.195.2.4 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

8.195.2.5 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
8.195.2.6  getJobTkNodeType()

virtual eDOMJobTkNodeType IDOMJobTkNode::getJobTkNodeType ( ) [pure virtual]

Retrieves the node type.

Returns

eDOMJobTkNodeType. Returns the node type.

8.195.2.7  getQName()

virtual bool IDOMJobTkNode::getQName ( 
    EDLQName & qname ) const [pure virtual]

Retrieves the node qname.

Parameters

    qname  Qualified name value

Returns

    bool. Returns true on success, false if the call fails.

8.195.2.8  getQNameAsString()

virtual bool IDOMJobTkNode::getQNameAsString ( 
    EDLString & strQName ) const [pure virtual]

Retrieves the full qualified name of the node as a string.

Parameters

    strQName  Reference parameter to receive the node's qualified name as a string.

Returns

    bool. Returns true on success, false if the call fails.
8.195.2.9 setJobTkNodeType()

```cpp
template<>
virtual bool IDOMJobTkNode::setJobTkNodeType ( 
    eDOMJobTkNodeType nodeType ) [pure virtual]
```

Sets the node type.

**Parameters**

| `nodeType` | The node type. See eDOMJobTkNodeType. |

**Returns**

bool. Returns true on success, false if the call fails.

8.195.2.10 setQName()

```cpp
template<>
virtual bool IDOMJobTkNode::setQName ( 
    const EDLQName & qname ) [pure virtual]
```

Sets the qualified name of the node.

URI references can contain characters not allowed in names, and are often inconveniently long, so expanded names are not used directly to name elements and attributes in XML documents. Instead qualified names are used. A qualified name is a name subject to namespace interpretation.

**Parameters**

| `qname` | Qualified name value |

**Returns**

bool. Returns true on success, false if the call fails.

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

- `idomjobtk.h`

8.196 IDOMJobTkOwner Class Reference

Interface to the IDOMJobTkOwner node.

```cpp
#include <idomjobtk.h>
```
Public Member Functions

- virtual bool getJobTicket (IDOMJobTkPtr &ptrJobTicket) const =0
  Retrieves the JobTicket held by this node.
- virtual bool setJobTicket (const IDOMJobTkPtr &ptrJobTicket)=0
  Sets the JobTicket for the node.

Static Public Member Functions

- static const CClassID & classID ()
  Retrieves the class id of IDOMJobTkOwner.

Additional Inherited Members

8.196.1 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.
8.196.2  Member Function Documentation

8.196.2.1  classID()  

static const CClassID& IDOMJobTkOwner::classID ( )  [inline], [static]

Retrieves the class id of IDOMJobTkOwner.

Returns  

CClassID. Class id of the element.

8.196.2.2  getJobTicket()  

virtual bool IDOMJobTkOwner::getJobTicket (  
    IDOMJobTkPtr & ptrJobTicket ) const  [pure virtual]

Retrieves the JobTicket held by this node.

Parameters  

| **ptrJobTicket** | Smart pointer receive a reference to the JobTicket. |

Returns  

bool. Returns true on success, false if the call fails.

8.196.2.3  setJobTicket()  

virtual bool IDOMJobTkOwner::setJobTicket (  
    const IDOMJobTkPtr & ptrJobTicket )  [pure virtual]

Sets the JobTicket for the node.

Parameters  

| **ptrJobTicket** | Smart pointer to the new JobTicket. |

Returns  

bool. Returns true on success, false if the call fails.
The documentation for this class was generated from the following file:

- idomjobtk.h

8.197 IDOMJobTkValue Class Reference

Represents a Job Ticket value element.

#include <idomjobtk.h>

Inheritance diagram for IDOMJobTkValue:

![Inheritance Diagram]

Classes

- class Data
  
  *Initialization data.*

Public Member Functions

- virtual bool isValid () const =0
  
  *Returns an indicator of the validity of the data.*
- virtual bool setValue (const PValue &value)=0
  
  *Sets the node value.*
- virtual bool getValue (PValue &value) const =0
  
  *Retrieves the node value.*
Static Public Member Functions

- static const CClassID & classID()

  Retrieves the class id of IDOMJobTk.

Additional Inherited Members

8.197.1 Detailed Description

Represents a Job Ticket value element.

8.197.2 Member Function Documentation

8.197.2.1 classID()

static const CClassID & IDOMJobTkValue::classID() [inline], [static]

Retrieves the class id of IDOMJobTk.

Returns

  CClassID. The class id of the element.

8.197.2.2 getValue()

virtual bool IDOMJobTkValue::getValue ( 
    PValue & value ) const [pure virtual]

Retrieves the node value.

Returns

  bool Returns true on success.

8.197.2.3 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.

8.197.2.4 setValue()

virtual bool IDOMJobTkValue::setValue ( 
    const PValue & value ) [pure virtual]

Sets the node value.
Parameters

- **value** The new node value.

Returns

- *bool* Returns true on success.

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

- idomjobtk.h

### 8.198 IDOMJPEGImage Class Reference

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

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

Inheritance diagram for IDOMJPEGImage:
Static Public Member Functions

- **create** (IEDLClassFactory *pFactory, const IInputStreamPtr &stream)
  
  Create a JPEG Image resource with the given JPEG stream. Throws an IEDLError on failure.

- **encode** (const ISessionPtr &pSession, const IDOMImagePtr &image, const IOutputStreamPtr &stream, uint8 quality=3)
  
  Encode an image as a JPEG stream. This routine may convert the source image into a form that may be encoded as JPEG, such as by stripping alpha channels or converting to a supported colour space. Throws an IEDLError on failure.

- **encode** (const ISessionPtr &pSession, const IImageFramePtr &frame, const IOutputStreamPtr &stream, uint8 quality=3)
  
  Encode the contents of an IImageFrame as a JPEG stream, returning the stream. This routine may convert the source image into a form that may be encoded as JPEG, such as by stripping alpha channels or converting to a supported colour space.

- **classID**()
  
  Retrieves class id of IDOMJPEGImage.

Additional Inherited Members

8.198.1 Detailed Description

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

8.198.2 Member Function Documentation

8.198.2.1 classID()

static const CClassID & IDOMJPEGImage::classID () [inline], [static]

Retrieves class id of IDOMJPEGImage.

Returns

**CClassID** Class id of the element

8.198.2.2 create()

static EDL_API IDOMJPEGImagePtr IDOMJPEGImage::create (IEDLClassFactory *pFactory,

const IInputStreamPtr &stream ) [static]

Create a JPEG Image resource with the given JPEG stream. Throws an IEDLError on failure.
Parameters

<table>
<thead>
<tr>
<th>pFactory</th>
<th>The EDL Class factory to use.</th>
</tr>
</thead>
<tbody>
<tr>
<td>stream</td>
<td>The stream containing the JPEG image.</td>
</tr>
</tbody>
</table>

Returns

IDOMImagePtr The new image.

8.198.2.3 encode() [1/2]

static EDL_API void IDOMJPEGImage::encode (const ISessionPtr & pSession, const IDOMImagePtr & image, const IOutputStreamPtr & stream, uint8 quality = 3) [static]

Encode an image as a JPEG stream. This routine may convert the source image into a form that may be encoded as JPEG, such as by stripping alpha channels or converting to a supported colour space. Throws an IEDLError on failure.

Parameters

<table>
<thead>
<tr>
<th>pSession</th>
<th>The relevant EDL session</th>
</tr>
</thead>
<tbody>
<tr>
<td>image</td>
<td>The image to be encoded</td>
</tr>
<tr>
<td>stream</td>
<td>The stream to use to store the image data.</td>
</tr>
<tr>
<td>quality</td>
<td>The desired quality in the range 1 through 5, with 1 being lowest and 5 being highest. Default is 3.</td>
</tr>
</tbody>
</table>

8.198.2.4 encode() [2/2]

static EDL_API void IDOMJPEGImage::encode (const ISessionPtr & pSession, const IImageFramePtr & frame, const IOutputStreamPtr & stream, uint8 quality = 3) [static]

Encode the contents of an IImageFrame as a JPEG stream, returning the stream. This routine may convert the source image into a form that may be encoded as JPEG, such as by stripping alpha channels or converting to a supported colour space.

Parameters

<table>
<thead>
<tr>
<th>pSession</th>
<th>The relevant EDL session</th>
</tr>
</thead>
<tbody>
<tr>
<td>frame</td>
<td>The frame providing the source image data</td>
</tr>
<tr>
<td>stream</td>
<td>The stream to use to store the image data.</td>
</tr>
<tr>
<td>quality</td>
<td>The desired quality in the range 1 through 5, with 1 being lowest and 5 being highest. Default is 3.</td>
</tr>
</tbody>
</table>
The documentation for this class was generated from the following file:

- `idomimageresource.h`

### 8.199 IDOMLinearGradientBrush Class Reference

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

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

Inheritance diagram for `IDOMLinearGradientBrush`:

![Inheritance Diagram](image)

#### Classes

- **class Data**
  
  *Initialization data.*

Generated by Doxygen
Public Member Functions

- virtual const FPoint & getStartPoint () const =0
  Retrieves the start point of the linear gradient.
- virtual bool 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 bool setEndPoint (const FPoint &pt)=0
  Sets the end point of the linear gradient.
- virtual bool createShading (IEDLClassFactory ∗pFactory, IDOMShadingPatternType2BrushPtr &ptr ← ShadingBrush, bool useFirstStopColorSpace=false)=0
  Create a Type2 Shading Pattern brush from this linear brush.

Static Public Member Functions

- static const CClassID & classID ()
  Retrieves class id of IDOMLinearGradientBrush.

Additional Inherited Members

8.199.1 Detailed Description

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

8.199.2 Member Function Documentation

8.199.2.1 classID()

static const CClassID & IDOMLinearGradientBrush::classID () [inline], [static]

Retrieves class id of IDOMLinearGradientBrush.

Returns

  CClassID. Returns the class id of IDOMLinearGradientBrush.

8.199.2.2 createShading()

virtual bool IDOMLinearGradientBrush::createShading ( 
  IEDLClassFactory * pFactory,
  IDOMShadingPatternType2BrushPtr & ptrShadingBrush,
  bool useFirstStopColorSpace = false ) [pure virtual]

Create a Type2 Shading Pattern brush from this linear brush.

Only Pad repeat mode is supported; a plain Type 2 brush is supported. All alpha information in the gradient stops will be dropped.
### Parameters

<table>
<thead>
<tr>
<th>pFactory</th>
<th>The EDL class factory</th>
</tr>
</thead>
<tbody>
<tr>
<td>ptrShadingBrush</td>
<td>reference to receive the new shading brush.</td>
</tr>
<tr>
<td>useFirstStopColorSpace</td>
<td>If true the colour space in the first stop will be used for the shading. Otherwise either sRGB or scRGB will be used depending on the interpolation mode.</td>
</tr>
</tbody>
</table>

### Returns

bool. Returns true on success, false if the call fails.

#### 8.199.2.3 getEndPoint()

virtual const FPoint& IDOMLinearGradientBrush::getEndPoint ( ) const [pure virtual]

Retrieves the end point of the linear gradient.

#### 8.199.2.4 getStartPoint()

virtual const FPoint& IDOMLinearGradientBrush::getStartPoint ( ) const [pure virtual]

Retrieves the start point of the linear gradient.

#### 8.199.2.5 setEndPoint()

virtual bool IDOMLinearGradientBrush::setEndPoint ( const FPoint & pt ) [pure virtual]

Sets the end point of the linear gradient.

### Parameters

| pt | The new end point |
Returns

bool. Returns true on success, false if the call fails.

8.199.2.6 setStartPoint()

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

Sets the start point of the linear gradient.

Parameters

<table>
<thead>
<tr>
<th>pt</th>
<th>The new start point.</th>
</tr>
</thead>
</table>

Returns

bool. Returns true on success, false if the call fails.

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

- idombrush.h

8.200 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>

Generated by Doxygen
Inheritance diagram for IDOMMaskedBrush:

Classes

- class **Data**
  
  *Initialization data.*

Public Member Functions

- virtual bool **getBrush** (IDOMBrushPtr &brush) const =0
  
  *Retrieves smart pointer to the brush to be painted through the image.*

- virtual bool **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 bool **getEquivalentXPSBrush** (IEDLClassFactory *pFactory, IDOMBrushPtr &xpsForm, const FRect &enclosedBounds)=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.

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=FMMatrix(), float opacity=1.0f)
  Simplified creator for a masked brush. Throws an IEDLError on failure.

- static const CClassID & classID ()
  Retrieves class id of IDOM.

Additional Inherited Members

8.200.1 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.

8.200.2 Member Function Documentation

8.200.2.1 classID()

static const CClassID & IDOMMaskedBrush::classID ( ) [inline], [static]

Retrieves class id of IDOM.

Returns

 CClassID Class id of the element

8.200.2.2 create()

static EDL_API IDOMMaskedBrushPtr IDOMMaskedBrush::create ( IEDLClassFactory * pFactory, 
  const IDOMImagePtr & mask,
  const IDOMBrushPtr & brush,
  const FRect & viewBox,
  const FRect & viewPort,
  const FMMatrix & renderTransform = FMMatrix(),
  float opacity = 1.0f ) [static]

Simplified creator for a masked brush. Throws an IEDLError on failure.
### 8.200.2.3 getBrush()

```cpp
virtual bool IDOMMaskedBrush::getBrush (  
    IDOMBrushPtr & brush ) const [pure virtual]
```

Retrieves smart pointer to the brush to be painted through the image.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>brush</td>
<td>Smart pointer to the brush</td>
</tr>
</tbody>
</table>

**Returns**

`bool` Returns true on success

### 8.200.2.4 getEquivalentXPSBrush()

```cpp
virtual bool IDOMMaskedBrush::getEquivalentXPSBrush (  
    IEDLClassFactory * pFactory,  
    IDOMBrushPtr & xpsForm,  
    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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pFactory</code></td>
<td>A pointer to an EDL class factory</td>
</tr>
<tr>
<td><code>xpsForm</code></td>
<td>A reference to receive the xps compatible form</td>
</tr>
<tr>
<td><code>enclosedBounds</code></td>
<td>The bounds in internal object space of the object that has this brush.</td>
</tr>
</tbody>
</table>
Returns

bool Returns true on success.

8.200.2.5 getIsSoftMask()

virtual bool IDOMMaskedBrush::getIsSoftMask (
    IEDLClassFactory * pFactory ) [pure virtual]

Is the mask a soft mask? That is, is the mask not a single bit image? An outright error will result in an exception of type IEDLError.

Parameters

| pFactory | A pointer to an EDL class factory |

Returns

bool Returns true if the mask is soft.

8.200.2.6 setBrush()

virtual bool IDOMMaskedBrush::setBrush ( 
    const IDOMBrushPtr & brush ) [pure virtual]

Sets brush.

Parameters

| brush | Smart pointer to the brush resource to be painted through the image. |

Returns

bool Returns true on success

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

- idombrush.h

8.201 IDOMMatrix Class Reference

Defines the render transform matrix.
#include <idomresources.h>

Inheritance diagram for IDOMMatrix:

```
    IRCObject
       |
       v
    IEDLObject
       |
       v
    IDOMMatrix
```

### Classes
- **class Data**
  Initialization data.

### Public Member Functions
- **virtual bool getRenderTransform (FMatrix &matrix) const =0**
  Retrieves the render transform matrix.
- **virtual bool setRenderTransform (const FMatrix &matrix)=0**
  Sets the render transform matrix.

### Static Public Member Functions
- **static const CClassID & classID ()**
  Retrieves the class id of IDOMMatrix.

### Additional Inherited Members

#### 8.201.1 Detailed Description
Defines the render transform matrix.

#### 8.201.2 Member Function Documentation
8.201.2.1 classID()

static const CClassID & IDOMMatrix::classID() [inline], [static]

Retrieves the class id of IDOMMatrix.

Returns

CClassID. Returns the class id of the element.

8.201.2.2 getRenderTransform()

virtual bool IDOMMatrix::getRenderTransform (FMatrix & matrix) const [pure virtual]

Retrieves the render transform matrix.

Parameters

| matrix | Render transform matrix |

Returns

bool. Returns true on success, false if the call fails.

8.201.2.3 setRenderTransform()

virtual bool IDOMMatrix::setRenderTransform (const FMatrix & matrix) [pure virtual]

Sets the render transform matrix.

Parameters

| matrix | Render transform matrix |

Returns

bool. Returns true on success, false if the call fails.

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

- idomresources.h

Generated by Doxygen
8.202  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>

8.202.1 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

8.203  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:

![Inheritance diagram](image-url)

Public Types

- enum eType {
  eType, eCoreProperties = 0, eDocumentInfo, eViewerPreferences, ePageView,
  ePDFInfo, eMetadataTypeCnt }

  Metadata types data type.

- enum eXmpContainerType ( eXmpContainer_Alt = 0, eXmpContainer_Bag = 1, eXmpContainer_Seq = 2 )

  The variant values passed to PValue when storing a CEDLStringVect.
8.203 IDOMMetadata Class Reference

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.

Static Public Member Functions

- static EDL_API IDOMMetadataPtr create (IEDLClassFactory *pFactory)
  Create a new IDOMMetadata object.

Additional Inherited Members

8.203.1 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**

- **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**

• **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**

• **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

Generated by Doxygen
• **Version** - `EDLSysString` property in a string form `M.m`, where `M` is the major and `m` is the minor version numbers.

• **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 or PDF/X 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

• **PageMode** - `EDLSysString`.

• **PageLayout** - `EDLSysString`.

#### 8.203.2 Member Enumeration Documentation

1. **eType**

```cpp
class IDOMMetadata
c{enum eType
    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.
};
```

2. **eXmpContainerType**

```cpp
class IDOMMetadata
{enum eXmpContainerType
    // The variant values passed to PValue when storing a CEDLStringVect.
};
```
### Enumerator

<table>
<thead>
<tr>
<th>eXmpContainer_Alt</th>
<th>The PValue variant for storing a CEDLStringVect representing an Xmp Alt container.</th>
</tr>
</thead>
<tbody>
<tr>
<td>eXmpContainer_Bag</td>
<td>The PValue variant for storing a CEDLStringVect representing an Xmp Bag container.</td>
</tr>
<tr>
<td>eXmpContainer_Seq</td>
<td>The PValue variant for storing a CEDLStringVect representing an Xmp Seq container.</td>
</tr>
</tbody>
</table>

#### 8.203.3 Member Function Documentation

##### 8.203.3.1 getProperty()

```cpp
template <eType propertyType>
const PValue & propertyValue
```

Retrieves the value of a named property.

**Parameters**

<table>
<thead>
<tr>
<th>propertyType</th>
<th>The type of the property whose value is to be retrieved.</th>
</tr>
</thead>
<tbody>
<tr>
<td>propertyName</td>
<td>The name of the property whose value is to be retrieved.</td>
</tr>
<tr>
<td>propertyValue</td>
<td>A smart pointer to receive the retrieved property value.</td>
</tr>
</tbody>
</table>

**Returns**

bool. Returns true on success, false if the call fails.

##### 8.203.3.2 getPropertyCollectionEnum()

```cpp
template <eType propertyType>
```

Retrieves a navigable list of the property names in the metadata.

**Returns**

IEDLSysStringCollectionEnumPtr. Returns a smart pointer to the list of property names.

##### 8.203.3.3 removeProperty()

```cpp
template <eType propertyType, const EDLSysString & propertyName>
```

Removes a property, if it exists.
Parameters

<table>
<thead>
<tr>
<th>propertyType</th>
<th>The type of the property to be removed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>propertyName</td>
<td>The name of the property to be removed.</td>
</tr>
</tbody>
</table>

Returns

bool. Returns true on success, false if the call fails.

8.203.4 setProperty()

virtual bool IDOMMetadata::setProperty (  
  eType propertyType,  
  const EDLSysString & propertyName,  
  const PValue & propertyValue ) [pure virtual]

Sets value of a property.

Parameters

<table>
<thead>
<tr>
<th>propertyType</th>
<th>The type of the property whose value is to be set.</th>
</tr>
</thead>
<tbody>
<tr>
<td>propertyName</td>
<td>The name of the property whose value is to be set.</td>
</tr>
<tr>
<td>PropertyValue</td>
<td>The new value to be set for the property.</td>
</tr>
</tbody>
</table>

Returns

bool. Returns true on success, false if the call fails.

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

- idommetadata.h

8.204 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.

#include <idomnode.h>
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 get_property (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 bool set_property (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 (IDOMNodePtr &child) const =0
  Gets the child node which follows the node passed in.
• virtual IDOMNodePtr getPreviousChild (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 bool appendChild (const IDOMNodePtr &child)=0
  Appends a node to the end of the node's child list.
• virtual bool insertChild (const IDOMNodePtr &ptrPreviousSibling, const IDOMNodePtr &child, bool bCheckComplete=true)=0
  Inserts 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 bool 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 bool setParentNode (const IDOMNodePtr &ptrParent)=0
  Sets the parent node.
• virtual bool setPreviousSibling (const IDOMNodePtr &ptrPreviousSibling)=0
  Sets the previous sibling node.
• virtual bool 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 bool getBounds (FRect &bounds, bool applyTransform=true, bool applyClip=true)
  Find the conservative bounding box of the marking content of the node.
• virtual FRect getBounds (bool applyTransform=true, bool applyClip=true)=0
  Find the conservative bounding box of the marking content of the node. An exception of type IEDLError is thrown on failure.
• 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. An exception of type IEDLError will be thrown on failure.
• virtual void completeTree ()=0
  Mark the entire tree from this point as complete. An exception of type IEDLError will be thrown on failure. You should not ordinarily need to call this function.
• virtual void removeCompleteFlagFromTree ()=0
Mark the entire tree from this point as complete. An exception of type \texttt{IEDLError} will be thrown on failure. You should not ordinarily need to call this function.

- virtual void findChildrenOfType (eDOMNodeType type, CEDLVector< IDOMNodePtr > &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:

**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**

8.204.1 Detailed Description

Abstract class providing the interface to basic DOM node functionality. \texttt{IDOMNode} is the base class for many of the other DOM node types, and defines many of the basic functions of DOM nodes.

8.204.2 Member Function Documentation

8.204.2.1 appendChild()

```
virtual bool IDOMNode::appendChild (const IDOMNodePtr &child) [pure virtual]
```

Appends a node to the end of the node's child list.

**Parameters**

- \texttt{child} The child node to append.

**Returns**

- \texttt{bool} True on success, false if the call fails.

8.204.2.2 cloneNode()

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

Generated by Doxygen
Simplified node cloning. An exception of type IEDLError will be thrown on failure.
### 8.204.2.3 cloneTree()

```cpp
virtual IDOMNodePtr IDOMNode::cloneTree (IEDLClassFactory ∗ pFactory) const [pure virtual]
```

Clone the tree of nodes beginning at this node. An exception of type `IEDLError` will be thrown on failure.

**Parameters**

| pFactory | The EDL class factory. |

**Returns**

`IDOMNodePtr` The cloned node.

### 8.204.2.4 cloneTreeAndAppend()

```cpp
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. An exception of type `IEDLError` will be thrown on failure.

**Parameters**

| pFactory | The EDL class factory. |
| dest     | The destination node |

### 8.204.2.5 copyNodeData()

```cpp
virtual bool IDOMNode::copyNodeData (IDOMNode ∗ pSourceNode) [pure virtual]
```

Generated by Doxygen
Copy the properties collection, the flags and the DOM ID from the given source node to this one.
Parameters

*pSourceNode* | Smart pointer to the source node.

Returns

**bool** True on success, false if the call fails.

### 8.204.2.6 effectiveTransformationOfNode()

```cpp
static EDL_API FMatrix IDOMNode::effectiveTransformationOfNode ( const IDOMNodePtr & node ) [static]
```

Attempt to find the effective transformation matrix external to the specified node relative to either a containing page or ultimate parent.

Parameters

*node* | The specified node

Returns

**FMatrix** The effective transformation matrix

### 8.204.2.7 extractChild()

```cpp
virtual IDOMNodePtr IDOMNode::extractChild ( const IDOMNodePtr & child ) [pure virtual]
```

Extracts (that is, finds and removes) a child node from the node children. After extraction the child node is no longer a part of the DOM. If no node is specified, the first available node will be extracted from the node's children.

Parameters

*child* | A pointer to the child node to extract. If set to NULL, the first available child node will be extracted.

Returns

**IDOMNodePtr** A smart pointer to the extracted child node.
### 8.204.2.8 findChildrenOfType()

```cpp
virtual void IDOMNode::findChildrenOfType(
    eDOMNodeType type,
    CEDLVector<IDOMNodePtr> & 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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>type</code></td>
<td>the type of node to find</td>
</tr>
<tr>
<td><code>nodes</code></td>
<td>Vector to receive the found nodes.</td>
</tr>
<tr>
<td><code>searchForms</code></td>
<td>If true, search inside forms referenced from form instance nodes.</td>
</tr>
</tbody>
</table>

### 8.204.2.9 getBounds() [1/2]

```cpp
virtual bool IDOMNode::getBounds (
    FRect & bounds,
    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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bounds</code></td>
<td>A reference to an FRect object to receive the bounding box.</td>
</tr>
<tr>
<td><code>applyTransform</code></td>
<td>Controls whether or not the receiver's RenderTransform is applied to the bounds (if it has one)</td>
</tr>
<tr>
<td></td>
<td>• Pass true (default) to return results in the coordinate space of the object enclosing the receiver</td>
</tr>
<tr>
<td></td>
<td>• Pass false to return results in the coordinate space active inside the object</td>
</tr>
<tr>
<td><code>applyClip</code></td>
<td>Controls whether or not the receiver's Clip is applied to the bounds (if it has a clip).</td>
</tr>
</tbody>
</table>

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**

bool True on success, false if the call fails.
8.204.2.10 getBounds() [2/2]

virtual FRect IDOMNode::getBounds (
    bool applyTransform = true,
    bool applyClip = true ) [pure virtual]

Find the conservative bounding box of the marking content of the node. An exception of type IEDLError is thrown on failure.

For composite objects such as canvases and pages, this function will recurse through its current children.

Parameters

<table>
<thead>
<tr>
<th>applyTransform</th>
<th>Controls whether or not the receiver’s RenderTransform is applied to the bounds (if it has one)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Pass true (default) to return results in the coordinate space of the object enclosing the receiver</td>
</tr>
<tr>
<td></td>
<td>• Pass false to return results in the coordinate space active inside the object</td>
</tr>
</tbody>
</table>

| applyClip | Controls whether or not the receiver’s Clip is applied to the bounds (if it has a clip). |

This parameter only applies to the current node clip path, it does not apply to child nodes. It is generally used to test if the current nodes clip path is effective.

Returns

FRect The bounding box

8.204.2.11 getDOMid()}

virtual DOMid IDOMNode::getDOMid ( ) const [pure virtual]

Retrieves the node ID.

Returns

DOMid The numeric value that uniquely identifies the node.

8.204.2.12 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.
8.204.2.13  getFlags()

virtual IDOMNodeFlags* IDOMNode::getFlags()  [pure virtual]

Retrieves the node's flags property.

Returns
  IDOMNodeFlags A pointer to the node's flags property.

8.204.2.14  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.

8.204.2.15  getNextChild()

virtual IDOMNodePtr IDOMNode::getNextChild (IDOMNodePtr & child) const  [pure virtual]

Gets the child node which follows the node passed in.

Parameters
  child The "current" child node; the node whose next sibling is required.

Returns
  IDOMNodePtr A smart pointer to the next child node. If the child node passed in was the last one under the node, returns a smart pointer to the node's first child.

8.204.2.16  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.
### 8.204.2.17 getNodeType()

```cpp
virtual eDOMNodeType IDOMNode::getNodeType ( ) const [pure virtual]
```

Retrieves the DOM node type.

**Returns**

- **eDOMNodeType** The DOM node type.

### 8.204.2.18 getParentNode()

```cpp
virtual IDOMNodePtr IDOMNode::getParentNode ( ) const [pure virtual]
```

Gets the parent node of this node.

**Returns**

- **IDOMNodePtr** A smart pointer to the parent node.

### 8.204.2.19 getPreviousChild()

```cpp
virtual IDOMNodePtr IDOMNode::getPreviousChild ( 
    IDOMNodePtr & child 
) const [pure virtual]
```

Gets the child node which precedes the node passed in.

**Parameters**

- **child** The "current" child node; the node whose previous sibling is required.

**Returns**

- **IDOMNodePtr** A smart pointer to the previous child node. If the child node passed in was the last one under the node, returns a smart pointer to the node's last child.

### 8.204.2.20 getPreviousSibling()

```cpp
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.
8.204.2.21 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

<table>
<thead>
<tr>
<th>propertyName</th>
<th>The name of the property.</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyValue</td>
<td>Smart pointer to receive the value of the property.</td>
</tr>
</tbody>
</table>

Returns

    bool True on success, false if the call fails.

8.204.2.22 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.

8.204.2.23 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.

8.204.2.24 insertChild()

virtual bool IDOMNode::insertChild {
    const IDOMNodePtr & ptrPreviousSibling,
    const IDOMNodePtr & child,
    bool bCheckComplete = true ) [pure virtual]

Insert a child node after ptrPreviousSibling.
Parameters

<table>
<thead>
<tr>
<th>ptrPreviousSibling</th>
<th>Smart pointer to the previous sibling node. If ptrPreviousSibling is NULL then child will be inserted as the first node</th>
</tr>
</thead>
<tbody>
<tr>
<td>child</td>
<td>The child node to insert.</td>
</tr>
<tr>
<td>bCheckComplete</td>
<td>If false insert node even if parent is complete.</td>
</tr>
</tbody>
</table>

Returns

bool True on success, false if the call fails.

8.204.2.25 isAncestor()

```cpp
virtual bool IDOMNode::isAncestor {
    const IDOMNodePtr & ptrCandidate ) [pure virtual]
```

Function tests whether a candidate node is a descendant of the node.

Parameters

ptrCandidate Smart pointer to the candidate node.

Returns

bool True if ptrCandidate is a descendant of this node.

8.204.2.26 isComplete()

```cpp
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.

8.204.2.27 removeProperty()

```cpp
virtual void IDOMNode::removeProperty {
    const EDLSysString & propertyName ) [pure virtual]
```

Removes property.

The EDL DOM node can store non-content or relationship information through the use of the "properties" feature of the node. The data is represented as key-value pairs; the key being a string and the value being an abstract container called a PValue. PValues can represent integers, strings, DOM nodes, and so on.
Parameters

| propertyName | The name of the property. |

8.204.2.28 replaceChild()

```cpp
virtual bool IDOMNode::replaceChild (  
    const IDOMNodePtr & oldChild,  
    const IDOMNodePtr & newChild ) [pure virtual]
```

Replaces the child node with another.

Parameters

| oldChild | Pointer to the child node to be replaced. |
| newChild | Pointer to the replacing node. |

Returns

 bool True on success, false if the call fails.

8.204.2.29 setDOMid()

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

Sets the node ID.

Parameters

| id | The new DOM ID for the node. |

8.204.2.30 setNextSibling()

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

Sets the next sibling node.

Parameters

| ptrNextSibling | Smart pointer to the new next sibling node. |
Returns

bool True on success, false if the call fails.

8.204.2.31 setParentNode()

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

Sets the parent node.

Parameters

ptrParent Smart pointer to the new parent node.

Returns

bool True on success, false if the call fails.

8.204.2.32 setPreviousSibling()

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

Sets the previous sibling node.

Parameters

ptrPreviousSibling Smart pointer to the new previous sibling node.

Returns

bool True on success, false if the call fails.

8.204.2.33 setProperty()

virtual bool 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.
### 8.205 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.

```cpp
#include <idomnode.h>
```

---

### Parameters

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>propertyName</td>
<td>The name of the property.</td>
</tr>
<tr>
<td>propertyValue</td>
<td>The new value for the property.</td>
</tr>
</tbody>
</table>

### Returns

**bool** Returns true on success, false if the call fails.

---

### 8.204.34 walkTree()

```cpp
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 node's 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>func</td>
<td>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.</td>
</tr>
<tr>
<td>priv</td>
<td>A bare pointer that is passed to func for each call.</td>
</tr>
<tr>
<td>descendIntoBrushes</td>
<td>If true, the walker will descend into composite brushes, only once per invocation.</td>
</tr>
<tr>
<td>descendIntoForms</td>
<td>If true, the walker will descend into forms, only once per invocation.</td>
</tr>
</tbody>
</table>

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

- idomnode.h
Public Types

- enum DOMNodeFlags { , eNodeDirtyFlag = 10, eNodeInterestingFlag = 11 }
  
  DOM node flag enumeration First eight bits (0..7) are reserved for EDL private use - OEM user cannot change these bits. next eight bits (8..15) are reserved for EDL public use - OEM user can change these bits the bits meaning is "published" for OEM, for example bit 8 is the eNodeRenderFlag next sixteen bits (16..32) are reserved for OEM usage.

Public Member Functions

- 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.

8.205.1 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 published. The next sixteen bits (16..32) are reserved for OEM usage.

8.205.2 Member Enumeration Documentation

8.205.2.1 DOMNodeFlags

enum IDOMNodeFlags::DOMNodeFlags

DOM node flag enumeration First eight bits (0..7) are reserved for EDL private use - OEM user cannot change these bits. next eight bits (8..15) are reserved for EDL public use - OEM user can change these bits the bits meaning is "published" for OEM, for example bit 8 is the eNodeRenderFlag next sixteen bits (16..32) are reserved for OEM usage.

Enumerator

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eNodeDirtyFlag</td>
<td>eNodeDirtyFlag marks a node as dirty (i.e with changed content)</td>
</tr>
<tr>
<td>eNodeInterestingFlag</td>
<td>eNodeInterestingFlag marks a node as interesting for rendering purposes. See IJawsRenderer for details.</td>
</tr>
</tbody>
</table>

8.205.3 Member Function Documentation
8.205.3.1  get()

virtual bool IDOMNodeFlags::get (  
    uint32 position ) const [pure virtual]

Tests whether the bit at a specified position is set to 1.

Parameters

| position | The bit position to test. |

Returns

bool Returns true if the flag bit is set to 1, false if it is set to zero.

8.205.3.2  set()

virtual void IDOMNodeFlags::set (  
    uint32 position,  
    bool value = true ) [pure virtual]

Sets the bit at the specified position to the specified value.

Parameters

| position | The bit position to set. |
| value    | The bit value to set. |

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

- idomnode.h

8.206  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:

![Inheritance Diagram](image)

### 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

#### 8.206.1 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.

#### 8.206.2 Member Function Documentation
8.206.1  classID()

static const CClassID & IDOMNullBrush::classID ( ) [inline], [static]

Retrieves class id of IDOMNullBrush.

Returns

   CClassID. Class id of the element

8.206.2  create()

static EDL_API IDOMNullBrushPtr IDOMNullBrush::create ( IEDLClassFactory ∗ pFactory ) [static]

Simplified creator for a Null brush. Throws an IEDLError on failure.

Parameters

   pFactory    The factory to use.

Returns

   IDOMNullBrushPtr The new brush.

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

   • idombrush.h

8.207   IDOMOPI Class Reference

Base class representing OPI proxy. Has two descendant interfaces IDOMOPI13 and IDOMOPI20.

#include <idomopi.h>
Public Member Functions

- virtual void getVersion (int32 &major, int32 &minor) const =0
  
  Retrieves the version of the OPI supported by the interface in the form: major.minor.

Additional Inherited Members

8.207.1 Detailed Description

Base class representing OPI proxy. Has two descendant interfaces IDOMOPI13 and IDOMOPI20.

8.207.2 Member Function Documentation

8.207.2.1 getVersion()

virtual void IDOMOPI::getVersion (  
    int32 & major,  
    int32 & minor ) const [pure virtual]

Retrieves the version of the OPI supported by the interface in the form: major.minor.
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>major</td>
<td>integer representing the major version of OPI.</td>
</tr>
<tr>
<td>minor</td>
<td>integer representing the minor version of OPI.</td>
</tr>
</tbody>
</table>

Implemented in IDOMOPI20, and IDOMOPI13.

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

- idomopi.h

### 8.208 IDOMOPI13 Class Reference

The interface representing OPI proxy with accordance to Open Prepress Interface Specification 1.3.

```cpp
#include "idomopi.h"
```

Inheritance diagram for IDOMOPI13:

```
IRCOBJECT

IEDLOBJECT

IDOMOPI

IDOMOPI13
```

#### Public Member Functions

- virtual void `getVersion` (int32 &major, int32 &minor) const
  
  Retrieves the version of the OPI supported by the interface in the form: major.minor.

- virtual bool `getF` (IFileSpecPtr &ptrF) const =0
  
  Retrieves the external file containing the image.

- virtual bool `getID` (EDLSysString &strID) const =0
Retrieves the ID string.
• virtual bool getComments (EDLSysString &strComments) const =0
  Retrieves the Comments in human readable form.
• virtual bool getSize (IntPoint &size) const =0
  Retrieves the dimensions of the image in pixels.
• virtual bool getCropRect (IntRect &cropRect) const =0
  Retrieves the CropRect (portion of the image to be used)
• virtual bool getCropFixed (FRect &cropFixed) const =0
  Retrieves the CropFixed (same as CropRect but in real numbers)
• virtual bool getPosition (double position[]) const =0
  Retrieves the position on the page of the cropped image.
• virtual bool getResolution (FPoint &resolution) const =0
  Retrieves the resolution of the image in samples per inch.
• virtual eOPIColorType getType () const =0
  Retrieves the color type of the image.
• virtual bool getColorValue (double value[]) const =0
  Retrieves the values of the image color.
• virtual bool getColorName (EDLSysString &colorName) const =0
  Retrieves the name of the image color.
• virtual double getTint () const =0
  Retrieves the concentration of the color.
• virtual bool getOverPrint () const =0
  Retrieves the flag whether the image is to overprint or knock out underlying marks on other separations.
• virtual bool getImageType (IntPoint &imageType) const =0
  Retrieves the image type as a set of two numbers, first specifying the number of samples per pixel and second specifying the number of bits per sample.
• virtual bool getTransparency () const =0
  Retrieves the flag specifying whether white pixels in the image are to be treated as transparent.
• virtual int32 getGrayMapSize () const =0
  Retrieves the size of the GrayMap.
• virtual bool getGrayMap (int32 grayMap[]) const =0
  Retrieves the GrayMap as the array of integers in the range 0 to 65535.
• virtual bool setGrayMap (int32 size, int32 grayMap[]) =0
  Sets the GrayMap as the array of integers in the range 0 to 65535.
• virtual int32 getTagsSize () const =0
  Retrieves the size of the Tags array.
• virtual bool getTags (int32 tags[]) const =0
  Retrieves the set of Tags (as the array of integers) supported by the OPI proxy.
• virtual bool getTagText (int32 tag, EDLSysString &tagText) const =0
  Retrieves the Tag text for a given tag number.
• virtual bool addTag (int32 tag, const EDLSysString &tagText)=0
  Adds the new Tag to the set of Tags supported by the OPI proxy.
• virtual bool deleteTag (int32 tag)=0
  Deletes the Tag from the set of Tags supported by the OPI proxy.

Additional Inherited Members

8.208.1 Detailed Description

The interface representing OPI proxy with accordance to Open Prepress Interface Specification 1.3.
8.208.2 Member Function Documentation

8.208.2.1 addTag()

virtual bool IDOMOPI13::addTag (  
    int32 tag,    
    const EDLSysString & tagText  ) [pure virtual]

Adds the new Tag to the set of Tags supported by the OPI proxy.

Parameters

<table>
<thead>
<tr>
<th>tag</th>
<th>tag number to add</th>
</tr>
</thead>
<tbody>
<tr>
<td>tagText</td>
<td>tag text corresponding to the tag number to add</td>
</tr>
</tbody>
</table>

Returns

bool Returns true on success

8.208.2.2 deleteTag()

virtual bool IDOMOPI13::deleteTag (  
    int32 tag  ) [pure virtual]

Deletes the Tag from the set of Tags supported by the OPI proxy.

Parameters

| tag    | tag number to delete |

Returns

bool Returns true on success

8.208.2.3 getColorName()

virtual bool IDOMOPI13::getColorName (  
    EDLSysString & colorName  ) const [pure virtual]

Retrieves the name of the image color.

corresponds to the color name in ALDImageColor in OPI1.3 specs
Parameters

| colorName   | image color name |

Returns

bool Returns true on success

8.208.2.4 getColorType()

virtual eOPIColorType IDOMOPI13::getColorType() const [pure virtual]

Retrieves the color type of the image.

corresponds to ALDImageColorType in OPI1.3 specs

Returns

eOPIColorType Returns color type of the image

8.208.2.5 getColorValue()

virtual bool IDOMOPI13::getColorValue(
    double value[] ) const [pure virtual]

Retrieves the values of the image color.

corresponds to the color values in ALDImageColor in OPI1.3 specs

Parameters

| value     | an array of four numbers in the form [C M Y K] |

Returns

bool Returns true on success

8.208.2.6 getComments()

virtual bool IDOMOPI13::getComments(
    EDLSysString & strComments ) const [pure virtual]

Retrieves the Comments in human readable form.

corresponds to ALDObjectComments in OPI1.3 specs
Parameters

| strComments | Comments string |

Returns

bool Returns true on success

8.208.2.7 getCropFixed()

virtual bool IDOMOPI13::getCropFixed (  
    FRect & cropFixed ) const  [pure virtual]

Retrieves the CropFixed (same as CropRect but in real numbers)

corresponds to ALDImageCropFixed in OPI1.3 specs

Parameters

| cropFixed | CropFixed of the image |

Returns

bool Returns true on success

8.208.2.8 getCropRect()

virtual bool IDOMOPI13::getCropRect (  
    IntRect & cropRect ) const  [pure virtual]

Retrieves the CropRect (portion of the image to be used)

corresponds to ALDImageCropRect in OPI1.3 specs

Parameters

| cropRect | CropRect of the image |

Returns

bool Returns true on success
8.208.2.9  getF()

virtual bool IDOMOPI13::getF ( 
    IFileSpecPtr & ptrF ) const  [pure virtual]

Retrieves the external file containing the image.
corresponds to ALDImageFilename in OPI1.3 specs

Parameters

| ptrF | Smart pointer to file specification |

Returns

bool Returns true on success

---

8.208.2.10  getGrayMap()

virtual bool IDOMOPI13::getGrayMap ( 
    int32 grayMap[ ] ) const  [pure virtual]

Retrieves the GrayMap as the array of integers in the range 0 to 65535.
corresponds to ALDImageGrayMap in OPI1.3 specs

Parameters

| grayMap | array of integers to hold the GrayMap values |

Returns

bool Returns true on success

---

8.208.2.11  getGrayMapSize()

virtual int32 IDOMOPI13::getGrayMapSize ( ) const  [pure virtual]

Retrieves the size of the GrayMap.

Returns

int32 Returns size of the GrayMap
8.208.2.12  getID()

virtual bool IDOMOPI13::getID ( 
        EDLSysString & strID ) const [pure virtual]

Retrieves the ID string.

corresponds to ALDImageID in OPI1.3 specs

Parameters

<table>
<thead>
<tr>
<th>strID</th>
<th>ID string</th>
</tr>
</thead>
</table>

Returns

bool Returns true on success

8.208.2.13  getImageType()

virtual bool IDOMOPI13::getImageType ( 
        IntPoint & imageType ) const [pure virtual]

Retrieves the image type as a set of two numbers, first specifying the number of samples per pixel and second specifying the number of bits per sample.

corresponds to ALDImageType in OPI1.3 specs

Parameters

<table>
<thead>
<tr>
<th>imageType</th>
<th>image type. x represents number of samples, y - number of bits</th>
</tr>
</thead>
</table>

Returns

bool Returns true on success

8.208.2.14  getOverPrint()

virtual bool IDOMOPI13::getOverPrint ( ) const [pure virtual]

Retrieves the flag whether the image is to overprint or knock out underlying marks on other separations.

corresponds to ALDImageOverprint in OPI1.3 specs

Returns

bool Returns true if overprint (default: false)
virtual bool IDOMOPI13::getPosition ( double position[] ) const [pure virtual]

Retrieves the position on the page of the cropped image.

corresponds to ALDImagePosition in OPI1.3 specs Position is an array of 8 doubles in the form [ llx lly ulx uly urx ury lrx lry ]

Parameters

- **position** array sufficient to hold 8 doubles

Returns

- bool Returns true on success


virtual bool IDOMOPI13::getResolution ( FPoint & resolution ) const [pure virtual]

Retrieves the resolution of the image in samples per inch.

corresponds to ALDImageResolution in OPI1.3 specs

Parameters

- **resolution** image resolution. x - horizontal and y - vertical resolution

Returns

- bool Returns true on success


virtual bool IDOMOPI13::getSize ( IntPoint & size ) const [pure virtual]

Retrieves the dimensions of the image in pixels.

corresponds to ALDImageDimensions in OPI1.3 specs, x represents width and y height
Parameters

| size | image dimensions |

Returns

bool Returns true on success

8.208.2.18 getTags()

```cpp
template bool IDOMOPI13::getTags (  
    int32 tags[] ) const [pure virtual]
```

Retrieves the set of Tags (as the array of integers) supported by the OPI proxy.

corresponds to ALDImageAsciiTag in OPI1.3 specs

Parameters

| tags array of integers to hold the values of the Tags supported |

Returns

bool Returns true on success

8.208.2.19 getTagsSize()

```cpp
template int32 IDOMOPI13::getTagsSize ( ) const [pure virtual]
```

Retrieves the size of the Tags array.

Returns

int32 Returns size of the Tags array

8.208.2.20 getTagText()

```cpp
template bool IDOMOPI13::getTagText (  
    int32 tag,  
    EDLSysString & tagText ) const [pure virtual]
```

Retrieves the Tag text for a given tag number.

corresponds to ALDImageAsciiTag in OPI1.3 specs
Parameters

<table>
<thead>
<tr>
<th>tag</th>
<th>tag number for which the text is to be retrieved</th>
</tr>
</thead>
<tbody>
<tr>
<td>tagText</td>
<td>tag text retrieved</td>
</tr>
</tbody>
</table>
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>major</td>
<td>integer representing the major version of OPI.</td>
</tr>
<tr>
<td>minor</td>
<td>integer representing the minor version of OPI.</td>
</tr>
</tbody>
</table>

Implements IDOMOPI.

8.208.2.24 setGrayMap()

```cpp
virtual bool IDOMOPI13::setGrayMap ( 
    int32 size, 
    int32 grayMap[] ) [pure virtual]
```

Sets the GrayMap as the array of integers in the range 0 to 65535.

corresponds to ALDImageGrayMap in OPI1.3 specs

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>size</td>
<td>the new size of the GrayMap. Size 0 removes the GrayMap</td>
</tr>
<tr>
<td>grayMap</td>
<td>array of integers holding the GrayMap values</td>
</tr>
</tbody>
</table>

Returns

bool Returns true on success

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

- idomopi.h

8.209 IDOMOPI20 Class Reference

The interface representing OPI proxy with accordance to Open Prepress Interface Specification 2.0.

```cpp
#include <idomopi.h>
```
Inheritance diagram for IDOMOPI20:

![Inheritance Diagram]

**Public Member Functions**

- virtual void **getVersion** (int32 &major, int32 &minor) const  
  Retrieves the version of the OPI supported by the interface in the form: major.minor.

**Additional Inherited Members**

8.209.1 Detailed Description

The interface representing OPI proxy with accordance to Open Prepress Interface Specification 2.0.

8.209.2 Member Function Documentation

8.209.2.1 getVersion()

```cpp
virtual void IDOMOPI20::getVersion {
    int32 & major,
    int32 & minor) const [inline], [virtual]
```

Retrieves the version of the OPI supported by the interface in the form: major.minor.
Parameters

| major | integer representing the major version of OPI. |
| minor | integer representing the minor version of OPI. |

Implements IDOMOPI.

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

- idomopi.h

8.210 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:

```
IRCOBJECT

IEDLObject

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.
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

8.210.1 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.

8.210.2 Member Function Documentation

8.210.2.1 create()

static EDL_API IDOMOutlinePtr IDOMOutline::create ( 
  IEDLClassFactory ∗pFactory, 
  const EDLString &language = EDLString(L"und") ) [static]

Simplified creator to create an outline.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pFactory</td>
<td>The EDL factory to use</td>
</tr>
<tr>
<td>language</td>
<td>The language code to use (see setLanguage below)</td>
</tr>
</tbody>
</table>

8.210.2.2 getLanguage()

virtual bool IDOMOutline::getLanguage ( 
  EDLString & strLanguage ) const [pure virtual]

Retrieves the default language of the outline node.

English is defined as en_GB and American English as en_US. There is no default setting. If the language is not known it is set to "und" (undetermined). For further information see http://www.w3.org/International/articles/language-tags/. The language is specified according to RFC 3066.
Parameters

| strLanguage | Reference parameter to receive the default language |

Returns

bool. Returns true on success, false if the call fails.

8.210.2.3 getOutlineTree()

virtual IDOMOutlineTreePtr IDOMOutline::getOutlineTree () [pure virtual]

Retrieves the outline tree.

Returns

IDOMOutlineTreePtr. Returns a smart pointer to the outline tree.

8.210.2.4 setLanguage()

virtual bool IDOMOutline::setLanguage {
    const EDLString & strLanguage } [pure virtual]

the default language of the outline node.

English is defined as en_GB and American English as en_US. There is no default setting. If the language is not known it is set to "und" (undetermined). For further information see http://www.w3.org/International/articles/language-tags/. The language is specified according to RFC 3066.

Parameters

| strLanguage | The new default language |

Returns

bool. Returns true on success, false if the call fails.

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

- idomoutline.h
8.211  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:

```
IRCOObject

IEDLObject

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.

Static Public Member Functions

static EDL_API IDOMOutlineTreeNodePtr createNode (IEDLClassFactory ∗pFactory, const EDLString &description, bool expanded=true, const IDOMTargetPtr &target=IDOMTargetPtr(), const IDOMColorPtr &textColor=IDOMColorPtr(), eTextStyle style=eTextStyleNone, EDLString language=EDLString(L"und"))
  
  Simplified creator to create an outline tree entry with a new outline tree node.

Additional Inherited Members

8.211.1 Detailed Description

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

@ see 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.

8.211.2 Member Enumeration Documentation

8.211.2.1 eTextStyle

enum IDOMOutlineEntry::eTextStyle

Specifies an outline text style.
## Enumerator

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eTextStyleNone</td>
<td>Default text style.</td>
</tr>
<tr>
<td>eTextStyleItalic</td>
<td>Italic text style.</td>
</tr>
<tr>
<td>eTextStyleBold</td>
<td>Bold text style.</td>
</tr>
<tr>
<td>eTextStyleBoldItalic</td>
<td>Bold and italic text style.</td>
</tr>
</tbody>
</table>

### 8.211.3 Member Function Documentation

#### 8.211.3.1 createNode()

```cpp
static EDL_API IDOMOutlineTreeNodePtr IDOMOutlineEntry::createNode ( 
    IEDLClassFactory ∗ pFactory, 
    const EDLString & description, 
    bool expanded = true, 
    const IDOMTargetPtr & target = IDOMTargetPtr(), 
    const IDOMColorPtr & textColor = IDOMColorPtr(), 
    eTextStyle style = eTextStyleNone, 
    EDLString language = EDLString(L"und") ) [static]
```

Simplified creator to create an outline tree entry with a new outline tree node.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pFactory</code></td>
<td>The EDL factory to use</td>
</tr>
<tr>
<td><code>description</code></td>
<td>The description of the outline entry</td>
</tr>
<tr>
<td><code>expanded</code></td>
<td>If the outline entry's children should be shown by default</td>
</tr>
<tr>
<td><code>target</code></td>
<td>The destination for the outline entry, or NULL for no target.</td>
</tr>
<tr>
<td><code>textColor</code></td>
<td>The color for the outline entry, or NULL for default. If provided, it must use the DeviceRGB color space.</td>
</tr>
<tr>
<td><code>textStyle</code></td>
<td>The desired text style</td>
</tr>
<tr>
<td><code>language</code></td>
<td>The language code to use (see setLanguage below)</td>
</tr>
</tbody>
</table>

#### 8.211.3.2 getDescription()

```cpp
virtual bool IDOMOutlineEntry::getDescription ( 
    EDLString & strDescription ) const [pure virtual]
```

Retrieves the description of the outline entry node.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>strDescription</code></td>
<td>Reference parameter to receive the outline entry node description.</td>
</tr>
</tbody>
</table>
8.211.3.3  getExpanded()

virtual bool IDOMOutlineEntry::getExpanded ( ) const [pure virtual]

Retrieves "expanded" flag value.

Returns

bool. Returns true if the outline entry is expanded, false otherwise.

8.211.3.4  getLanguage()

virtual bool IDOMOutlineEntry::getLanguage ( 
    EDLString & strLanguage ) const [pure virtual]

Retrieves the default language of the outline entry node.

English is defined as en_GB and American English as en_US. There is no default setting. If the language
is not known it is set to "und" (undetermined). For further information see http://www.w3.org/International/articles/language-tags/. The language is specified according to RFC 3066.

Parameters

| strLanguage | Default language |

Returns

bool. Returns true on success, false if the call fails.

8.211.3.5  getStructureElement()

virtual bool IDOMOutlineEntry::getStructureElement ( 
    IEDLObjectPtr & ptrSE ) const [pure virtual]

Retrieves the structure element.

Parameters

| ptrSE | Smart pointer to receive the structure element |

Generated by Doxygen
Returns

bool. Returns true on success, false if the call fails.

8.211.3.6  getTarget()

virtual bool IDOMOutlineEntry::getTarget (  
   IDOMTargetPtr & ptrTarget ) const  [pure virtual]

Retrieves the target of the outline entry node. If outline entry doesn't have target then ptrTarget will be set to NULL.

Parameters

| ptrTarget | Smart pointer to receive the target of the outline entry node. |

Returns

bool. Returns true on success, false if the call fails.

8.211.3.7  getTextColor()

virtual bool IDOMOutlineEntry::getTextColor (  
   IDOMColorPtr & ptrColor ) const  [pure virtual]

Retrieves the color to be used for the outline entry's text.

Parameters

| ptrColor | Smart pointer to the outline entry text's color. |

Returns

bool. Returns true on success, false if the call fails.

8.211.3.8  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.

8.211.3.9 setDescription()

virtual bool IDOMOutlineEntry::setDescription ( const EDLString & strDescription ) [pure virtual]

Sets the description of the outline entry node.

Parameters

- strDescription: The new description for the outline entry node.

Returns

- bool. Returns true on success, false if the call fails.

8.211.3.10 setExpanded()

virtual void IDOMOutlineEntry::setExpanded ( bool expanded ) [pure virtual]

Sets the outline entry's "expanded" flag.

Parameters

- expanded: New value for the outline entry's "expanded" flag.

8.211.3.11 setLanguage()

virtual bool IDOMOutlineEntry::setLanguage ( const EDLString & strLanguage ) [pure virtual]

Retrieves the default language of the outline entry node.

English is defined as en_GB and American English as en_US. There is no default setting. If the language is not known it is set to "und" (undetermined). For further information see http://www.w3.org/International/articles/language-tags/. The language is specified according to RFC 3066.
Parameters

| strLanguage | Default language |

Returns

bool. Returns true on success, false if the call fails.

8.211.3.12 setStructureElement()

virtual bool IDOMOutlineEntry::setStructureElement ( 
    const IEDLObjectPtr & ptrSE ) [pure virtual]

Sets the structure element.

Parameters

| ptrSE | Smart pointer to the new structure element. |

Returns

bool. Returns true on success, false if the call fails.

8.211.3.13 setTarget()

virtual bool IDOMOutlineEntry::setTarget ( 
    const IDOMTargetPtr & ptrTarget ) [pure virtual]

Sets the target of the outline entry node. NULL is a valid value of ptrTarget parameter.

Parameters

| ptrTarget | Smart pointer to the new target of the outline entry node. |

Returns

bool. Returns true on success, false if the call fails.

8.211.3.14 setTextColor()

virtual bool IDOMOutlineEntry::setTextColor ( 
    const IDOMColorPtr & ptrColor ) [pure virtual]
Sets the color to be used for the outline entry's text.
Parameters

| ptrColor | Smart pointer to the new outline entry text color. |

Returns

bool. Returns true on success, false if the call fails.

8.211.3.15  setTextStyle()

virtual void IDOMOutlineEntry::setTextStyle ( eTextStyle style ) [pure virtual]

Sets the outline entry's text style.

See also

eTextStyle

Parameters

| style | The new value for the outline entry's text style. |

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

- idomoutline.h

8.212  IDOMPage Class Reference

The base class for DOM page classes such as IDOMFixedPage.

#include <idompage.h>
Inheritance diagram for IDOMPage:

```
IRCOBJECT

IEDLObject

IDOMNode

IDOMPage
```

**Static Public Member Functions**

- static const CClassID & classID ()
  
  Retrieves the class id of IDOMPage.

**Additional Inherited Members**

**8.212.1 Detailed Description**

The base class for DOM page classes such as IDOMFixedPage.

**8.212.2 Member Function Documentation**

**8.212.2.1 classID()**

static const CClassID & IDOMPage::classID () [inline], [static]

Retrieves the class id of IDOMPage.

Returns

**CClassID** The class id of the element

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

- idompage.h

Generated by Doxygen
**IDOMPageRectTarget** nodes are used to describe hyperlinks on a page rectangle to targets on the same page.

```cpp
#include <idomtarget.h>
```

Inheritance diagram for IDOMPageRectTarget:

```
IRCOObject

IEDLObject

IDOMTarget

IDOMPageRectTarget
```

**Public Types**

- enum **eFitType**
  
  Destination fit types enumeration.

**Public Member Functions**

- virtual **DOMid** **getPageId** () const =0
  
  Retrieves the target page DOM id.
- virtual void **setPageId** (DOMid id)=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
8.213 IDOMPageRectTarget Class Reference

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.

Static Public Member Functions

• static EDL_API IDOMPageRectTargetPtr create (IEDLClassFactory ∗pFactory, DOMid pageId, eFitType fitType=eFTFit, double zoom=0.0, const doubleWithNull &left=doubleWithNull(), const doubleWithNull &top=doubleWithNull(), const doubleWithNull &right=doubleWithNull(), const doubleWithNull &bottom=doubleWithNull())
  Simplified creator for a page rect target. Throws an exception of type IEDLError on failure.

Additional Inherited Members

8.213.1 Detailed Description

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

8.213.2 Member Function Documentation

8.213.2.1 create()

static EDL_API IDOMPageRectTargetPtr IDOMPageRectTarget::create (IEDLClassFactory ∗pFactory, DOMid pageId, eFitType fitType=eFTFit, double zoom = 0.0, const doubleWithNull &left=doubleWithNull(), const doubleWithNull &top=doubleWithNull(), const doubleWithNull &right=doubleWithNull(), const doubleWithNull &bottom=doubleWithNull()) [static]

Simplified creator for a page rect target. Throws an exception of type IEDLError on failure.
Parameters

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

Returns

IDOMPageRectTarget The new target

8.213.2.2 `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.

8.213.2.3 `getFitType()`

virtual eFitType IDOMPageRectTarget::getFitType ( ) const [pure virtual]

Gets the fit type.

See also

eFitType

Returns

eFitType. Returns the fit type.
8.213.2.4 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.

8.213.2.5 getPageId()

virtual DOMid IDOMPageRectTarget::getPageId ( ) const [pure virtual]

Retrieves the target page DOM id.

Returns

DOMid. The target page id

8.213.2.6 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.

8.213.2.7 getTargetType()

virtual eTargetType IDOMPageRectTarget::getTargetType ( ) const [inline], [virtual]

Implementation of getTargetType for IDOMPageTarget.

Returns

eTargetType. Returns "ePage".

Implements IDOMTarget.
8.213.2.8 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.

8.213.2.9 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.

8.213.2.10 setBottom()

virtual void IDOMPageRectTarget::setBottom (  
    const doubleWithNull & bottom )  [pure virtual]

Sets the bottom coordinate value of the target.

Parameters

| left   | The new bottom coordinate value. |

8.213.2.11 setFitType()

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

Sets the fit type.

See also

- eFitType
Returns

eFitType. The new fit type.

### 8.213.2.12 setLeft()

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

Sets the left coordinate value of the target.

**Parameters**

| left | The new left coordinate value |

### 8.213.2.13 setPageId()

```cpp
virtual void IDOMPageRectTarget::setPageId (
    DOMId id ) [pure virtual]
```

Sets the target page DOM id.

**Parameters**

| id. | The new target page id |

### 8.213.2.14 setRight()

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

Sets the right coordinate value of the target.

**Parameters**

| left | The new right coordinate value |

### 8.213.2.15 setTop()

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

Sets the top coordinate value of the target.

Parameters

| left | The new top coordinate value. |

8.213.2.16 setZoom()

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

Sets for the zoom value.

Parameters

| zoom | The new zoom value. Zero zoom value specifies that the current value of that parameter is to be retained unchanged. |

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

- idomtarget.h

8.214 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:

```
IRCOBJECT

|IEDLObject
|IDOMTarget

|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.

**Additional Inherited Members**

**8.214.1 Detailed Description**

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

**8.214.2 Member Function Documentation**
8.214.2.1 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.

8.214.2.2 getTargetType()

virtual eTargetType IDOMPageTarget::getTargetType ( ) const [inline], [virtual]

Implementation of getTargetType for IDOMPageTarget.

Returns

eTargetType. Returns "ePage".

Implements IDOMTarget.

8.214.2.3 setTargetPage()

virtual void IDOMPageTarget::setTargetPage ( 
    uint32 page ) [pure virtual]

Sets the target page number for a page target. A page target has the whole page as the target. It refers to a page using the absolute page number within the document sequence, where the very first page in the first document is number 1.

Parameters

| page | The new target page number. |

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

- idomtarget.h

8.215 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 PathFigures.
#include <idompathgeometry.h>

Inheritance diagram for IDOMPathFigure:

```
IRCOBJECT

IEDOBJECT

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 bool 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.

- virtual const FPoint & getStartPoint () const =0
  
  Retrieves the start point for the first segment in the figure.

- virtual bool setStartPoint (const FPoint &sp)=0
  
  Sets the start point for the first segment in the figure.

- 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 bool 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.

- virtual IDOMSegmentCollectionEnumPtr getSegmentCollectionEnum ()=0
  
  Retrieves a navigable list of the path segments in the figure.

- virtual uint32 getSegmentsCount ()=0
  
  Retrieves the number of path segments in the figure.
• virtual void clearSegmentCollection ()=0
  Removes all path segments from the figure.
• virtual bool addSegment (const IDOMPathSegmentPtr &ptrSegment)=0
  Append a path segment to the figure.
• virtual bool getBounds (FRect &bounds)=0
  Finds the conservative bounding box of the figure.

Static Public Member Functions

• static const CClassID & classID ()
  Retrieves the class id of IDOMPathFigure.

Additional Inherited Members

8.215.1 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 PathFigures.

See also
  IDOMPathNode
  IDOMPathSegment
  IDOMPathGeometry

8.215.2 Member Function Documentation

8.215.2.1 addSegment()

virtual bool IDOMPathFigure::addSegment {
    const IDOMPathSegmentPtr & ptrSegment ) [pure virtual]

Append a path segment to the figure.

Parameters

pathSegment  Smart pointer to the path segment to add.

Returns

  bool Returns true on success, false if the call fails.
8.215.2.2 classID()

static const CClassID& IDOMPathFigure::classID() [inline], [static]

Retrieves the class id of IDOMPathFigure.

Returns

CClassID. Returns the class id of the element.

8.215.2.3 getBounds()

virtual bool IDOMPathFigure::getBounds (
        FRect & bounds) [pure virtual]

Finds the conservative bounding box of the figure.

Parameters

| bounds | Reference variable to receive the bounding box. |

Returns

bool Returns true on success, false if the call fails.

8.215.2.4 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. Returns true if the path is closed.
8.215.2.5 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.

8.215.2.6 getSegmentCollectionEnum()

virtual IDOMSegmentCollectionEnumPtr IDOMPathFigure::getSegmentCollectionEnum ( ) [pure virtual]

Retrieves a navigable list of the path segments in the figure.

Returns

IDOMSegmentCollectionEnumPtr. Returns a smart pointer to the navigable list.

8.215.2.7 getSegmentsCount()

virtual uint32 IDOMPathFigure::getSegmentsCount ( ) [pure virtual]

Retrieves the number of path segments in the figure.

Returns

uint32. Returns the number of path segments in the figure.

8.215.2.8 getStartPoint()

virtual const FPoint& IDOMPathFigure::getStartPoint ( ) const [pure virtual]

Retrieves the start point for the first segment in the figure.

Returns

FPoint. Returns the start point for the first segment in the figure.

8.215.2.9 setIsClosed()

virtual bool 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.

The default is false.
8.215 IDOMPathFigure Class Reference

Parameters

| closed | New value of IsClosed |

Returns

bool. Returns true on success, false if the call fails.

8.215.2.10 setIsFilled()

virtual bool 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.

Parameters

| filled | New value of IsFilled |

Returns

bool. Returns true on success, false if the call fails.

8.215.2.11 setStartPoint()

virtual bool IDOMPathFigure::setStartPoint ( const FPoint & sp ) [pure virtual]

Sets the start point for the first segment in the figure.

Parameters

| sp | New value of the start point of the first segment in the figure |

Returns

bool. Returns true on success, false if the call fails.

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

- idompathgeometry.h
8.216  IDOMPathGeometry Class Reference

Interface to a path geometry node.

#include <idompathgeometry.h>

Inheritance diagram for IDOMPathGeometry:

```
IRLObject

IEDLObject

IDOMPathGeometry
```

Classes

• class Data
  
  Initialization data.

Public Types

• enum eFillRule { eEvenOdd, eNonZero }
  
  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 bool setFillRule (eFillRule fr)=0
  
  Sets the fill rule for the path. The valid values are specified by eFillRule.

• virtual bool getRenderTransform (FMatrix &matrix) const =0
  
  Retrieves the render transform matrix for the path.

• virtual bool setRenderTransform (const FMatrix &matrix)=0
  
  Sets the render transform matrix for the path.

• virtual IDOMFigureCollectionEnumPtr getFigureCollectionEnum ()=0
  
  Retrieves a navigable list of the figures within the path.

• virtual uint32 getFiguresCount ()=0

Generated by Doxygen
Retrieves the number of figures in the path.

- virtual void clearFigureCollection ()=0
  Removes all the figures from the path.

- virtual bool addFigure (IDOMPathFigurePtr &ptrFigure)=0
  Appends a figure to the path figure collection.

- virtual bool getBounds (FRect &bounds, bool applyTransform=true)=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 getShape (IDOMShapePtr &ptrShape, FMatrix &transform, float resolution, IEDLClassFactory *pFactory)=0
  Get the scan-converted shape of this path geometry.

- virtual bool getSimplifiedGeometry (IDOMPathGeometryPtr &simpleGeometry, bool simplifyQuads, bool simplifyArcs, IEDLClassFactory *pFactory)=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 (const FMatrix &transform, float resolution, IEDLClassFactory *pFactory)=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. And exception of type IEDLError is thrown on failure.

### Static Public Member Functions

- static EDL_API IDOMPathGeometryPtr create (IEDLClassFactory *pFactory, const FRect &rect, bool close=true)
  Simplified creator for a rectangular path geometry. The geometry will consist of a path beginning at x,y value of the rect, proceeding in a clockwise direction. Throws an IEDLError on failure.

- static const CClassID & classID ()
  Retrieves class id of IDOMPathGeometry.

### Additional Inherited Members

8.216.1 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.
8.216.2 Member Enumeration Documentation

8.216.2.1 eFillRule

```cpp
global 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.

**Enumerators**

<table>
<thead>
<tr>
<th>enumerator</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eEvenOdd</td>
<td>Use the even/odd fill algorithm.</td>
</tr>
<tr>
<td>eNonZero</td>
<td>Use the non-zero fill algorithm.</td>
</tr>
</tbody>
</table>

8.216.3 Member Function Documentation

8.216.3.1 addFigure()

```cpp
virtual bool IDOMPathGeometry::addFigure ( 
    IDOMPathFigurePtr & ptrFigure ) [pure virtual]
```

Appends a figure to the path figure collection.

**Parameters**

| pathFigure | Path figure to add |

**Returns**

`bool`. Returns true on success, false if the call fails.
8.216.3.2  classID()

static const CClassID & IDOMPathGeometry::classID ( ) [inline], [static]

Retrieves class id of IDOMPathGeometry.

Returns

CClassID class id of the element

8.216.3.3  create()

static EDL_API IDOMPathGeometryPtr IDOMPathGeometry::create (  
    IEDLClassFactory * pFactory,  
    const FRect & rect,  
    bool close = true ) [static]

Simplified creator for a rectangular path geometry. The geometry will consist of a path beginning at x,y value of the rect, proceeding in a clockwise direction. Throws an IEDLError on failure.

Parameters

| pFactory | The factory to use. |
| rect | The rect to use. |
| close | Whether or not the rectangular path should be closed |

Returns

IDOMPathGeometryPtr The new geometry.

8.216.3.4  getBounds()

virtual bool IDOMPathGeometry::getBounds (  
    FRect & bounds,  
    bool applyTransform = true ) [pure virtual]

Finds the conservative bounding box of the geometry.

Parameters

| bounds | Reference parameter to receive the bounding box. |
| applyTransform | Controls whether or not the receiver's transform is applied to the bounds (if it has one). Passing true (default) gives you results in the coordinate space of the object enclosing the receiver, while passing false will give you results in the coordinate space active inside the object. |
Returns

bool. Returns true on success, false if the call fails.

8.216.3.5 getFigureCollectionEnum()

virtual IDOMFigureCollectionEnumPtr IDOMPathGeometry::getFigureCollectionEnum ( ) [pure virtual]

Retrieves a navigable list of the figures within the path.

Returns

IDOMFigureCollectionEnumPtr. Returns a smart pointer to the list of path figures.

8.216.3.6 getFiguresCount()

virtual uint32 IDOMPathGeometry::getFiguresCount ( ) [pure virtual]

Retrieves the number of figures in the path.

Returns

uint32. Returns the number of figures in the path collection.

8.216.3.7 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 (EvenOdd default)

8.216.3.8 getFlattenedGeometry()

virtual IDOMPathGeometryPtr IDOMPathGeometry::getFlattenedGeometry ( const FMatrix & transform, float resolution, IEDLClassFactory * pFactory ) [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. An exception of type IEDLError is thrown on failure.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>transform</td>
<td>The external transformation applied to the geometry. Needed to determine the visible size of the geometry, and hence influences the level of curve approximation.</td>
</tr>
<tr>
<td>resolution</td>
<td>The intended display resolution. The flattened geometry will be computed such that the curved approximation is not noticeable when viewed at that resolution.</td>
</tr>
<tr>
<td>pFactory</td>
<td>A pointer to the EDL class factory.</td>
</tr>
</tbody>
</table>

Returns

*IDOMPathGeometryPtr* The resulting flattened geometry, or this geometry if no flattening is required.

---

**8.216.3.9 getIsRect()**

```cpp
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rect</td>
<td>Reference to receive the dimensions of the rectangle if the geometry indeed represents a rectangle.</td>
</tr>
</tbody>
</table>

Returns

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

---

**8.216.3.10 getRenderTransform()**

```cpp
virtual bool IDOMPathGeometry::getRenderTransform ( 
    FMatrix & matrix ) const [pure virtual]
```

Retrieves the render transform matrix for the path.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>matrix</td>
<td>Reference parameter to receive the render transform matrix.</td>
</tr>
</tbody>
</table>

Returns

*bool*. Returns true on success, false if the call fails.
8.216.3.11 getShape()

virtual bool IDOMPathGeometry::getShape (  
    IDOMShapePtr & ptrShape,  
    FMatrix & transform,  
    float resolution,  
    IEDLClassFactory * pFactory ) [pure virtual]

Get the scan-converted shape of this path geometry.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ptrShape</td>
<td>Reference parameter to receive a smart pointer to the shape.</td>
</tr>
<tr>
<td>transform</td>
<td>The transform that should be applied before scan conversion.</td>
</tr>
<tr>
<td>resolution</td>
<td>The resolution that should be used for scan conversion.</td>
</tr>
<tr>
<td>pFactory</td>
<td>A pointer to the EDL class factory.</td>
</tr>
</tbody>
</table>

Returns

bool. Returns true on success, false if the call fails.

8.216.3.12 getSimplifiedGeometry()

virtual bool IDOMPathGeometry::getSimplifiedGeometry (  
    IDOMPathGeometryPtr & simpleGeometry,  
    bool simplifyQuads,  
    bool simplifyArcs,  
    IEDLClassFactory * pFactory ) [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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>simpleGeometry</td>
<td>Reference parameter to receive the simple geometry.</td>
</tr>
<tr>
<td>simplifyQuads</td>
<td>If true, any quadratic segments will be converted to regular cubic segments in the simplified geometry.</td>
</tr>
<tr>
<td>simplifyArcs</td>
<td>If true, any arc segments will be converted to regular cubic segments or line segments as appropriate.</td>
</tr>
<tr>
<td>pFactory</td>
<td>A pointer to the EDL class factory.</td>
</tr>
</tbody>
</table>

Returns

bool Returns true on success.
8.216.3.13  setFillRule()

virtual bool 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

<table>
<thead>
<tr>
<th>fr</th>
<th>Fill rule</th>
</tr>
</thead>
</table>

Returns

bool Returns true on success.

8.216.3.14  setRenderTransform()

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

Sets the render transform matrix for the path.

Parameters

| matrix | The new render transform matrix. |

Returns

bool. Returns true on success, false if the call fails.

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

- idompathgeometry.h

8.217  IDOMPathNode Class Reference

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

#include <idompath.h>
Inheritance diagram for IDOMPathNode:

```
IRObject
  |   |
  |   IEDLObject
  |   |
  |   IDOMNode
  |   |
  
```

### 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 bool getRenderTransform (FMatrix &matrix) 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 bool 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 bool 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 bool 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

Sets 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 bool 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 bool 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 bool 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 bool setStrokeMiterLimitTreatment (eStrokeMiterLimitTreatment treatment)=0

Sets the stroke miter limit treatment for this path. 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 bool 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 bool 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 bool 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 bool 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 bool setStrokeDashLineCap (eStrokeLineCap slc)=0
  Sets the stroke dash line cap.
- virtual bool getSnapsToDevicePixels () const =0
  Retrieves the snapsToDevicePixels setting.
- virtual bool setSnapsToDevicePixels (bool s2dp)=0
  Sets snapsToDevicePixels.
- virtual bool getLanguage (EDLString &lang) const =0
  Retrieves the default language of the path node and any of its children.
- virtual bool setLanguage (const EDLString &lang)=0
  Sets default language of the <Path> element and any of its children.
- virtual bool getNavigateLink (IDOMTargetPtr &target) const =0
  Retrieves the target of a hyperlink.
- virtual bool setNavigateLink (const IDOMTargetPtr &target)=0
  Sets the target of a hyperlink.
- virtual bool getAutomationPropertiesName (EDLString &propname) const =0
  Retrieves the automation properties name of the path.
- virtual bool setAutomationPropertiesName (const EDLString &propname)=0
  Sets the automation properties name of the path.
- virtual bool getAutomationPropertiesHelpText (EDLString &helptext) const =0
  Retrieves the automation properties help text of the path.
- virtual bool 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 IDoubleCollectionEnumPtr getStrokeDashCollectionEnum ()=0
  Retrieves an enumerator - navigable list interface - of the stroke dash array.
- virtual uint32 getStrokeDashsCount ()=0
  Retrieves the length of the stroke dash array.
- virtual void clearStrokeDashCollection ()=0
  Clears the stroke dash array.
- virtual bool addStrokeDash (double value)=0
  Append a stroke dash to the stroke dash array.
- virtual bool getFill (IDOMBrushPtr &ptrFill) const =0
  Retrieves the fill brush for the path.
- virtual bool setFill (const IDOMBrushPtr &ptrFill)=0
  Sets the fill brush for the path.
- virtual bool getStroke (IDOMBrushPtr &ptrStroke) const =0
  Retrieves the stroke brush for the path.
Retrieves the stroke brush for the path.

- virtual bool setStroke (const IDOMBrushPtr &ptrStroke)=0
  Sets the stroke brush for the path.

- virtual bool getOpacityMask (IDOMBrushPtr &ptrOpacityMask) const =0
  Retrieves the opacity mask for the path.

- virtual bool setOpacityMask (const IDOMBrushPtr &ptrOpacityMask)=0
  Sets the opacity mask for the path.

- virtual bool getPathData (IDOMPathGeometryPtr &ptrPathData) const =0
  Retrieves a smart pointer to the path geometry node.

- virtual bool setPathData (const IDOMPathGeometryPtr &ptrPathData)=0
  Sets the path geometry node.

- virtual bool getClip (IDOMPathGeometryPtr &ptrClip) const =0
  Retrieves a smart pointer to the clip geometry node.

- virtual bool 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 bool getShape (IDOMShapePtr &ptrShape, const FMatrix &transform, float resolution, IEDLClassFactory ∗pFactory)=0
  Get the scan-converted shape of this path.

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=eClipLongMiters, 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.

Additional Inherited Members

8.217.1 Detailed Description

Interface to an EDL path node. A path node specifies a geometry that can be filled with a brush.
Vector graphics are created using the PathNode class. A full set of properties is available to describe the visual characteristics of the graphic. The description of the geometry of the path is described by the Data property. Raster images are included in fixed page markup by specifying a PathNode filled with an ImageBrush.

A PathNode is the sole means of adding vector graphics and images to a fixed page. It defines a single vector graphic to be rendered on a page. Some properties of the PathNode are composable, meaning that the markings rendered to the page are determined by a combination of the property and all of the like-named properties of its parent and ancestor elements.

### 8.217.2 Member Enumeration Documentation

#### 8.217.2.1 eStrokeLineCap

```
enum IDOMPathNode::eStrokeLineCap
```

Specifies the different types of line end caps available.

**Enumerator**

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eFlatCap</td>
<td>Flat end caps.</td>
</tr>
<tr>
<td>eSquareCap</td>
<td>Square end caps.</td>
</tr>
<tr>
<td>eRoundCap</td>
<td>Round end caps.</td>
</tr>
<tr>
<td>eTriangleCap</td>
<td>Triangular end caps.</td>
</tr>
</tbody>
</table>

#### 8.217.2.2 eStrokeLineJoin

```
enum IDOMPathNode::eStrokeLineJoin
```

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

**Enumerator**

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eMiterJoin</td>
<td>Use mitered joins.</td>
</tr>
<tr>
<td>eBevelJoin</td>
<td>Use bevelled joins.</td>
</tr>
<tr>
<td>eRoundJoin</td>
<td>Use rounded joins.</td>
</tr>
</tbody>
</table>

#### 8.217.2.3 eStrokeMiterLimitTreatment

```
enum IDOMPathNode::eStrokeMiterLimitTreatment
```

Generated by Doxygen
Chooses how mitered lines 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.
8.217.3 Member Function Documentation

8.217.3.1 addStrokeDash()

virtual bool IDOMPathNode::addStrokeDash ( double value ) [pure virtual]

Append a stroke dash to the stroke dash array.

Parameters

| value | The new stroke dash array item. |

Returns

bool True on success, false if the call fails.

8.217.3.2 classID()

static const CClassID & IDOMPathNode::classID ( ) [inline], [static]

Retrieves class id of IDOMPathNode.

Returns

CClassID Class id of the element

8.217.3.3 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 = eEEMDefault ) [static]

Simplified creator for a filled path. Throws an IEDLError on failure.
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pFactory</td>
<td>The factory to use.</td>
</tr>
<tr>
<td>geometry</td>
<td>The path data/geometry to fill.</td>
</tr>
<tr>
<td>brush</td>
<td>The fill brush to use.</td>
</tr>
<tr>
<td>renderTransform</td>
<td>The render transform to use.</td>
</tr>
<tr>
<td>clip</td>
<td>The geometry to use for clipping. NULL if no clip.</td>
</tr>
<tr>
<td>opacity</td>
<td>The opacity to use.</td>
</tr>
<tr>
<td>blendMode</td>
<td>The blend mode to use.</td>
</tr>
<tr>
<td>edgeMode</td>
<td>The edge mode to use.</td>
</tr>
</tbody>
</table>

Returns

`IDOMPathNodePtr` The new path.

8.217.3.4 `createImage()`

```cpp
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pFactory</td>
<td>The factory to use.</td>
</tr>
<tr>
<td>image</td>
<td>The image to use.</td>
</tr>
<tr>
<td>viewPort</td>
<td>The destination rectangle.</td>
</tr>
<tr>
<td>renderTransform</td>
<td>The render transform to apply.</td>
</tr>
</tbody>
</table>

Returns

`IDOMPathNodePtr` The new path.

8.217.3.5 `createStroked()`

```cpp
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

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

Returns

IDOMPathNodePtr The new path.

8.217.3.6 getAutomationPropertiesHelpText()

virtual bool IDOMPathNode::getAutomationPropertiesHelpText ( "helptext" 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.

**Parameters**

| helptext | Automation properties help text |

**Returns**

`bool` True on success, false if the call fails.

---

### 8.217.3.7 getAutomationPropertiesName()

```cpp
virtual bool IDOMPathNode::getAutomationPropertiesName ( EDLString & propname ) 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.

**Parameters**

| propname | Automation properties name |

**Returns**

`bool` True on success, false if the call fails.

---

### 8.217.3.8 getBlendMode()

```cpp
virtual eBlendMode IDOMPathNode::getBlendMode ( ) const [pure virtual]
```

Get the blend mode to be used for rendering this path.

**Returns**

`eBlendMode` The blend mode.
virtual bool IDOMPathNode::getClip (IDOMPathGeometryPtr & ptrClip) 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.

Parameters

ptrClip Reference parameter to receive a smart pointer to the clip geometry node.

Returns

bool True on success, false if the call fails.

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 function returns the edge mode of the path.

virtual bool IDOMPathNode::getFill (IDOMBrushPtr & ptrFill) const [pure virtual]

Retrieves the fill brush for the path.

Parameters

ptrFill Reference parameter to receive a smart pointer to the fill brush.

Returns

bool True on success, false if the call fails.
8.217.3.12 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.

8.217.3.13 getLanguage()

virtual bool IDOMPathNode::getLanguage(
    EDLString & lang) 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.

Parameters

| lang | Default language |

Returns

bool True on success, false if the call fails.

8.217.3.14 getNavigateLink()

virtual bool IDOMPathNode::getNavigateLink(
    IDOMTargetPtr & target) const [pure virtual]

Retrieves the target of a hyperlink.

Parameters

| target | A smart pointer to receive the target of the hyperlink. |
Returns

bool True on success, false if the call fails.

8.217.3.15 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

8.217.3.16 getOpacityMask()

virtual bool IDOMPathNode::getOpacityMask (IDOMBrushPtr & ptrOpacityMask) 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.

Parameters

ptrOpacityMask Smart pointer to opacity mask

Returns

bool True on success, false if the call fails.

8.217.3.17 getPathData()

virtual bool IDOMPathNode::getPathData (IDOMPathGeometryPtr & ptrPathData) const [pure virtual]

Retrieves a smart pointer to the path geometry node.
Parameters

| ptrPathData | Reference parameter to receive a pointer to the path geometry node. |

Returns

bool True on success, false if the call fails.

8.217.3.18 getRenderTransform()

virtual bool IDOMPathNode::getRenderTransform
(
    FMatrix & matrix
) 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.

Parameters

| matrix | Render transform matrix. |

Returns

bool True on success, false if the call fails.

8.217.3.19 getShape()

virtual bool IDOMPathNode::getShape
(
    IDOMShapePtr & ptrShape,
    const FMatrix & transform,
    float resolution,
    IEDLClassFactory ∗ pFactory
) [pure virtual]

Get the scan-converted shape of this path.

Parameters

| ptrShape | Reference parameter to receive a smart pointer to the shape. |
| transform | The transform that should be applied before scan conversion. |
| resolution | The resolution that should be used for scan conversion. |
| pFactory | The factory to use. |

Returns

bool True on success, false if the call fails.
8.217.3.20  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.

8.217.3.21  getSnapsToDevicePixels()

virtual bool IDOMPathNode::getSnapsToDevicePixels ( ) const [pure virtual]

Retrieves the snapsToDevicePixels setting.

The ignorable attribute snapsToDevicePixels can be set to allow consumers or viewers that perform anti-aliasing to "snap" any path control points that are situated on the path bounding box to whole device pixels.

Returns

  bool True if control points snap to the nearest device pixels.

8.217.3.22  getStroke()

virtual bool IDOMPathNode::getStroke (  
    IDOMBrushPtr & ptrStroke ) const [pure virtual]

Retrieves the stroke brush for the path.

Parameters

| ptrStroke | Reference parameter to receive a smart pointer to the stroke brush |

Returns

  bool True on success, false if the call fails.
8.217.3.23 getStrokeDashCollectionEnum()

virtual IDoubleCollectionEnumPtr IDOMPathNode::getStrokeDashCollectionEnum() [pure virtual]

Retrieves an enumerator - navigable list interface - of 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

IDoubleCollectionEnumPtr The enumerator of the stroke dash array

8.217.3.24 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)

8.217.3.25 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)
8.217.3.26 getStrokeDashsCount()

virtual uint32 IDOMPathNode::getStrokeDashsCount ( ) [pure virtual]

Retrieves the length of the stroke dash array.

Returns
  uint32 The number of entries in the stroke dash array.

8.217.3.27 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.

8.217.3.28 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)
8.217.3.29  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).

8.217.3.30  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).

8.217.3.31  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.
8.217.3.32  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)

8.217.3.33  setAutomationPropertiesHelpText()

virtual bool IDOMPathNode::setAutomationPropertiesHelpText ( const EDLString & helptext )  [pure virtual]

Sets the automation properties help text of the path. Automation properties help text is a detailed description of the path's content, used for accessibility purposes, particularly if the path is filled with an image brush, or a set of vector graphics and text elements intended to comprise a single vector graphic.

Parameters

| helptext | Automation properties help text |

Returns

   bool True on success, false if the call fails.

8.217.3.34  setAutomationPropertiesName()

virtual bool IDOMPathNode::setAutomationPropertiesName ( const EDLString & propname )  [pure virtual]

Sets the automation properties name of the path. Automation properties help text is a detailed description of the path's content, used for accessibility purposes, particularly if the path is filled with an image brush, or a set of vector graphics and text elements intended to comprise a single vector graphic.

Parameters

| propname | The new automation properties name. |
8.217 IDOMPathNode Class Reference

Returns

**bool** True on success, false if the call fails.

8.217.3.35 **setBlendMode()**

```cpp
virtual bool IDOMPathNode::setBlendMode ( 
    eBlendMode blendMode ) [pure virtual]
```

Set the blend mode to be used for rendering this path. Note: modes other than Normal are not directly representable in XPS.

Parameters

| blendMode | The desired blend mode. |

Returns

**bool** True on success, false if the call fails.

8.217.3.36 **setClip()**

```cpp
virtual bool IDOMPathNode::setClip ( 
    const IDOMPathGeometryPtr & ptrClip ) [pure virtual]
```

Sets the clip geometry.

The clip geometry node specifies a clipping region which describes the geometric area to be preserved. The remainder is not rendered.

Parameters

| ptrClip   | Smart pointer to the new clip geometry node. |

Returns

**bool** True on success, false if the call fails.

8.217.3.37 **setEdgeMode()**

```cpp
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 anti-aliasing.

Parameters

- **em**: The new edge mode for the canvas.

### 8.217.3.38 setFill()

```cpp
class IDOMPathNode {
  virtual bool setFill (const IDOMBrushPtr & ptrFill) = 0;
}
```

Sets the fill brush for the path.

Parameters

- **ptrFill**: Smart pointer to the new fill brush.

Returns

- **bool**: True on success, false if the call fails.

### 8.217.3.39 setLanguage()

```cpp
class IDOMPathNode {
  virtual bool setLanguage (const EDLString & lang) = 0;
}
```

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/](http://www.w3.org/International/articles/language-tags/).

The language is specified according to RFC 3066.

Parameters

- **lang**: Default language

Returns

- **bool**: True on success, false if the call fails.
8.217.3.40 setNavigateLink()

virtual bool IDOMPathNode::setNavigateLink (
    const IDOMTargetPtr & target ) [pure virtual]

Sets the target of a hyperlink.

Parameters

- **target**: The target for the hyperlink.

Returns

- **bool**: True on success, false if the call fails.

8.217.3.41 setOpacity()

virtual bool IDOMPathNode::setOpacity ( 
    float opc ) [pure virtual]

Sets the opacity value of the path. The opacity value defines the uniform transparency of the path. This is a number between 0 (fully transparent) and 1 (fully opaque).

Parameters

- **opc**: The new opacity value.

Returns

- **bool**: True on success, false if the call fails.

8.217.3.42 setOpacityMask()

virtual bool IDOMPathNode::setOpacityMask ( 
    const IDOMBrushPtr & ptrOpacityMask ) [pure virtual]

Sets the opacity mask for the path.

The opacity mask specifies a mask of alpha values that is applied to the path in the same fashion as the simple opacity setting, but allowing different alpha values for different area of the canvas. With an opacity mask, you can combine an object with one or more other objects to define transparent areas of the shape.

Opacity masks use the values of one object or image to define the transparent areas of another. The objects that define the transparency can be any type of visual element - solid objects, strokes, gradients, raster images, text or combinations of all of the above.
Parameters

`ptrOpacityMask`  Smart pointer to brush

Returns

`bool`  True on success, false if the call fails.

8.217.3.43  setPathData()

```cpp
virtual bool IDOMPathNode::setPathData (  
    const IDOMPathGeometryPtr & ptrPathData ) [pure virtual]
```

Sets the path geometry node.

Parameters

`ptrPathData`  Smart pointer to the new path geometry node.

Returns

`bool`  True on success, false if the call fails.

8.217.3.44  setRenderTransform()

```cpp
virtual bool IDOMPathNode::setRenderTransform (  
    const FMatrix & matrix ) [pure virtual]
```

Sets the render transform matrix. The render transform matrix establishes a new coordinate frame for all attributes of the path and for all child elements of the path, such as the path geometry.

Parameters

`matrix`  The new render transform matrix.

Returns

`bool`  True on success, false if the call fails.

8.217.3.45  setShouldZeroWidthLinesBeVisible()

```cpp
virtual bool IDOMPathNode::setShouldZeroWidthLinesBeVisible (  
    bool visible ) [pure virtual]
```
Sets whether or not zero width strokes in this path should be rendered visibly as a cosmetic line.

Parameters

| visible |

Returns

bool True on success, false if the call fails.

8.217.3.46 setSnapsToDevicePixels()

virtual bool IDOMPathNode::setSnapsToDevicePixels ( bool s2dp ) [pure virtual]

Sets snapsToDevicePixels.

The ignorable attribute snapsToDevicePixels can be set to allow consumers or viewers that perform anti-aliasing to “snap” any path control points that are situated on the path bounding box to whole device pixels.

Parameters

| s2dp | New value of snapsToDevicePixels. |

Returns

bool True on success, false if the call fails.

8.217.3.47 setStroke()

virtual bool IDOMPathNode::setStroke ( const IDOMBrushPtr & ptrStroke ) [pure virtual]

Sets the stroke brush for the path.

Parameters

| ptrStroke | Smart pointer to the new stroke brush. |

Returns

bool True on success, false if the call fails.
8.217.3.48 setStrokeDashLineCap()

virtual bool IDOMPathNode::setStrokeDashLineCap (  
    eStrokeLineCap slc )  [pure virtual]  

Sets the stroke dash line cap. 

See also  
  eStrokeLineCap

Parameters

| slc  | The stroke dash line cap (Flat default) |

Returns

  bool True on success, false if the call fails.

8.217.3.49 setStrokeDashOffset()

virtual bool IDOMPathNode::setStrokeDashOffset (  
    double sdo )  [pure virtual]  

Sets the stroke dash offset value. This adjusts the start point for repeating the dash array pattern. If this value is omitted, the dash array aligns with the origin of the stroke. Values are specified as multiples of the stroke thickness.

Parameters

| sdo  | Stroke dash offset (0.0 default) |

Returns

  bool True on success, false if the call fails.

8.217.3.50 setStrokeEndLineCap()

virtual bool IDOMPathNode::setStrokeEndLineCap (  
    eStrokeLineCap slc )  [pure virtual]  

Sets the line cap type for the end of the stroke.

See also  
  eStrokeLineCap
Parameters

| slc                  | The stroke end line cap (Flat default) |

Returns

bool True on success, false if the call fails.

8.217.3.51 setStrokeLineJoin()

virtual bool IDOMPathNode::setStrokeLineJoin (eStrokeLineJoin slj) [pure virtual]

Sets the stroke line join value. The stroke line join specifies how a stroke is drawn at a corner of a path. Valid values are specified by eStrokeLineJoin. If mitered joins are selected, the value of StrokeMiterLimit is used in drawing the stroke.

See also

eStrokeLineJoin

Parameters

| slj                  | The stroke line join (Miter default) |

Returns

bool True on success, false if the call fails.

8.217.3.52 setStrokeMiterLimit()

virtual bool IDOMPathNode::setStrokeMiterLimit (double sml) [pure virtual]

Sets the stroke miter limit. The stroke miter limit is the ratio between the maximum miter length and half of the stroke thickness. This value must be equal to or greater than 1.0. The value is significant only if the StrokeLineJoin attribute specifies mitered joins.

Parameters

| sml                  | Stroke miter limit (10.0 default). |
Returns

**bool** True on success, false if the call fails.

8.217.3.53  setStrokeMiterLimitTreatment()

virtual bool IDOMPathNode::setStrokeMiterLimitTreatment (  
  eStrokeMiterLimitTreatment treatment ) [pure virtual]

Sets the stroke miter limit treatment. The stroke miter treatment specifies how miters extending beyond the limit should be treated.

Parameters

| treatment | The desired stroke miter treatment |

Returns

**bool** True on success, false if the call fails.

8.217.3.54  setStrokeStartLineCap()

virtual bool IDOMPathNode::setStrokeStartLineCap (  
  eStrokeLineCap slc ) [pure virtual]

Sets the line cap type for the start of the stroke.

See also

[ eStrokeLineCap ]

Parameters

| slc | The stroke start line cap (Flat default) |

Returns

**bool** True on success, false if the call fails.

8.217.3.55  setStrokeThickness()

virtual bool 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>st</td>
<td>Stroke thickness (1.0 default)</td>
</tr>
</tbody>
</table>

Returns

`bool` True on success, false if the call fails.

8.217.3.56 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pFactory</td>
<td>The factory to use.</td>
</tr>
</tbody>
</table>

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

8.218 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:

```
IRCObject  EDLObject  IDOMPathSegment
       |                     |
       |                     |
       |                     |
       |                     |
       |                     |
       DOMArcSegment       
       DOMPolyBezierSegment
       DOMPolyLineSegment
       DOMPolyQuadraticBezierSegment
```

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 bool 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.
- virtual bool getBounds (FPoint &startPoint, FRect &bounds)=0
  Gets the conservative bounding box of the segment given the start point.
- virtual bool getEndPoint (FPoint &endPoint)=0
  Gets the end point of the segment.

Additional Inherited Members

8.218.1 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.

8.218.2 Member Function Documentation

8.218.2.1 getBounds()

```
virtual bool IDOMPathSegment::getBounds {
    FPoint & startPoint,
    FRect & bounds ) [pure virtual]
```

Gets the conservative bounding box of the segment given the start point.
### Parameters

<table>
<thead>
<tr>
<th>startPoint</th>
<th>The start point of the segment.</th>
</tr>
</thead>
<tbody>
<tr>
<td>bounds</td>
<td>Reference parameter to receive the bounding box.</td>
</tr>
</tbody>
</table>

### Returns

bool. Returns true on success, false if the call fails.

#### 8.218.2.2 getEndPoint()

```cpp
class IDOMPathSegment {
    virtual bool getEndPoint(FPoint &endPoint) const; // pure virtual
};
```

Gets the end point of the segment.

**Parameters**

<table>
<thead>
<tr>
<th>endPoint</th>
<th>Reference parameter to receive the end point of the segment.</th>
</tr>
</thead>
</table>

**Returns**

bool. Returns true on success, false if the call fails.

#### 8.218.2.3 setIsStroked()

```cpp
class IDOMPathSegment {
    virtual bool setIsStroked(bool isStroked) const; // 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.

**Parameters**

| isStroked | Whether the stroke for this segment of the path is drawn.
|-----------|---------------------------------------------------|

**Returns**

bool. Returns true if this line segment is stroked, false if it is not.
Parameters

| isStroked | New value of IsStroked. |

Returns

`bool`. Returns true on success, false if the call fails.

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

- `idompathgeometry.h`

### 8.219 IDOMPCLImage Class Reference

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

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

Inheritance diagram for IDOMPCLImage:

```
ÍRObject

IEDLObject

IDOMHashable

IDOMResource

IDOMImage

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

8.219.1 Detailed Description

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

8.219.2 Member Function Documentation

8.219.2.1 classID()

static const CClassID & IDOMCLImage::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
8.220 IDOMPDFImage Class Reference

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

```cpp
#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 IDecodeParamsPtr getDecodeParameters () const =0
  
  Retrieves the decode parameters. May be NULL.

- 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 uint32 getBitsPerComponent () const =0
  
  Retrieves the bits per component of the source data. For JPEG2000, this information is available only in the image stream and this function will return 0. In this case please use the IImageStream interface.

- virtual CEDLVector<uint16> getColorKey () const =0
  
  Retrieves the color key for mask generation. An empty vector will be returned if there is no color key.

- virtual IDOMColorSpacePtr getColorSpace () const =0
  
  Retrieves the color space to be used with this image. For JPEG2000, this may be NULL if the colorspace needs to be determined from the image data. In this case please use the IImageStream interface.

- virtual eImageAlpha getAlphaDetails () const =0
  
  Returns if the image has alpha, and if the colour samples are premultiplied.

Static Public Member Functions

- static const CClassID & classID ()

  Retrieves class id of IDOMPDFImage.

Additional Inherited Members

8.220.1 Detailed Description

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

8.220.2 Member Function Documentation

8.220.2.1 classID()

static const CClassID & IDOMPDFImage::classID ( ) [inline], [static]

Retrieves class id of IDOMPDFImage.

Returns

  CClassID Class id of the element
8.220.2.2 getAlphaDetails()

virtual eImageAlpha IDOMPDFImage::getAlphaDetails ( ) const [pure virtual]

Returns if the image has alpha, and if the colour samples are premultiplied.

Returns

\textbf{eImageAlpha} The image alpha details

8.220.2.3 getBitsPerComponent()

virtual uint32 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 \textbf{IImageStream} interface.

Returns

\textbf{uint32} The number of bits per component

8.220.2.4 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

\textbf{CEDLVector<\texttt{uint16}>} The color key

8.220.2.5 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 \textbf{IImageStream} interface.

Returns

\textbf{IDOMColorSpacePtr} The color space
8.220.2.6 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

8.220.2.7 getDecodeParameters()

virtual IDecodeParamsPtr IDOMPDFImage::getDecodeParameters() const [pure virtual]

Retrieves the decode parameters. May be NULL.

Returns

IDecodeParamsPtr The decode parameters

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

- idomimageresource.h

8.221 IDOMPNGImage Class Reference

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

#include <idomimageresource.h>
Inheritance diagram for IDOMPNGImage:

```
                     IRCObject
                      |
                      |    IEDLObject
                      |     |
                      |     |  IDOMHashable
                      |     |  IDOMResource
                      |     |
                      |     IDOMImage
                      |       |
                      |       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. Throws an IEDLError on failure.

- **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. Throws an IEDLError on failure.

- **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. Throws an IEDLError on failure.

- **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. Throws an IEDLError on failure.

- **static const CClassID & classID ()**

  Retrieves class id of IDOM.
Additional Inherited Members

8.221.1 Detailed Description

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

8.221.2 Member Function Documentation

8.221.2.1 classID() 

```
static const CClassID & IDOMPNGImage::classID() [inline], [static]
```

Retrieves class id of IDOM.

Returns

**CClassID** Class id of the element

8.221.2.2 create()

```
static EDL_API IDOMImagePtr IDOMPNGImage::create (
    IEDLClassFactory * pFactory,
    const IInputStreamPtr & stream ) [static]
```

Create a PNG Image resource with the given PNG stream. Throws an IEDLError on failure.

Parameters

| **pFactory** | The EDL Class factory to use. |
| **stream**   | The stream containing the PNG image. |

Returns

**IDOMImagePtr** The new image.

8.221.2.3 encode() [1/3]

```
static EDL_API void IDOMPNGImage::encode ( 
    const ISessionPtr & pSession,
```

Generated by Doxygen
const IDOMImagePtr & image,
const IOutputStreamPtr & stream ) [static]

Encode an image as a PNG stream, returning the stream. This routine may convert the image samples into a form
that may be encoded as PNG if required, such as by converting to a supported color space. Throws an IEDLError
on failure.

Parameters

<table>
<thead>
<tr>
<th>pSession</th>
<th>The relevant EDL session</th>
</tr>
</thead>
<tbody>
<tr>
<td>image</td>
<td>The image to be encoded</td>
</tr>
<tr>
<td>stream</td>
<td>The stream to use to store the image data.</td>
</tr>
</tbody>
</table>

8.221.2.4 encode() [2/3]

static EDL_API void IDOMPNGImage::encode (
const ISessionPtr & pSession,
const IImageFramePtr & frame,
const IOutputStreamPtr & stream ) [static]

Encode the contents of an IImageFrame as a PNG stream, returning the stream. This routine may convert the
image samples into a form that may be encoded as PNG if required, such as by converting to a supported color
space. Throws an IEDLError on failure.

Parameters

<table>
<thead>
<tr>
<th>pSession</th>
<th>The relevant EDL session</th>
</tr>
</thead>
<tbody>
<tr>
<td>frame</td>
<td>The frame providing the source image data</td>
</tr>
<tr>
<td>stream</td>
<td>The stream to use to store the image data.</td>
</tr>
</tbody>
</table>

8.221.2.5 encode() [3/3]

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.
Throws an IEDLError on failure.
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pSession</td>
<td>The relevant EDL session</td>
</tr>
<tr>
<td>colorSpace</td>
<td>The color space of the frame buffer</td>
</tr>
<tr>
<td>dpi</td>
<td>The image resolution in dots-per-inch</td>
</tr>
<tr>
<td>bpc</td>
<td>Bits per component (pixel)</td>
</tr>
<tr>
<td>frameBuffer</td>
<td>The frame buffer to be encoded</td>
</tr>
<tr>
<td>width</td>
<td>The width of the frame buffer</td>
</tr>
<tr>
<td>height</td>
<td>The height of the frame buffer</td>
</tr>
<tr>
<td>stride</td>
<td>The offset in bytes in the frameBuffer from one scanline to the next. May be negative.</td>
</tr>
<tr>
<td>hasAlpha</td>
<td>Set true if the frame buffer has an alpha channel.</td>
</tr>
<tr>
<td>stream</td>
<td>The stream to use to store the image data.</td>
</tr>
</tbody>
</table>

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

- idomimageresource.h

8.222 IDOMPolyBezierSegment Class Reference

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

#include <idompathgeometry.h>

Inheritance diagram for IDOMPolyBezierSegment:

![Inheritance Diagram](image-url)
Classes

- class Data
  
  *Initialization data.*

Public Member Functions

- virtual IFPointCollectionEnumPtr getPointEnum ()=0
  
  *Retrieves a navigable list-an enumerator-of the points in the Bézier segment. The points list specifies control points for multiple Bézier segments.*

- virtual uint32 getPointsCount ()=0
  
  *Retrieves the number of points in the points list.*

- virtual void clearPoints ()=0
  
  *Clears the points list.*

- virtual bool addPoint (const FPoint &point)=0
  
  *Append a point to the list.*

Static Public Member Functions

- static const CClassID & classID ()
  
  *Retrieves class id of IDOMPolyBezierSegment.*

Additional Inherited Members

8.222.1 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 curve in the segment and terminate at the third point (x3n,y3n) in the Points attribute (where n is the curve being drawn). The tangents and curvature of each Bézier curve are controlled by the first two control points (x3n-2,y3n-2 and x3n-1,y3n-1) in the Points attribute. The Points attribute contains a multiple of three whitespace delimited pairs of comma delimited x,y values.

8.222.2 Member Function Documentation

8.222.2.1 addPoint()

```cpp
virtual bool IDOMPolyBezierSegment::addPoint (  
    const FPoint & point ) [pure virtual]
```

*Append a point to the list.*
Parameters

| FPoint | Point to append. |

Returns

bool. Returns true on success, false if the call fails.

8.222.2 classID()

static const CClassID & IDOMPolyBezierSegment::classID ( ) [inline], [static]

Retrieves class id of IDOMPolyBezierSegment.

Returns

CClassID class id of the element

8.222.3 getPointEnum()

virtual IFPointCollectionEnumPtr IDOMPolyBezierSegment::getPointEnum ( ) [pure virtual]

Retrieves a navigable list—an enumerator—of the points in the Bézier segment. The points list specifies control points for multiple Bézier segments.

Returns

IFPointCollectionEnumPtr. Returns the enumerator for the points list.

8.222.4 getPointsCount()

virtual uint32 IDOMPolyBezierSegment::getPointsCount ( ) [pure virtual]

Retrieves the number of points in the points list.

Returns

uint32. Returns the number of points in the points list.

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

- idompathgeometry.h
8.223  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.

```cpp
#include <idompathgeometry.h>
```

Inheritance diagram for IDOMPolyLineSegment:

```
Class Diagram
```

Classes

- class Data
  
  *Initialization data.*

Public Member Functions

- virtual IFPointCollectionEnumPtr getPointEnum ()=0
  
  *Retrieves a navigable list-an enumerator-of the points in the polyline segment.*

- virtual uint32 getPointsCount ()=0
  
  *Retrieves the number of points in the segment.*

- virtual void clearPoints ()=0
  
  *Clears the points list.*

- virtual bool addPoint (const FPoint &point)=0
  
  *Appends a point to the list.*

Static Public Member Functions

- static const CClassID & classID ()
  
  *Retrieves class id of IDOMPolyLineSegment.*
Additional Inherited Members

8.223.1 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.

8.223.2 Member Function Documentation

8.223.2.1 addPoint()

```cpp
virtual bool IDOMPolyLineSegment::addPoint (  
    const FPoint & point ) [pure virtual]
```

Appends a point to the list.

Parameters

- `FPoint` Point to append

Returns

- `bool`. Returns true on success, false if the call fails.

8.223.2.2 classID()

```cpp
static const CClassID & IDOMPolyLineSegment::classID ( ) [inline], [static]
```

Retrieves class id of IDOMPolyLineSegment.

Returns

- `CClassID` class id of the element

8.223.2.3 getPointEnum()

```cpp
virtual IFPointCollectionEnumPtr IDOMPolyLineSegment::getPointEnum ( ) [pure virtual]
```

Retrieves a navigable list-an enumerator-of the points in the polyline segment.

Returns

- `IFPointCollectionEnumPtr`. Returns a pointer to the points list enumerator.
8.223.2.4  getPointsCount()

virtual uint32 IDOMPolyLineSegment::getPointsCount () [pure virtual]

Retrieves the number of points in the segment.

Returns

uint32. Returns the size of the points list.

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

- idompathgeometry.h

8.224  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 (x2n-1, y2n-1) followed by the end point (x2n, y2n) for each quadratic Bézier curve (where n represents the quadratic Bézier curve).

#include <idompathgeometry.h>

Inheritance diagram for IDOMPolyQuadraticBezierSegment:
Classes

• class Data
  
  Initialization data.

Public Member Functions

• virtual IFPointCollectionEnumPtr getPointEnum ()=0
  
  Retrieves a navigable list-an enumerator-of the points in the polyquadratic Bézier segment.

• virtual uint32 getPointsCount ()=0
  
  Retrieves the number of points in the polyquadratic Bézier segment.

• virtual void clearPoints ()=0
  
  Clears the points list.

• virtual bool addPoint (const FPoint &point)=0
  
  Append a point to the list.

• virtual bool convertToCubicBezierSegment (IDOMPolyBezierSegmentPtr &newSegment, FPoint &startPoint, IEDLClassFactory *pFactory)=0
  
  Create a segment that represents this curve using a cubic bezier.

Static Public Member Functions

• static const CClassID & classID ()
  
  Retrieves the class id of IDOMPolyQuadraticBezierSegment.

Additional Inherited Members

8.224.1 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 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).

8.224.2 Member Function Documentation

8.224.2.1 addPoint()

virtual bool IDOMPolyQuadraticBezierSegment::addPoint (  
    const FPoint & point ) [pure virtual]

Append a point to the list.
Parameters

| FPoint     | Point to append |

Returns

bool. Returns true on success, false if the call fails.

8.224.2.2 classID()

static const CClassID & IDOMPolyQuadraticBezierSegment::classID () [inline], [static]

Retrieves the class id of IDOMPolyQuadraticBezierSegment.

Returns

CClassID. Returns the class id of the element.

8.224.2.3 convertToCubicBezierSegment()

virtual bool IDOMPolyQuadraticBezierSegment::convertToCubicBezierSegment ( 
    IDOMPolyBezierSegmentPtr & newSegment,
    FPoint & startPoint,
    IEDLClassFactory * pFactory ) [pure virtual]

Create a segment that represents this curve using a cubic bezier.

Parameters

<table>
<thead>
<tr>
<th>newSegment</th>
<th>Smart pointer to receive the resulting cubic bezier</th>
</tr>
</thead>
<tbody>
<tr>
<td>startPoint</td>
<td>The start point for the curve</td>
</tr>
<tr>
<td>factory</td>
<td>Pointer to the class factory to be used</td>
</tr>
</tbody>
</table>

Returns

bool. The function returns true on success, false on failure.

8.224.2.4 getPointEnum()

virtual IFPointCollectionEnumPtr IDOMPolyQuadraticBezierSegment::getPointEnum ( ) [pure virtual]

Retrieves a navigable list-an enumerator-of the points in the polyquadratic Bézier segment.
Returns

IFPointCollectionEnumPtr. Returns the enumerator to the points list.

8.224.2.5 getPointsCount()

virtual uint32 IDOMPolyQuadraticBezierSegment::getPointsCount() [pure virtual]

Retrieves the number of points in the polyquadratic Bézier segment.

Returns

uint32. Returns the size of the points list.

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

- idompathgeometry.h

8.225 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 bool getCalculatorAsPostScriptStream (IInputStreamPtr &stream)=0
  
  *Get the PostScript Calculator function as a PostScript Procedure.*

• virtual bool getCalculator (IEDLObjectPtr &psObj)=0
  
  *Get the PostScript Calculator function as an executable PS object.*

• virtual bool evaluate (float *inputValues, float *outputValues)=0
  
  *Evaluate the input through the function and return the result.*

• virtual bool evaluate (int *inputValues, float *outputValues)=0
  
  *Evaluate the input through the function and return the result.*

Static Public Member Functions

• static const CClassID & classID ()
  
  *Retrieves class id of IDOM.*

Additional Inherited Members

8.225.1 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.

8.225.2 Member Function Documentation

8.225.2.1 classID()

    static const CClassID & IDOMPostScriptCalculatorFunction::classID () [inline], [static]

    Retrieves class id of IDOM.

    Returns

    CClassID Class id of the element

8.225.2.2 evaluate() [1/2]

    virtual bool IDOMPostScriptCalculatorFunction::evaluate ( float * inputValues, 
                                                            float * outputValues ) [pure virtual]

    Evaluate the input through the function and return the result.
8.225 IDOMPostScriptCalculatorFunction Class Reference

Parameters

| inputValues | An array of floats that are input into the function. The size of the array must be the same as the required number of inputValues |
| outputValues | An array of floats that are the result of evaluating the input through the function. The size of the array must be the same as the required number of inputValues |

Returns

bool Returns true on success. False if there is an error evaluating the input.

Implements IDOMFunction.

8.225.2.3 evaluate() [2/2]

virtual bool IDOMPostScriptCalculatorFunction::evaluate ( int * inputValues, float * outputValues ) [pure virtual]

Evaluate the input through the function and return the result.

Parameters

| inputValues | An array of integers that are input into the function. The size of the array must be the same as the required number of inputValues |
| outputValues | An array of floats that are the result of evaluating the input through the function. The size of the array must be the same as the required number of inputValues |

Returns

bool Returns true on success. False if there is an error evaluating the input.

Implements IDOMFunction.

8.225.2.4 getCalculator()

virtual bool IDOMPostScriptCalculatorFunction::getCalculator ( IEDLObjectPtr & psObj ) [pure virtual]

Get the PostScript Calculator function as an executable PS object.

Parameters

| psArray | A reference to receive the function |

Generated by Doxygen
Returns

bool Returns true on success.

8.225.2.5 getCalculatorAsPostScriptStream()

virtual bool IDOMPostScriptCalculatorFunction::getCalculatorAsPostScriptStream (IInputStreamPtr & stream) [pure virtual]

Get the PostScript Calculator function as a PostScript Procedure.

Parameters

<table>
<thead>
<tr>
<th>procStream</th>
<th>A reference to receive the function as an input Stream</th>
</tr>
</thead>
</table>

Returns

bool Returns true on success.

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

- idomfunction.h

8.226 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

8.226.1 Detailed Description

IDOMPrintTicket interface.

8.226.2 Member Function Documentation

8.226.2.1 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
8.227  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 bool 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 bool 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 bool 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 bool setRadiusY (double r)=0
  
  Sets the y-radius of the gradient brush ellipse.

- virtual bool createShading (IEDLClassFactory ∗pFactory, IDOMShadingPatternType3BrushPtr &ptr←ShadingBrush, const FRect &fillArea, bool useFirstStopColorSpace=false)=0
  
  Create a Type3 Shading Pattern brush from this radial brush.

- virtual IDOMRadialGradientBrushPtr getSimplifiedGradient (IEDLClassFactory ∗pFactory, const FRect &fillArea)=0
  
  Create a simplified radial gradient brush, where any repeat or reflect pad mode is converted to simple padding by repeating gradient stops as required. The resulting brush will be generated to fill the given rectangle. An exception of type IEDLError is thrown on failure.

Static Public Member Functions

- static const CClassID & classID ()
  
  Retrieves class id of IDOM.

Additional Inherited Members

8.227.1 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 of 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.

8.227.2 Member Function Documentation

Generated by Doxygen
8.227.2.1 classID()

static const CClassID & IDOMRadialGradientBrush::classID ( ) [inline], [static]

Retrieves class id of IDOM.

Returns

CClassID Class id of the element

8.227.2.2 createShading()

virtual bool IDOMRadialGradientBrush::createShading ( IEDLClassFactory * pFactory, IDOMShadingPatternType3BrushPtr & ptrShadingBrush, 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pFactory</td>
<td>The EDL class factory</td>
</tr>
<tr>
<td>ptrShadingBrush</td>
<td>reference to receive the new shading brush.</td>
</tr>
<tr>
<td>rect</td>
<td>The rectangular area that needs to be filled by the resulting brush.</td>
</tr>
<tr>
<td>useFirstStopColorSpace</td>
<td>If true the colour space in the first stop will be used for the shading. Otherwise either sRGB or scRGB will be used depending on the interpolation mode.</td>
</tr>
</tbody>
</table>

Returns

bool. Returns true on success, false if the call fails.

8.227.2.3 getCenter()

virtual const FPoint& IDOMRadialGradientBrush::getCenter ( ) const [pure virtual]

Retrieves the center point of the gradient brush ellipse.

Returns

FPoint. Returns the center point of the gradient brush ellipse.
8.227.4  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. Returns the origin point of the radial gradient.

8.227.5  getRadiusX()

virtual double IDOMRadialGradientBrush::getRadiusX() const [pure virtual]

Retrieves the x-radius of the gradient brush ellipse.

Returns

double. Returns the x-radius of the gradient brush ellipse.

8.227.6  getRadiusY()

virtual double IDOMRadialGradientBrush::getRadiusY() const [pure virtual]

Retrieves the y-radius of the gradient brush ellipse.

Returns

double. Returns the y-radius of the gradient brush ellipse.

8.227.7  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. An exception of type IEDLError is thrown on failure.

Parameters

<table>
<thead>
<tr>
<th>pFactory</th>
<th>The EDL class factory</th>
</tr>
</thead>
<tbody>
<tr>
<td>rect</td>
<td>The rectangular area that needs to be filled by the resulting brush.</td>
</tr>
</tbody>
</table>

Generated by Doxygen
Returns

IDOMRadialGradientBrushPtr The simplified brush, or this brush if no action was required.

8.227.2.8 setCenter()

virtual bool IDOMRadialGradientBrush::setCenter (const FPoint & pt) [pure virtual]

Sets the center point of the gradient brush ellipse.

Parameters

pt Reference parameter to receive the center point of the gradient brush ellipse.

Returns

bool. Returns true on success, false if the call fails.

8.227.2.9 setGradientOrigin()

virtual bool IDOMRadialGradientBrush::setGradientOrigin (const FPoint & pt) [pure virtual]

Sets the origin point of the radial gradient, corresponding to the gradient stop with an offset of 0.0.

Parameters

pt The origin point of the radial gradient.

Returns

bool. Returns true on success, false if the call fails.

8.227.2.10 setRadiusX()

virtual bool IDOMRadialGradientBrush::setRadiusX (double r) [pure virtual]

Sets the x-radius of the gradient brush ellipse.
Parameters

\[ r \] The x-radius of the ellipse

Returns

bool. Returns true on success, false if the call fails.

### 8.227.2.11 setRadiusY()

virtual bool IDOMRadialGradientBrush::setRadiusY (double r) [pure virtual]

Sets the y-radius of the gradient brush ellipse.

Parameters

\[ r \] The y-radius of the gradient brush ellipse.

Returns

bool. Returns true on success, false if the call fails.

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

- idombrush.h

**8.228 IDOMRawDataFile Class Reference**

IDOMRawDataFile interface.

#include <idomresources.h>
Inheritance diagram for IDOMRawDataFile:

```
IROObject

IEDLObject

IDOMResource

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.*

**Static Public Member Functions**

- static const **CClassID & classID ()**
  
  *Retrieves class id of IDOM.*

**Additional Inherited Members**

8.228.1 Detailed Description

IDOMRawDataFile interface.

8.228.2 Member Function Documentation
8.228.1 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

8.229 IDOMRawImage Class Reference

Interface to a class representing a raw image.

#include <idomimageresource.h>

Inheritance diagram for IDOMRawImage:

```
IRCOBJECT
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>IEDLObject</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>------</td>
</tr>
</tbody>
</table>
IDOMResource| IDOMHashable

IDOMImage

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.

Static Public Member Functions

• static const CClassID & classID ()
  Retrieves class id of IDOMRawImage.

• static EDL_API IDOMImagePtr createWriterAndImage (const ISessionPtr &session, IImageFrameWriterPtr &frame, const IDOMColorSpacePtr &colorSpace, int32 width, int32 height, uint8 bitsPerComponent=8, double xResolution=96.0, double yResolution=96.0, ImageExtraChannel_t extraChannel=ImageExtraChannel_t::None, const IInputStreamPtr &inStream=IInputStreamPtr(), const IOutputStreamPtr &outStream=IOutputStreamPtr())
  Create an IDOMRawImage and frame that can be used to populate same.

Additional Inherited Members

8.229.1 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 colour space.

8.229.2 Member Function Documentation

8.229.2.1 classID()

static const CClassID & IDOMRawImage::classID () [inline], [static]

Retrieves class id of IDOMRawImage.

Returns

  CClassID Class id of the element
8.229.2.2 createWriterAndImage()

```cpp
static EDL_API IDOMImagePtr IDOMRawImage::createWriterAndImage (  
    const ISessionPtr & session,  
    IImageFrameWriterPtr & frame,  
    const IDOMColorSpacePtr & colorSpace,  
    int32 width,  
    int32 height,  
    uint8 bitsPerComponent = 8,  
    double xResolution = 96.0,  
    double yResolution = 96.0,  
    ImageExtraChannel_t extraChannel = ImageExtraChannel_None,  
    const IInputStreamPtr & inStream = IInputStreamPtr(),  
    const IOutputStreamPtr & outStream = IOutputStreamPtr() ) [static]
```

Create an IDOMRawImage and frame that can be used to populate same.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>session</code></td>
<td>The session to use</td>
</tr>
<tr>
<td><code>frame</code></td>
<td>On exit, this is populated with a frame ready to receive image data via frame-&gt;writeScanLine(). Use frame-&gt;flushData() to complete the encoding process.</td>
</tr>
<tr>
<td><code>colorSpace</code></td>
<td>The color space to use. Any valid color space may be used for IDOMRawImages.</td>
</tr>
<tr>
<td><code>width</code></td>
<td>The width of the image, in pixels.</td>
</tr>
<tr>
<td><code>height</code></td>
<td>The height of the image, in pixels.</td>
</tr>
<tr>
<td><code>bitsPerComponent</code></td>
<td>The bits per component to use. 1, 2, 4, 8, 12 and 16 bits per sample are supported.</td>
</tr>
<tr>
<td><code>xResolution</code></td>
<td>The x resolution, in pixels-per-inch.</td>
</tr>
<tr>
<td><code>yResolution</code></td>
<td>The y resolution, in pixels-per-inch.</td>
</tr>
<tr>
<td><code>extraChannel</code></td>
<td>The type of extra channel, if provided.</td>
</tr>
<tr>
<td><code>inStream</code></td>
<td>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.</td>
</tr>
<tr>
<td><code>outStream</code></td>
<td>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.</td>
</tr>
</tbody>
</table>

**Returns**

`IDOMImagePtr` The resulting image. Not valid until the frame is flushed.

8.229.2.3 getSynthetic()

```cpp
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
8.230  IDOMRecombineAlpha Class Reference

Similar to IDOMRecombineImage, but instead combines an image comprising the colour 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 colour image. Images with Indexed colour spaces will be converted to the base spaces.

#include <idomimageresource.h>

8.230.1  Detailed Description

Similar to IDOMRecombineImage, but instead combines an image comprising the colour 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 colour image. Images with Indexed colour spaces will be converted to the base spaces.

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

- idomimageresource.h

8.231  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 bool setStream (const IInputStreamPtr &stream)
  
  *This image type does not allow direct access to the underlying streams.*

Static Public Member Functions

- static const CClassID & classID ()
  
  *Retrieves class id of IDOMRecombineImage.*

Additional Inherited Members

8.231.1 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.

8.231.2 Member Function Documentation

8.231.2.1 classID()

static const CClassID& IDOMRecombineImage::classID () [inline], [static]

*Retrieves class id of IDOMRecombineImage.*

Returns

  CClassID Class id of the element

8.231.2.2 getStream()

virtual bool IDOMRecombineImage::getStream ( IInputStreamPtr & stream ) const [inline], [virtual]

*This image type does not allow direct access to the underlying streams.*
Parameters

| stream | A smart pointer to the stream |

Returns

bool Always false

Implements IDOMResource.

8.231.2.3 setStream()

virtual bool IDOMRecombineImage::setStream (const IInputStreamPtr & stream) [inline], [virtual]

This image type does not allow direct access to the underlying streams.

Parameters

| stream | A smart pointer to the stream |

Returns

bool Always false

Implements IDOMResource.

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

- idomimageresource.h

8.232 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>
Public Member Functions

- virtual bool getStream (IInputStreamPtr &stream) const =0
  Retrieves the resource stream.
- virtual bool setStream (const IInputStreamPtr &stream)=0
  Sets the resource stream for the node.
- virtual bool getStreamLength (uint64 &length) const =0
  Retrieves the stream length, if it is available.
- virtual bool getUri (EDLSysString &uri) const =0
  Retrieves the resource URI.
- virtual bool setUri (const EDLSysString &uri)=0
  Sets the resource URI.

Additional Inherited Members

8.232.1 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.

8.232.2 Member Function Documentation

8.232.2.1 getStream()

virtual bool IDOMResource::getStream (  
    IInputStreamPtr & stream ) const  [pure virtual]

Retrieves the resource stream.
Parameters

| stream | Reference parameter to receive a smart pointer to the resource stream. |

Returns

bool. Returns true on success, false if the call fails.

Implemented in IDOMFilteredImage, IDOMCompositeImage, and IDOMRecombineImage.

8.232.2.2 getStreamLength()

virtual bool IDOMResource::getStreamLength (  
    uint64 & length ) const [pure virtual]

Retrieves the stream length, if it is available.

Parameters

| length | Reference parameter to receive the stream length. |

Returns

bool. Returns true on success, false if the call fails.

8.232.2.3 getUri()

virtual bool IDOMResource::getUri (  
    EDLSysString & uri ) const [pure virtual]

Retrieves the resource URI.

Parameters

| uri | Reference parameter to receive the resource URI. |

Returns

bool. Returns true on success, false if the call fails.
8.232.4 setStream()

virtual bool IDOMResource::setStream (const IInputStreamPtr & stream) [pure virtual]

Sets the resource stream for the node.

Parameters

| stream | Smart pointer to the new resource stream. |

Returns

bool. Returns true on success, false if the call fails.

Implemented in IDOMFilteredImage, IDOMCompositImage, and IDOMRecombineImage.

8.232.5 setUri()

virtual bool IDOMResource::setUri (const EDL::SysString & uri) [pure virtual]

Sets the resource URI.

Parameters

| uri | The new resource URI. |

Returns

bool. Returns true on success, false if the call fails.

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

- idomresources.h

8.233 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:

![Inheritance Diagram]

Classes

- **class Data**
  
  *Initialization data.*

Public Member Functions

- virtual bool get (const EDLSysString &name, IEDLObjectPtr &element) const =0
  
  _Retrieves a smart pointer to a resource stored in the resource dictionary._

- virtual bool put (const EDLSysString &name, const IEDLObjectPtr &element)=0
  
  _Registers a resource element in the resource dictionary._

Static Public Member Functions

- static const CClassID & classID ()
  
  _Retrieves the class id of IDOMResourceDictionary._

Additional Inherited Members

8.233.1 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.
8.233.2 Member Function Documentation

8.233.2.1 classID()

static const CClassID & IDOMResourceDictionary::classID ( ) [inline], [static]

Retrieves the class id of IDOMResourceDictionary.

Returns

CClassID. Returns the class id of the element

8.233.2.2 get()

virtual bool IDOMResourceDictionary::get (  
    const EDLSysString & name,  
    IEDLObjectPtr & element ) const [pure virtual]

Retrieves a smart pointer to a resource stored in the resource dictionary.

Parameters

<table>
<thead>
<tr>
<th>name</th>
<th>The name of the resource to be retrieved.</th>
</tr>
</thead>
<tbody>
<tr>
<td>element</td>
<td>Reference parameter to receive a smart pointer to the resource (as an IEDLObject pointer).</td>
</tr>
</tbody>
</table>

Returns

bool. Returns true on success, false if the call fails.

8.233.2.3 put()

virtual bool IDOMResourceDictionary::put (  
    const EDLSysString & name,  
    const IEDLObjectPtr & element ) [pure virtual]

Registers a resource element in the resource dictionary.

Parameters

<table>
<thead>
<tr>
<th>name</th>
<th>The name of the resource to add.</th>
</tr>
</thead>
<tbody>
<tr>
<td>element</td>
<td>Smart pointer to an IEDLObject representing the resource.</td>
</tr>
</tbody>
</table>
Returns

bool. Returns true on success, false if the call fails.

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

- idomresources.h

### 8.234 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.

```cpp
#include <idomfunction.h>
```

Inheritance diagram for IDOMSampledFunction:

Classes

- class Data
  
  *Initialization data.*

Public Types

- enum eInterpolationMethod

  *Enum for interpolation types.*
Public Member Functions

- virtual bool getTableDimension (int inputNum, uint32 &dimension)=0
  Get the dimension of the sample table for a given input number.
- virtual bool getBitsPerSample (uint32 &bitsPerSample)=0
  Get the bits per sample for the sample table.
- virtual bool getInterpolationMethod (eInterpolationMethod &interpolationMethod)=0
  Get the interpolation method used for sample lookup.
- virtual bool getInputEncode (int inputNum, float &low, float &high)=0
  Get the input encode range for a given input to the function.
- virtual bool getOutputDecode (int outputNum, float &low, float &high)=0
  Get the output range for a given input to the function.
- virtual const CEDLVector<uint8> &getTableData ()=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.
- virtual bool evaluate (float *inputValues, float *outputValues)=0
  Evaluate the input through the function and return the result.
- virtual bool evaluate (int *inputValues, float *outputValues)=0
  Evaluate the input through the function and return the result.

Static Public Member Functions

- static const CClassID & classID ()
  Retrieves class id of IDOM.

Additional Inherited Members

8.234.1 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.

8.234.2 Member Function Documentation

8.234.2.1 classID()

static const CClassID & IDOMSampledFunction::classID () [inline], [static]

Retrieves class id of IDOM.

Returns

CClassID class id of the element

8.234.2.2 evaluate() [1/2]

virtual bool IDOMSampledFunction::evaluate (  
    float * inputValues,  
    float * outputValues ) [pure virtual]

Evaluate the input through the function and return the result.
Parameters

<table>
<thead>
<tr>
<th>inputValues</th>
<th>An array of floats that are input into the function. The size of the array must be the same as the required number of inputValues</th>
</tr>
</thead>
<tbody>
<tr>
<td>outputValues</td>
<td>An array of floats that are the result of evaluating the input through the function. The size of the array must be the same as the required number of inputValues</td>
</tr>
</tbody>
</table>

Returns

bool Returns true on success. False if there is an error evaluating the input.

Implements IDOMFunction.

8.234.2.3 evaluate()

```cpp
evirtual bool IDOMSampledFunction::evaluate (  
    int * inputValues,  
    float * outputValues ) [pure virtual]
```

Evaluate the input through the function and return the result.

Parameters

<table>
<thead>
<tr>
<th>inputValues</th>
<th>An array of integers that are input into the function. The size of the array must be the same as the required number of inputValues</th>
</tr>
</thead>
<tbody>
<tr>
<td>outputValues</td>
<td>An array of floats that are the result of evaluating the input through the function. The size of the array must be the same as the required number of inputValues</td>
</tr>
</tbody>
</table>

Returns

bool Returns true on success. False if there is an error evaluating the input.

Implements IDOMFunction.

8.234.2.4 getBitsPerSample()

```cpp
virtual bool IDOMSampledFunction::getBitsPerSample (  
    uint32 & bitsPerSample ) [pure virtual]
```

Get the bits per sample for the sample table.

Parameters

| bitsPerSample | A reference to receive the value. |
8.234.2.5 getInputEncode()

virtual bool IDOMSampledFunction::getInputEncode ( 
    int inputNum, 
    float & low, 
    float & high ); [pure virtual]

Get the input encode range for a given input to the function.

Parameters

<table>
<thead>
<tr>
<th>inputNum</th>
<th>The 0-indexed input number</th>
</tr>
</thead>
<tbody>
<tr>
<td>low</td>
<td>A reference to receive the low encode bound for the given inputNum</td>
</tr>
<tr>
<td>high</td>
<td>A reference to receive the high encode bound for the given inputNum</td>
</tr>
</tbody>
</table>

Returns

bool Returns true on success

8.234.2.6 getInterpolationMethod()

virtual bool IDOMSampledFunction::getInterpolationMethod ( 
    eInterpolationMethod & interpolationMethod ); [pure virtual]

Get the interpolation method used for sample lookup.

Parameters

| interpolationMethod | A reference to receive the value. |

Returns

bool Returns true on success

8.234.2.7 getOutputDecode()

virtual bool IDOMSampledFunction::getOutputDecode ( 
    int outputNum, 
    Generated by Doxygen
float & low,
float & high ) [pure virtual]

Get the output range for a given input to the function.
Parameters

<table>
<thead>
<tr>
<th>outputNum</th>
<th>The 0-indexed output number</th>
</tr>
</thead>
<tbody>
<tr>
<td>low</td>
<td>A reference to receive the low decode bound for the given outputNum</td>
</tr>
<tr>
<td>high</td>
<td>A reference to receive the high decode bound for the given outputNum</td>
</tr>
</tbody>
</table>

Returns

bool Returns true on success.

8.234.2.8 getTableData()

virtual const CELVector<uint8>& IDOMSampledFunction::getTableData () [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.

8.234.2.9 getTableDimension()

virtual bool IDOMSampledFunction::getTableDimension ( int inputNum,
                                                        uint32 & dimension ) [pure virtual]

Get the dimension of the sample table for a given input number.

Parameters

<table>
<thead>
<tr>
<th>inputNum</th>
<th>The 0-indexed input number</th>
</tr>
</thead>
<tbody>
<tr>
<td>dimension</td>
<td>A reference to receive the dimension.</td>
</tr>
</tbody>
</table>

Returns

bool Returns true on success

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

- idomfunction.h

Generated by Doxygen
8.235  IDOMSecurityInfo Class Reference

Base DOM security class.

#include <idomsecurity.h>

Inheritance diagram for IDOMSecurityInfo:

```
+------------------+
| IRCObject        |
+------------------+
    ^              |
    |                |
    |                |
+------------------+
| IEDLObject       |
+------------------+
    |                |
    v                |
+------------------+
| IDOMSecurityInfo |
```

Additional Inherited Members

8.235.1 Detailed Description

Base DOM security class.

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

- idomsecurity.h

8.236  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 int getShadingType ()=0
  Retrieves the shading type.
- virtual bool getBBox (float *bbox)=0
  Retrieves the bounding box for the shade.
- virtual bool setBBox (float *bbox)=0
  Sets the bounding box for the shade.
- virtual bool setBackgroundColor (const IDOMColorPtr &color)=0
  Sets the background color to use before painting the shade.
- virtual bool getBackgroundColor (IDOMColorPtr &color)=0
  Gets the background color to use before painting the shade. If the return is false then the background is not painted before applying the shade.
- virtual bool setColorSpace (const IDOMColorSpacePtr &colorSpace)=0
  Sets the colorspace to use for painting the shading.
- virtual bool getColorSpace (IDOMColorSpacePtr &colorSpace)=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. Throws exceptions of type IEDLError on failure.
- virtual bool setAntiAlias (bool antiAlias)=0
  Sets anti aliasing flag.
- virtual bool getAntiAlias (bool &antiAlias)=0
  Gets anti aliasing flag.

Additional Inherited Members

8.236.1 Detailed Description

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

8.236.2 Member Function Documentation

8.236.2.1 getAntiAlias()

virtual bool IDOMShadingPatternBrush::getAntiAlias ( bool & antiAlias ) [pure virtual]

Gets anti aliasing flag.

Parameters

- **antiAlias**: Reference to anti aliasing indicator
Returns

bool Returns true on success.

8.236.2.2 getBackgroundColor()

virtual bool IDOMShadingPatternBrush::getBackgroundColor ( 
    IDOMColorPtr & color ) [pure virtual]

Gets the background color to use before painting the shade. If the return is false then the background is not painted before applying the shade.

Parameters

| color | The smart pointer to the color object. |

Returns

bool Returns true on success.

8.236.2.3 getBBox()

virtual bool IDOMShadingPatternBrush::getBBox ( 
    float * bbox ) [pure virtual]

Retrieves the bounding box for the shade.

Parameters

| bbox | Reference to bbox |

Returns

bool Returns true on success.

8.236.2.4 getColorSpace()

virtual bool IDOMShadingPatternBrush::getColorSpace ( 
    IDOMColorSpacePtr & colorSpace ) [pure virtual]

Gets the colorspace object to be used when painting the shading.
Parameters

| colorSpace | The smart pointer to the colorspace object. |

Returns

bool Returns true on success.

8.236.2.5 getShadingType()

virtual int IDOMShadingPatternBrush::getShadingType() [pure virtual]

Retrieves the shading type.

Returns

int Returns 1 to 7 for the different PS shading types

8.236.2.6 setAntiAlias()

virtual bool IDOMShadingPatternBrush::setAntiAlias(bool antiAlias) [pure virtual]

Sets anti aliasing flag.

Parameters

| antiAlias | New value of anti aliasing flag |

Returns

bool Returns true on success.

8.236.2.7 setBackgroundColor()

virtual bool IDOMShadingPatternBrush::setBackgroundColor(const IDOMColorPtr & color) [pure virtual]

Sets the background color to use before painting the shade.
Parameters

| **color** | The smart pointer to the color object |

Returns

bool Returns true on success.

### 8.236.2.8 setBBox()

virtual bool IDOMShadingPatternBrush::setBBox ( float * bbox ) [pure virtual]

Sets the bounding box for the shade.

Parameters

| **bbox** | Reference to the bounding box |

Returns

bool Returns true on success.

### 8.236.2.9 setColorSpace() [1/2]

virtual bool IDOMShadingPatternBrush::setColorSpace ( const IDOMColorSpacePtr & colorSpace ) [pure virtual]

Sets the colorspace to use for painting the shade.

Parameters

| **colorSpace** | The smart pointer to the colorspace object |

Returns

bool Returns true on success.

### 8.236.2.10 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. Throws exceptions of type IEDLError on failure.

Parameters

<table>
<thead>
<tr>
<th>pFactory</th>
<th>A pointer to an EDL class factory</th>
</tr>
</thead>
<tbody>
<tr>
<td>colorSpace</td>
<td>The smart pointer to the color space. Must not be a complex or composite color space such as Indexed or DeviceN.</td>
</tr>
<tr>
<td>intent</td>
<td>The rendering intent to use for conversion.</td>
</tr>
<tr>
<td>bpc</td>
<td>Black point compensation treatment. If in double, use eBPCDefault.</td>
</tr>
</tbody>
</table>

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

- idombrush.h

8.237 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 bool setDomain (float *domain)=0
  *Sets the domain range.*
- virtual bool getDomain (float *domain)=0
  *Gets the domain range.*
- virtual bool setMatrix (FMatrix &matrix)=0
  *Sets the Type 1 shade matrix.*
- virtual bool getMatrix (FMatrix &matrix)=0
  *Gets the Type 1 shade matrix.*
- virtual bool setFunction (IDOMFunctionPtr &function)=0
  *Sets the shade function.*
- virtual bool getFunction (IDOMFunctionPtr &function)=0
  *Gets the shade function object.*
Static Public Member Functions

- static const CClassID & classID ()
  
  Retrieves class id of IDOM.

Additional Inherited Members

8.237.1 Detailed Description

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

8.237.2 Member Function Documentation

8.237.2.1 classID()

static const CClassID & IDOMShadingPatternType1Brush::classID () [inline], [static]

Retrieves class id of IDOM.

Returns

  CClassID Class id of the element

8.237.2.2 getDomain()

virtual bool IDOMShadingPatternType1Brush::getDomain ( 
  float * domain ) [pure virtual]

Gets the domain range.

Parameters

  \textit{domain} | Array of 4 floats

Returns

  bool Returns true on success.
8.237.2.3 getFunction()

virtual bool IDOMShadingPatternType1Brush::getFunction (  
    IDOMFunctionPtr & function )  [pure virtual]

Gets the shade function object.

Parameters

| function | Reference to a pointer to a function object |

Returns

dbool Returns true on success.

8.237.2.4 getMatrix()

virtual bool IDOMShadingPatternType1Brush::getMatrix (  
    FMatrix & matrix )  [pure virtual]

Gets the Type 1 shade matrix.

Parameters

| matrix | Reference to matrix object |

Returns

dbool Returns true on success.

8.237.2.5 setDomain()

virtual bool IDOMShadingPatternType1Brush::setDomain (  
    float * domain )  [pure virtual]

Sets the domain range.

Parameters

| domain | Array of 4 floats |

Returns

dbool Returns true on success.
8.237.2.6 setFunction()

virtual bool IDOMShadingPatternType1Brush::setFunction (IDOMFunctionPtr & function) [pure virtual]

Sets the shade function.

Parameters

| function | Pointer to a function object |

Returns

bool Returns true on success.

8.237.2.7 setMatrix()

virtual bool IDOMShadingPatternType1Brush::setMatrix (FMatrix & matrix) [pure virtual]

Sets the Type 1 shade matrix.

Parameters

| matrix | Reference to a matrix object |

Returns

bool Returns true on success.

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

- idombrush.h

8.238 IDOMShadingPatternType2Brush Class Reference

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

#include <idombrush.h>
Inheritance diagram for IDOMShadingPatternType2Brush:

```
+--- IRCOBJECT
    +--- IEDLOBJECT
    |    +--- IDOMBRUSH
    |         +--- IDOMTransformableBrush
    |                     +--- IDOMShadingPatternBrush
    |                                    +--- IDOMShadingPatternType2Brush
```

Classes

- class **Data**
  
  *Initialization data.*

Public Member Functions

- virtual bool **setDomain** (float *domain)=0
  
  *Sets the domain range.*

- virtual bool **getDomain** (float *domain)=0
  
  *Gets the domain range.*

- virtual bool **setCoords** (float *coord)=0
  
  *Sets the coords.*

- virtual bool **getCoords** (float *coord)=0
  
  *Gets the shade coordinate.*

- virtual bool **setExtend** (bool extendX, bool extendY)=0
  
  *Sets the shading Extend flag to represent whether or not to extend beyond the start and end points for each axis.*
• virtual bool **getExtend**(bool &extendX, bool &extendY)=0  
  *Gets the shading Extend flag to represent whether or not to extend beyond the start and end points for each axis.*

• virtual bool **setFunction**(IDOMFunctionPtr &function)=0  
  *Sets the shade function.*

• virtual bool **getFunction**(IDOMFunctionPtr &function)=0  
  *Gets the shade function object.*

• virtual bool **getEquivalentSimpleBrush** (IEDLClassFactory ∗pFactory, IDOMBrushPtr &simple, uint32 max←Samples=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.*

**Static Public Member Functions**

• static const CClassID & **classID**( ) [inline], [static]  
  *Retrieves class id of IDOM.*

**Additional Inherited Members**

8.238.1 Detailed Description

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

8.238.2 Member Function Documentation

8.238.2.1 **classID**( )

static const CClassID & IDOMShadingPatternType2Brush::classID ( ) [inline], [static]

Retrieves class id of IDOM.

**Returns**

CClassID Class id of the element

8.238.2.2 **getCoords**( )

virtual bool IDOMShadingPatternType2Brush::getCoords (  
  float * coord ) [pure virtual]

Gets the shade coordinate.
Parameters

| coord | An array of 4 floats that represent coordinates |

Returns

bool Returns true on success.

8.238.2.3 getDomain()

virtual bool IDOMShadingPatternType2Brush::getDomain (  
    float * domain) [pure virtual]

Gets the domain range.

Parameters

| domain | Array of 4 floats |

Returns

bool Returns true on success.

8.238.2.4 getEquivalentSimpleBrush()

virtual bool IDOMShadingPatternType2Brush::getEquivalentSimpleBrush (  
    IEDLClassFactory * pFactory,  
    IDOMBrushPtr & simple,  
    uint32 maxSamples = 255 ) [pure virtual]

Gets an equivalent Linear gradient or Visual brush, which may involve sampling the functions. A Visual brush is required if the shading pattern has a bounding box, in order to apply a clip. This is intended to be used to generate a brush that can be expressed directly in XPS.

Parameters

| pFactory | A pointer to an EDL class factory |
| simple   | A reference to receive the equivalent gradient |
| maxSamples | The maximum number of samples that should be taken from the function to generate gradient stops. |

Returns

bool Returns true on success.
8.238.2.5 getExtend()

virtual bool IDOMShadingPatternType2Brush::getExtend (  
    bool & extendX,  
    bool & extendY ) [pure virtual]

Gets the shading Extend flag to represent whether or not to extend beyond the start and end points for each axis.

Parameters

<table>
<thead>
<tr>
<th>extendX</th>
<th>Reference to bool parameter representing the flags for the x axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>extendY</td>
<td>Reference to bool parameter representing the flags for the y axis</td>
</tr>
</tbody>
</table>

Returns

bool Returns true on success.

8.238.2.6 getFunction()

virtual bool IDOMShadingPatternType2Brush::getFunction (  
    IDOMFunctionPtr & function ) [pure virtual]

Gets the shade function object.

Parameters

| function | Reference to a pointer to a function object |

Returns

bool Returns true on success.

8.238.2.7 setCoords()

virtual bool IDOMShadingPatternType2Brush::setCoords (  
    float * coord ) [pure virtual]

Sets the coords.

Parameters

| coord | An array of 4 floats that represent coordinates |

Generated by Doxygen
8.238.2.8 setDomain()

virtual bool IDOMShadingPatternType2Brush::setDomain (  
    float * domain  ) [pure virtual]

Sets the domain range.

Parameters

| domain   | Array of 4 floats |

Returns

bool Returns true on success.

8.238.2.9 setExtend()

virtual bool IDOMShadingPatternType2Brush::setExtend (  
    bool extendX,  
    bool extendY  ) [pure virtual]

Sets the shading Extend flag to represent whether or not to extend beyond the start and end points for each axis.

Parameters

| extendX | parameter representing the flags for the x axis |
| extendY | parameter representing the flags for the y axis |

Returns

bool Returns true on success.

8.238.2.10 setFunction()

virtual bool IDOMShadingPatternType2Brush::setFunction (  
    IDOMFunctionPtr & function  ) [pure virtual]

Sets the shade function.
8.239 IDOMShadingPatternType3Brush Class Reference

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

#include <idombrush.h>

Inheritance diagram for IDOMShadingPatternType3Brush:

Parameters

| function | Pointer to a function object |

Returns

bool Returns true on success.

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

- idombrush.h
Classes

• class Data
  *Initialization data.*

Public Member Functions

• virtual bool setDomain (float *domain)=0
  *Sets the domain range.*
• virtual bool getDomain (float *domain)=0
  *Gets the domain range.*
• virtual bool setCoords (float *coord)=0
  *Sets the coords.*
• virtual bool getCoords (float *coord)=0
  *Gets the shade coordinate.*
• virtual bool setExtend (bool extendX, bool extendY)=0
  *Sets the shading Extend flag to represent whether or not to extend beyond the start and end points for each axis.*
• virtual bool getExtend (bool &extendX, bool &extendY)=0
  *Gets the shading Extend flag to represent whether or not to extend beyond the start and end points for each axis.*
• virtual bool setFunction (IDOMFunctionPtr &function)=0
  *Sets the shade function.*
• virtual bool getFunction (IDOMFunctionPtr &function)=0
  *Gets the shade function object.*
• virtual bool getEquivalentSimpleBrush (IEDLClassFactory *pFactory, IDOMBrushPtr &simple, 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.*

Static Public Member Functions

• static const CClassID & classID ()
  *Retrieves class id of IDOM.*

Additional Inherited Members

8.239.1 Detailed Description

IDOMShadingPatternType3Brush provides a way of representing a PS style type 2 shading pattern.
8.239.2.1 classID()

static const CClassID & IDOMShadingPatternType3Brush::classID () [inline], [static]

Retrieves class id of IDOM.

Returns

   CClassID Class id of the element

8.239.2.2 getCoords()

virtual bool IDOMShadingPatternType3Brush::getCoords ( 
   float * coord ) [pure virtual]

Gets the shade coordinate.

Parameters

| coord | An array of 6 floats that represent coordinates |

Returns

   bool Returns true on success.

8.239.2.3 getDomain()

virtual bool IDOMShadingPatternType3Brush::getDomain ( 
   float * domain ) [pure virtual]

Gets the domain range.

Parameters

| domain | Array of 4 floats |

Returns

   bool Returns true on success.
8.239.2.4  getEquivalentSimpleBrush()

virtual bool IDOMShadingPatternType3Brush::getEquivalentSimpleBrush (  
    IEDLClassFactory * pFactory,  
    IDOMBrushPtr & simple,  
    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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pFactory</td>
<td>A pointer to an EDL class factory</td>
</tr>
<tr>
<td>simple</td>
<td>A reference to receive the equivalent gradient</td>
</tr>
<tr>
<td>maxSamples</td>
<td>The maximum number of samples that should be taken from the function to generate gradient stops.</td>
</tr>
</tbody>
</table>

Returns

bool Returns true on success.

8.239.2.5  getExtend()

virtual bool IDOMShadingPatternType3Brush::getExtend (  
    bool & extendX,  
    bool & extendY  
) [pure virtual]

Gets the shading Extend flag to represent whether or not to extend beyond the start and end points for each axis.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>extendX</td>
<td>Reference to boolean parameter representing the flags for the x axis</td>
</tr>
<tr>
<td>extendY</td>
<td>Reference to boolean parameter representing the flags for the y axis</td>
</tr>
</tbody>
</table>

Returns

bool Returns true on success.

8.239.2.6  getFunction()

virtual bool IDOMShadingPatternType3Brush::getFunction (  
    IDOMFunctionPtr & function  
) [pure virtual]

Gets the shade function object.
Parameters

| function | Reference to a pointer to a function object |

Returns

bool Returns true on success.

8.239.2.7 setCoords()

virtual bool IDOMShadingPatternType3Brush::setCoords (float * coord) [pure virtual]

Sets the coords.

Parameters

| coord | An array of 6 floats that represent coordinates |

Returns

bool Returns true on success.

8.239.2.8 setDomain()

virtual bool IDOMShadingPatternType3Brush::setDomain (float * domain) [pure virtual]

Sets the domain range.

Parameters

| domain | Array of 4 floats |

Returns

bool Returns true on success.

8.239.2.9 setExtend()

virtual bool IDOMShadingPatternType3Brush::setExtend (bool extendX, bool extendY) [pure virtual]
Sets the shading Extend flag to represent whether or not to extend beyond the start and end points for each axis.

**Parameters**

<table>
<thead>
<tr>
<th>extendX</th>
<th>Boolean parameter representing the flags for the x axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>extendY</td>
<td>Boolean parameter representing the flags for the y axis</td>
</tr>
</tbody>
</table>

**Returns**

`bool` Returns true on success.

---

### 8.239.2.10 setFunction()

```cpp
virtual bool IDOMShadingPatternType3Brush::setFunction ( 
    IDOMFunctionPtr & function ) [pure virtual]
```

Sets the shade function.

**Parameters**

| function  | Pointer to a function object |

**Returns**

`bool` Returns true on success.

---

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

- `idombrush.h`

---

### 8.240 IDOMShadingPatternType4567Brush Class Reference

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

```cpp
#include <idombrush.h>
```
Inheritance diagram for IDOMShadingPatternType4567Brush:

```
    IRCObject
    |      |
    |      |<
    IEDLObject
    |      |
    |      |<
    IDOMBrush
    |      |
    |      |<
    IDOMTransformableBrush
    |      |
    |      |<
    IDOMShadingPatternBrush
    |      |
    |      |<
    IDOMShadingPatternType4567Brush
```

Classes

- class **Data**
  
  *Initialization data.*

Public Member Functions

- virtual void **setShadingType** (int shadingType)=0
  
  *Sets the shading type.*

- virtual bool **setDataSource** (IEDLObjectPtr &dataSource)=0
  
  *Sets the data source property.*

- virtual bool **getDataSource** (IEDLObjectPtr &dataSource)=0
  
  *Gets the data source property.*

- virtual bool **setBitsPerCoordinate** (int bitsPerCoordinate)=0
  
  *Sets the bits per coordinate.*

- virtual bool **getBitsPerCoordinate** (int &bitsPerCoordinate)=0
Gets the bits per coordinate parameter.

- virtual bool setBitsPerComponent (int bitsPerComponent)=0
  Sets the bits per component.
- virtual bool getBitsPerComponent (int &bitsPerComponent)=0
  Gets the bits per component parameter.
- virtual bool setBitsPerFlag (int bitsPerFlag)=0
  Sets the bits per flag.
- virtual bool getBitsPerFlag (int &bitsPerFlag)=0
  Gets the bits per flag parameter.
- virtual bool setVerticesPerRow (int verticesPerRow)=0
  Sets the vertices per row flag.
- virtual bool getVerticesPerRow (int &verticesPerRow)=0
  Gets the vertices per row parameter.
- virtual bool setDecode (int decodeSize, float *decodeArray)=0
  Sets the decode array.
- virtual bool getDecodeSize (int &decodeSize)=0
  Gets the size of the decode array.
- virtual bool getDecode (float *decodeArray)=0
  Gets the decode array.
- virtual bool setFunction (IDOMFunctionPtr &function)=0
  Sets the shade function.
- virtual bool getFunction (IDOMFunctionPtr &function)=0
  Gets the shade function object.

Static Public Member Functions

- static const CClassID & classID ()
  Retrieves class id of IDOM.

Additional Inherited Members

8.240.1 Detailed Description

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

8.240.2 Member Function Documentation

8.240.2.1 classID()

static const CClassID& IDOMShadingPatternType4567Brush::classID ( ) [inline], [static]

Retrieves class id of IDOM.

Returns

  CClassID class id of the element
8.240.2.2 getBitsPerComponent()

virtual bool IDOMShadingPatternType4567Brush::getBitsPerComponent (  
    int & bitsPerComponent ) [pure virtual]

Gets the bits per component parameter.

Parameters

| reference | to an int which receives the parameter |

Returns

bool Returns true on success.

8.240.2.3 getBitsPerCoordinate()

virtual bool IDOMShadingPatternType4567Brush::getBitsPerCoordinate (  
    int & bitsPerCoordinate ) [pure virtual]

Gets the bits per coordinate parameter.

Parameters

| reference | to an int which receives the parameter |

Returns

bool Returns true on success.

8.240.2.4 getBitsPerFlag()

virtual bool IDOMShadingPatternType4567Brush::getBitsPerFlag (  
    int & bitsPerFlag ) [pure virtual]

Gets the bits per flag parameter.

Parameters

| reference | to an int which receives the parameter |

Returns

bool Returns true on success.
8.240.2.5 getDataSource()

virtual bool IDOMShadingPatternType4567Brush::getDataSource (IEDLObjectPtr & dataSource) [pure virtual]

Gets the data source property.

Parameters

| reference | to a smart pointer which receives a data source object |

Returns

bool Returns true on success.

8.240.2.6 getDecode()

virtual bool IDOMShadingPatternType4567Brush::getDecode (float * decodeArray) [pure virtual]

Gets the decode array.

Parameters

| array | of floats, it is assumed that the array is big enough |

Returns

bool Returns true on success.

8.240.2.7 getDecodeSize()

virtual bool IDOMShadingPatternType4567Brush::getDecodeSize (int & decodeSize) [pure virtual]

Gets the size of the decode array.

Parameters

| reference | to an int which receives number of floats in the array |
8.240.2.8 getFunction()

virtual bool IDOMShadingPatternType4567Brush::getFunction (  
    IDOMFunctionPtr & function ) [pure virtual]

Gets the shade function object.

Parameters

| Reference | to a pointer to a function object |

Returns

bool Returns true on success.

8.240.2.9 getVerticesPerRow()

virtual bool IDOMShadingPatternType4567Brush::getVerticesPerRow (  
    int & verticesPerRow ) [pure virtual]

Gets the vertices per row parameter.

Parameters

| reference | to an int which receives the parameter |

Returns

bool Returns true on success.

8.240.2.10 setBitsPerComponent()

virtual bool IDOMShadingPatternType4567Brush::setBitsPerComponent (  
    int bitsPerComponent ) [pure virtual]

Sets the bits per component.
Parameters

**bitsPerComponent**

Returns

bool Returns true on success.

8.240.2.11 setBitsPerCoordinate()

virtual bool IDOMShadingPatternType4567Brush::setBitsPerCoordinate ( int bitsPerCoordinate ) [pure virtual]

Sets the bits per coordinate.

Parameters

**bitsPerCoordinate**

Returns

bool Returns true on success.

8.240.2.12 setBitsPerFlag()

virtual bool IDOMShadingPatternType4567Brush::setBitsPerFlag ( int bitsPerFlag ) [pure virtual]

Sets the bits per flag.

Parameters

**bitsPerFlag**

Returns

bool Returns true on success.

8.240.2.13 setDataSource()

virtual bool IDOMShadingPatternType4567Brush::setDataSource ( IEDLObjectPtr & dataSource ) [pure virtual]
Sets the data source property.
Parameters

*reference* to data source object

Returns

`bool` Returns true on success.

### 8.240.2.14 setDecode()

```cpp
virtual bool IDOMShadingPatternType567Brush::setDecode ( int decodeSize, float * decodeArray ) [pure virtual]
```

Sets the decode array.

Parameters

*number* of floats in the array

*array* of floats

Returns

`bool` Returns true on success.

### 8.240.2.15 setFunction()

```cpp
virtual bool IDOMShadingPatternType567Brush::setFunction ( IDOMFunctionPtr & function ) [pure virtual]
```

Sets the shade function.

Parameters

*Pointer* to a function object

Returns

`bool` Returns true on success.
8.240.2.16 setShadingType()

virtual void IDOMShadingPatternType4567Brush::setShadingType ( int shadingType ) [pure virtual]

Sets the shading type.

Parameters

| an integer which represents the PS style shading type (one of 4, 5, 6, 7) |

Returns

bool Returns true on success.

8.240.2.17 setVerticesPerRow()

virtual bool IDOMShadingPatternType4567Brush::setVerticesPerRow ( int verticesPerRow ) [pure virtual]

Sets the vertices per row flag.

Parameters

| verticesPerRow property |

Returns

bool Returns true on success.

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

- idombrush.h

8.241 IDOMShape Class Reference

Interface to an IDOMShape.

#include <idomshape.h>

Generated by Doxygen
Inheritance diagram for IDOMShape:

```
  IRCObject
    |
    v
  IEDLObject
    |
    v
IDOMShape
```

**Classes**

- class Data
  
  *Initialization data.*

**Public Member Functions**

- virtual bool `getAsImage` (IDOMImagePtr &image, ISessionPtr &session, bool bUseTempFileForImage) const =0
  
  *Return in image an 1-bit-per-pixel DeviceGray image representation of this shape.*

- virtual bool `getBounds` (FRect &bounds) const =0
  
  *Get the bounds (in pixels) of the shape.*

- virtual float `getResolution` () const =0
  
  *Detect 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 bool `unite` (const IDOMShapePtr &ptrShape)=0
  
  *Unite this shape with another.*

- virtual bool `intersect` (const IDOMShapePtr &ptrShape)=0
  
  *Reduce this shape to its intersection with another shape.*

- virtual bool `difference` (const IDOMShapePtr &ptrShape)=0
  
  *Reduce this shape by subtracting another shape.*

- 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.*
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.

Additional Inherited Members

8.241.1 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.

8.241.2 Member Function Documentation

8.241.2.1 classID()

static const CClassID& IDOMShape::classID ( ) [inline], [static]

Retrieves class id of IDOMShape.

Returns

  CClassID Class id of the element

8.241.2.2 completelyContainsShape()

virtual bool IDOMShape::completelyContainsShape ( const IDOMShapePtr & ptrShape ) const [pure virtual]

Detect if another shape fits completely inside this shape.
Parameters

| in  | ptrShape | The shape to test. |

Returns

**bool** True if the given shape fits completely in this shape.

8.241.2.3 difference()

```cpp
virtual bool IDOMShape::difference {
    const IDOMShapePtr & ptrShape } [pure virtual]
```

Reduce this shape by subtracting another shape.

Parameters

| in  | ptrShape | The shape to subtract. |

Returns

**bool** True on success, false if the call fails.

8.241.2.4 getAsImage()

```cpp
virtual bool IDOMShape::getAsImage {
    IDOMImagePtr & image,
    ISessionPtr & session,
    bool bUseTempFileForImage ) const [pure virtual]
```

Return in **image** an 1-bit-per-pixel DeviceGray image representation of this shape.

Parameters

| out | image | A reference to an IDOMImage; the caller should ensure this is NULL, as this method does the work of populating it with real content. |
| in  | session | The current session. |
| in  | bUseTempFileForImage | If true, the method creates a temporary file (the lifecycle of which EDL 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

**bool** True on success, false if the call fails.

### 8.241.2.5 getBounds()

```cpp
virtual bool IDOMShape::getBounds (FRect & bounds) const [pure virtual]
```

Get the bounds (in pixels) of the shape.

**Parameters**

- **out bounds** A reference to receive the bounds of the shape.

Returns

**bool** True on success, false if the call fails.

### 8.241.2.6 getIsEmpty()

```cpp
virtual bool IDOMShape::getIsEmpty () const [pure virtual]
```

Detect an empty shape.

Returns

**bool** True if the shape is empty.

### 8.241.2.7 getIsRect()

```cpp
virtual bool IDOMShape::getIsRect () const [pure virtual]
```

Detect if a shape is a rectangle.

Returns

**bool** True if the shape is rectangular.
virtual float IDOMShape::getResolution ( ) const [pure virtual]

Get the resolution of the shape.

Returns

float Returns the resolution of the shape.

virtual bool IDOMShape::intersect ( const IDOMShapePtr & ptrShape ) [pure virtual]

Reduce this shape to its intersection with another shape.

Parameters

| in  | ptrShape | The shape with which to intersect. |

Returns

bool True on success, false if the call fails.

virtual bool IDOMShape::intersects ( const IDOMShapePtr & ptrShape ) const [pure virtual]

Detect if this shape intersects with another.

Parameters

| in  | ptrShape | The shape to test with. |

Returns

bool True if the shapes intersect, false otherwise.
8.241.2.11 isEqualTo()

virtual bool IDOMShape::isEqualTo (const IDOMShapePtr & ptrShape ) const [pure virtual]

Check this shape for equality to another shape.

Parameters

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>ptrShape</td>
<td>The shape to compare.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td></td>
<td>True if the shapes are equivalent.</td>
</tr>
</tbody>
</table>

8.241.2.12 unite()

virtual bool IDOMShape::unite (const IDOMShapePtr & ptrShape ) [pure virtual]

Unite this shape with another.

Parameters

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>ptrShape</td>
<td>The shape with which to unite.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td></td>
<td>True on success, false if the call fails.</td>
</tr>
</tbody>
</table>

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

- idomshape.h

8.242 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 Member Functions

- virtual eSoftMaskType getSoftMaskType ()=0
  Retrieves the soft mask type.
- virtual bool getGroup (IDOMTransparencyGroupPtr &group)=0
  Retrieves the transparency group used for the mask.
- virtual bool getBackColor (IDOMColorPtr &backdropColor)=0
  Retrieves the color used for luminosity backdrop composition.
- virtual bool getTransferFunction (IDOMFunctionPtr &transfer)=0
  Retrieves the function used for luminosity used for deriving mask values.
Static Public Member Functions

- static EDL_API IDOMSoftMaskBrushPtr create (IEDLClassFactory *pFactory, const IDOMTransparencyGroupPtr &group, eSoftMaskType type, const FMatrix &renderTransform=FMMatrix(), 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

8.242.1 Detailed Description

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.

8.242.2 Member Function Documentation

8.242.2.1 classID()

static const CClassID& IDOMSoftMaskBrush::classID ( ) [inline], [static]

Retrieves class id of IDOM.

Returns

  CClassID Class id of the element

8.242.2.2 create()

static EDL_API IDOMSoftMaskBrushPtr IDOMSoftMaskBrush::create (IEDLClassFactory * pFactory, const IDOMTransparencyGroupPtr & group, eSoftMaskType type, const FMatrix & renderTransform = FMMatrix(), const IDOMColorPtr & backdropColor = IDOMColorPtr(), const IDOMFunctionPtr & transfer = IDOMFunctionPtr ( ) ) [static]

Simplified creator for a soft mask brush. Throws an IEDLError on failure.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pFactory</td>
<td>The factory to use.</td>
</tr>
<tr>
<td>group</td>
<td>The group to form the content of the soft mask.</td>
</tr>
<tr>
<td>type</td>
<td>The type of the soft mask.</td>
</tr>
<tr>
<td>renderTransform</td>
<td>The desired render transform.</td>
</tr>
<tr>
<td>backdropColor</td>
<td>The backdrop colour (must have the same colour space as the group). Optional.</td>
</tr>
<tr>
<td>transfer</td>
<td>The soft mask transfer function. Optional.</td>
</tr>
</tbody>
</table>
Returns
IDOMSoftMaskBrushPtr The new brush.

8.242.2.3 getBackColor()

virtual bool IDOMSoftMaskBrush::getBackColor ( 
    IDOMColorPtr & backdropColor ) [pure virtual]

Retrieves the color used for luminosity backdrop composition.

Parameters
backdropColor Smart pointer to receive the color.

Returns
bool Returns true on success. Will return false if there is no color, or if the mask is eAlpha type.

8.242.2.4 getGroup()

virtual bool IDOMSoftMaskBrush::getGroup ( 
    IDOMTransparencyGroupPtr & group ) [pure virtual]

Retrieves the transparency group used for the mask.

Parameters
group Smart pointer to receive the group.

Returns
bool Returns true on success.

8.242.2.5 getSoftMaskType()

virtual eSoftMaskType IDOMSoftMaskBrush::getSoftMaskType ( ) [pure virtual]

Retrieves the soft mask type.

Returns
eSoftMaskType The soft mask type.
virtual bool IDOMSoftMaskBrush::getTransferFunction (  
    IDOMFunctionPtr & transfer ) [pure virtual]

Retrieves the function used for luminosity used for deriving mask values.

Parameters

    transfer  Smart pointer to receive the function.

Returns

    bool  Returns true on success. Will return false if there is no function.

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

- idombrush.h

### 8.243 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 bool getColor (IDOMColorPtr &color) const =0
  
  *Retrieves the color value of the solid color brush.*

- virtual bool setColor (const IDOMColorPtr &color)=0
  
  *Sets color value of the solid color brush.*

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 const CClassID & classID ()
  
  *Retrieves class id of IDOMIDOMSolidColorBrush.*

Additional Inherited Members

8.243.1 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.

8.243.2 Member Function Documentation

8.243.2.1 classID()

static const CClassID & IDOMSolidColorBrush::classID ( ) [inline], [static]

*Retrieves class id of IDOMIDOMSolidColorBrush.*

Returns

CClassID. Class id of the element

8.243.2.2 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

<table>
<thead>
<tr>
<th><strong>pFactory</strong></th>
<th>The EDL Class factory to use.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>color</strong></td>
<td>The color to use.</td>
</tr>
<tr>
<td><strong>opacity</strong></td>
<td>The alpha to use.</td>
</tr>
</tbody>
</table>

Returns

IDOMSolidColorBrushPtr The new brush.

### 8.243.2.3 getColor()

```cpp
virtual bool IDOMSolidColorBrush::getColor ( IDOMColorPtr & color ) const [pure virtual]
```

Retrieves the color value of the solid color brush.

Parameters

| **color** | Reference parameter to receive the color value |

Returns

bool. Returns true on success, false if the call fails.

### 8.243.2.4 setColor()

```cpp
virtual bool IDOMSolidColorBrush::setColor ( const IDOMColorPtr & color ) [pure virtual]
```

Sets color value of the solid color brush.

Parameters

| **color** | The color value |

Returns

bool. Returns true on success, false if the call fails.

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

- idombrush.h
8.244  IDOMStandardPDFSecurityInfo Class Reference

Represents security information from PDF Standard encryption handler.

#include <idomsecurity.h>

Inherits IDOMPDFSecurityInfo.

Public Types

- enum ePermissionsFlags {
  ePrintAllowed = 0x04UL, eEditingAllowed = 0x08UL, eCopyingAllowed = 0x10UL, eAnnotationEditingAllowed = 0x20UL,
  eFormFillingAllowed = 0x100UL, eContentAccessibilityExtractionAllowed = 0x200UL, eDocumentAssemblyAllowed = 0x400UL,
  eHighQualityPrintAllowed = 0x800UL, eEverythingAllowed = 0xfffffffcUL
}

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 is 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
  Is printing allowed?
- bool isHighQualityPrintingAllowed () const
  Is high quality printing allowed?
- bool isEditingAllowed () const
  Is modification of the contents by operations other than those specified by isEditingAnnotationsAllowed(), isFillingFormAllowed() and isDocAssemblyAllowed() allowed?
- bool isCopyingAllowed () const
  Is copying of text and graphics from the document for operations other than specified by isExtractionAllowed() al-
  lowed?
- bool isAnnotationEditingAllowed () const
  Is editing of annotations and forms allowed?
- bool isFillingFormsAllowed () const
  Is form-filling allowed?
- bool isContentAccessibilityExtractionAllowed () const
  Is extracting for content accessibility allowed?
- bool isDocumentAssemblyAllowed () const
  Is document assembly 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.
8.244.1 Detailed Description

Represents security information from PDF Standard encryption handler.

8.244.2 Member Enumeration Documentation

8.244.2.1 ePermissionsFlags

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ePrintAllowed</td>
<td>Allow printing (bit 3)</td>
</tr>
<tr>
<td>eEditingAllowed</td>
<td>Allow editing (bit 4)</td>
</tr>
<tr>
<td>eCopyingAllowed</td>
<td>Allow copying (bit 5)</td>
</tr>
<tr>
<td>eAnnotationEditingAllowed</td>
<td>Allow annotation editing (bit 6)</td>
</tr>
<tr>
<td>eFormFillingAllowed</td>
<td>Allow form filling (bit 9)</td>
</tr>
<tr>
<td>eContentAccessibilityExtractionAllowed</td>
<td>Allow content extraction in support of accessibility (bit 10)</td>
</tr>
<tr>
<td>eDocumentAssemblyAllowed</td>
<td>Allow document assembly (bit 11)</td>
</tr>
<tr>
<td>eHighQualityPrintAllowed</td>
<td>Allow low-resolution printing (bit 12)</td>
</tr>
<tr>
<td>eEverythingAllowed</td>
<td>All operations permitted</td>
</tr>
</tbody>
</table>

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

- `idomsecurity.h`

8.245 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.

```c
#include <idomfunction.h>
```
Inheritance diagram for IDOMStitchingFunction:

```plaintext
IRObject

IDOMStitchingFunction
```

Classes

- class Data
  
  *Initialization data.*

Public Member Functions

- virtual bool getNumFunctions (uint32 &numFunctions)=0
  
  *Get the number of functions that are stitched.*

- virtual bool getFunctionAtIndex (uint32 index, IDOMFunctionPtr &function)=0
  
  *Get the function for a given function index.*

- virtual const CEDLVector<float> & getBoundsVector ()=0
  
  *Get a reference to the bounds vector for this function.*

- virtual const CEDLVector<float> & getEncodeVector ()=0
  
  *Get a reference to the encode vector for this function.*

- virtual bool evaluate (float ∗inputValues, float ∗outputValues)=0
  
  *Evaluate the input through the function and return the result.*

- virtual bool evaluate (int ∗inputValues, float ∗outputValues)=0
  
  *Evaluate the input through the function and return the result.*

Static Public Member Functions

- static const CClassID & classID ()
  
  *Retrieves class id of IDOM.*
Additional Inherited Members

8.245.1 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.

8.245.2 Member Function Documentation

8.245.2.1 classID()

static const CClassID & IDOMStitchingFunction::classID ( ) [inline], [static]

Retrieves class id of IDOM.

Returns

CClassID class id of the element

8.245.2.2 evaluate() [1/2]

virtual bool IDOMStitchingFunction::evaluate ( float * inputValues, float * outputValues ) [pure virtual]

Evaluate the input through the function and return the result.

Parameters

| inputValues | An array of floats that are input into the function. The size of the array must be the same as the required number of inputValues |
| outputValues | An array of floats that are the result of evaluating the input through the function. The size of the array must be the same as the required number of inputValues |

Returns

bool Returns true on success. False if there is an error evaluating the input.

Implements IDOMFunction.
8.245.2.3 evaluate() [2/2]

```cpp
virtual bool IDOMStitchingFunction::evaluate (  
    int * inputValues,  
    float * outputValues ) [pure virtual]
```

Evaluate the input through the function and return the result.

**Parameters**

| inputValues | An array of integers that are input into the function. The size of the array must be the same as the required number of inputValues |
| outputValues | An array of floats that are the result of evaluating the input through the function. The size of the array must be the same as the required number of inputValues |

**Returns**

`bool` Returns true on success. False if there is an error evaluating the input.

Implements `IDOMFunction`.

8.245.2.4 getBoundsVector()

```cpp
virtual const CEDLVector<float>& IDOMStitchingFunction::getBoundsVector ( ) [pure virtual]
```

Get a reference to the bounds vector for this function.

**Returns**

`CEDLVector<float>` Reference to a constant `CEDLBuffer<float>` containing the bounds.

8.245.2.5 getEncodeVector()

```cpp
virtual const CEDLVector<float>& IDOMStitchingFunction::getEncodeVector ( ) [pure virtual]
```

Get a reference to the encode vector for this function.

**Returns**

`CEDLVector<float>` Reference to a constant `CEDLBuffer<float>` containing the encode.
8.245.2.6 getFunctionAtIndex()

virtual bool IDOMStitchingFunction::getFunctionAtIndex (  
    uint32 index,  
    IDOMFunctionPtr & function ) [pure virtual]

Get the function for a given function index.

index A zero based index for the desired function. function A reference to receive the function.

Returns
   bool Returns true on success

8.245.2.7 getNumFunctions()

virtual bool IDOMStitchingFunction::getNumFunctions (  
    uint32 & numFunctions ) [pure virtual]

Get the number of functions that are stitched.

Parameters

| numFunctions | A reference to receive the number of functions. |

Returns
   bool Returns true on success

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

- idomfunction.h

8.246 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.

Additional Inherited Members

8.246.1 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.

8.246.2 Member Enumeration Documentation

8.246.2.1 eTargetType

enum IDOMTarget::eTargetType
  An enumeration of target types.
## 8.246 IDOMTarget Class Reference

### Enumerator

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eExternal</td>
<td>(PDF URI Action) Resolve a uniform resource identifier.</td>
</tr>
<tr>
<td>eInternal</td>
<td>XPS internal target.</td>
</tr>
<tr>
<td>ePage</td>
<td>XPS page target.</td>
</tr>
<tr>
<td>ePageRect</td>
<td>(PDF Goto Action) Go to a destination in the current document.</td>
</tr>
<tr>
<td>eActionGoToR</td>
<td>(&quot;Go-to remote&quot;) Go to a destination in another document.</td>
</tr>
<tr>
<td>eActionGoToE</td>
<td>(&quot;Go-to embedded&quot;; PDF 1.6) Go to a destination in an embedded file.</td>
</tr>
<tr>
<td>eActionLaunch</td>
<td>Launch an application, usually to open a file.</td>
</tr>
<tr>
<td>eActionThread</td>
<td>Begin reading an article thread.</td>
</tr>
<tr>
<td>eActionSound</td>
<td>(PDF 1.2) Play a sound.</td>
</tr>
<tr>
<td>eActionMovie</td>
<td>(PDF 1.2) Play a movie.</td>
</tr>
<tr>
<td>eActionHide</td>
<td>(PDF 1.2) Set an annotation's Hidden flag.</td>
</tr>
<tr>
<td>eActionNamed</td>
<td>(PDF 1.2) Execute an action predefined by the viewer application.</td>
</tr>
<tr>
<td>eActionSubmitForm</td>
<td>(PDF 1.2) Send data to a uniform resource locator.</td>
</tr>
<tr>
<td>eActionResetForm</td>
<td>(PDF 1.2) Set fields to their default values.</td>
</tr>
<tr>
<td>eActionImportData</td>
<td>(PDF 1.2) Import field values from a file.</td>
</tr>
<tr>
<td>eActionJavaScript</td>
<td>(PDF 1.3) Execute a JavaScript script.</td>
</tr>
<tr>
<td>eActionSetOCGState</td>
<td>(PDF 1.5) Set the states of optional content groups.</td>
</tr>
<tr>
<td>eActionRendition</td>
<td>(PDF 1.5) Controls the playing of multimedia content.</td>
</tr>
<tr>
<td>eActionTrans</td>
<td>(PDF 1.5) Updates the display of a document, using a transition dictionary.</td>
</tr>
<tr>
<td>eActionGoTo3DView</td>
<td>(PDF 1.6) Set the current view of a 3D annotation</td>
</tr>
<tr>
<td>eActionArray</td>
<td>Array of actions.</td>
</tr>
</tbody>
</table>

### 8.246.3 Member Function Documentation

#### 8.246.3.1 getTargetType()

```cpp
virtual eTargetType IDOMTarget::getTargetType ( ) const [pure virtual]
```

Retrieves the target type.

Returns
- `eTargetType` The target type

Implemented in `IDOMActionArray`, `IDOMActionLaunch`, `IDOMPageRectTarget`, `IDOMPageTarget`, `IDOMInternalTarget`, `IDOMExternalTarget`.

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

- `idomtarget.h`
8.247  IDOMTIFFImage Class Reference

IDOMTIFFImage interface.

#include <idomimageresource.h>

Inheritance diagram for IDOMTIFFImage:

```
  IRObject
     
  IEDLObject
     
  IDOMHashable  IDOMResource
                
  IDOMImage
                
  IDOMTIFFImage
```

Public Types

- enum eTIFFCompression { , eTCNone }
  Available TIFF compression schemes.

Static Public Member Functions

- static EDL_API IDOMTIFFImagePtr create (IEDLClassFactory *pFactory, const IInputStreamPtr &stream)
  Create a TIFF Image resource with the given TIFF stream. Throws an IEDLError on failure.

- static EDL_API void encode (const ISessionPtr &pSession, const IDOMImagePtr &image, const IOutputStreamPtr &stream, eTIFFCompression scheme=eTCAuto)
  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. Throws an IEDLError on failure.

- static EDL_API void encode (const ISessionPtr &pSession, const IImageFramePtr &frame, const IOutputStreamPtr &stream, eTIFFCompression scheme=eTCAuto)
  Encode the contents of an ImageFrame 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. Throws an IEDLError on failure.

- static const CClassID & classID ()
  Retrieves class id of IDOMTIFFImage.
Additional Inherited Members

8.247.1 Detailed Description

IDOMTIFFImage interface.

8.247.2 Member Enumeration Documentation

8.247.2.1 eTIFFCompression

enum IDOMTIFFImage::eTIFFCompression

Available TIFF compression schemes.

Enumerator

| eTCNone | Choose an appropriate scheme for the incoming image. |

8.247.3 Member Function Documentation

8.247.3.1 classID()

static const CClassID & IDOMTIFFImage::classID ( ) [inline], [static]

Retrieves class id of IDOMTIFFImage.

Returns

CClassID Class id of the element

8.247.3.2 create()

static EDL_API IDOMTIFFImagePtr IDOMTIFFImage::create ( 
    IEDLClassFactory * pFactory,
    const IInputStreamPtr & stream ) [static]

Create a TIFF Image resource with the given TIFF stream. Throws an IEDLError on failure.
Parameters

| pFactory | The EDL Class factory to use. |
| stream   | The stream containing the JPEG image. |

Returns

**IDOMImagePtr** The new image.

### 8.247.3.3 encode() [1/2]

static EDL_API void IDOMTIFFImage::encode (  
const ISessionPtr & pSession,  
const IDOMImagePtr & image,  
const IOutputStreamPtr & stream,  
eTIFFCompression scheme = eTCAuto ) [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. Throws an IEDLError on failure.

Parameters

| pSession | The relevant EDL session |
| image    | The image to be encoded |
| scheme   | The TIFF compression scheme to be used. |
| stream   | The stream to use to store the image data. |

### 8.247.3.4 encode() [2/2]

static EDL_API void IDOMTIFFImage::encode (  
const ISessionPtr & pSession,  
const IImageFramePtr & frame,  
const IOutputStreamPtr & stream,  
eTIFFCompression scheme = eTCAuto ) [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. Throws an IEDLError on failure.

Parameters

| pSession | The relevant EDL session |
| frame    | The frame providing the source image data |
| scheme   | The TIFF compression scheme to be used. |
| stream   | The stream to use to store the image data. |
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 int getPatternType ()=0
  Retrieves the pattern type.
- virtual bool getBBox (float *bbox)=0
  Retrieves the bounding box for the pattern.
- virtual bool setBBox (float *bbox)=0
  Sets the bounding box for the pattern.
- virtual bool getTilingStep (float &xstep, float &ystep)=0
  Retrieves the tiling step for the pattern.
- virtual bool setTilingStep (float xstep, float ystep)=0
  Sets the tiling step for the pattern.
- virtual bool getPaintType (int &paintType)=0
  Gets the paint type for the pattern.
- virtual bool setPaintType (int paintType)=0
  Sets the paint type for the pattern.
- virtual bool setTilingType (int tilingType)=0
  Sets the tiling type for the pattern.
- virtual bool getTilingType (int &tilingType)=0
  Gets the tiling type for the pattern.
- virtual bool setPatternColor (IDOMColorPtr &color)=0
  Sets the pattern color for uncolored pattern (paint type is 2)
- virtual bool getPatternColor (IDOMColorPtr &color) const =0
  Gets the pattern color for uncolored pattern (paint type is 2)
- virtual bool getVisual (IDOMNodePtr &ptrVisual) const =0
  Retrieves smart pointer to visual (path, glyphs, canvas) node.
- virtual bool setVisual (const IDOMNodePtr &ptrVisual)=0
  Sets visual node.
- virtual bool getRenderTransform (FMatrix &matrix) const =0
  Retrieves render transform matrix.
- virtual bool setRenderTransform (const FMatrix &matrix)=0
  Sets render transform matrix.
- virtual bool getEquivalentVisualBrush (IEDLClassFactory *pFactory, IDOMVisualBrushPtr &visualBrush)=0
  Gets an equivalent IDOMVisualBrush brush. If the brush has overlapping tiles, this call will fail.

Static Public Member Functions

- static EDL_API IDOMTilingPatternBrushPtr create (IEDLClassFactory *pFactory, const IDOMNodePtr &visual, const FRect &bbox, uint32 paintType, const IDOMColorPtr &color, float xStep, float yStep, uint32 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

8.248.1 Detailed Description

IDOMTilingPatternBrush provides a way of representing a PS style tiling pattern.
8.248.2 Member Function Documentation

8.248.2.1 classID()

static const CClassID & IDOMTilingPatternBrush::classID ( ) [inline], [static]

Retrieves class id of IDOM.

Returns

CClassID Class id of the element

8.248.2.2 create()

static EDL_API IDOMTilingPatternBrushPtr IDOMTilingPatternBrush::create ( 
  IEDLClassFactory * pFactory,
  const IDOMNodePtr & visual,
  const FRect & bBox,
  uint32 paintType,
  const IDOMColorPtr & color,
  float xStep,
  float yStep,
  uint32 tilingType = 1,
  const FMatrix & renderTransform = FMatrix() ) [static]

Simplified creator for a tiling brush. Throws an IEDLError on failure.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pFactory</td>
<td>The factory to use.</td>
</tr>
<tr>
<td>visual</td>
<td>The node or node tree representing the pattern's content.</td>
</tr>
<tr>
<td>bBox</td>
<td>the bounding box of the pattern's content.</td>
</tr>
<tr>
<td>paintType</td>
<td>The paint type, either 1 (coloured) or 2 (uncoloured)</td>
</tr>
<tr>
<td>color</td>
<td>The color to use. Must be NULL for paintType 1. For paintType 2, if NULL, A DeviceGray black will be used.</td>
</tr>
<tr>
<td>xStep</td>
<td>The tiling step in X</td>
</tr>
<tr>
<td>yStep</td>
<td>The tiling step in Y</td>
</tr>
<tr>
<td>tilingType</td>
<td>The tiling type; 1 for constant spacing, 2 for no distortion, 3 for constant spacing and faster tiling.</td>
</tr>
<tr>
<td>renderTransform</td>
<td>The desired render transform.</td>
</tr>
</tbody>
</table>

Returns

IDOMTilingPatternBrushPtr The new brush.
8.248.2.3 getBBox()

virtual bool IDOMTilingPatternBrush::getBBox ( 
    float * bbox ) [pure virtual]

Retrieves the bounding box for the pattern.

Parameters

bbox Reference to bbox

Returns

bool Returns true on success.

8.248.2.4 getEquivalentVisualBrush()

virtual bool IDOMTilingPatternBrush::getEquivalentVisualBrush ( 
    IEDLClassFactory * pFactory, 
    IDOMVisualBrushPtr & visualBrush ) [pure virtual]

Gets an equivalent IDOMVisualBrush brush. If the brush has overlapping tiles, this call will fail.

Parameters

pFactory A pointer to an EDL class factory
visualBrush A reference to receive the equivalent visual brush

Returns

bool Returns true on success.

8.248.2.5 getPaintType()

virtual bool IDOMTilingPatternBrush::getPaintType ( 
    int & paintType ) [pure virtual]

Gets the paint type for the pattern.

Parameters

paintType The paint type (1 for colored tiling pattern, 2 for uncolored)
Returns

bool Returns true on success.

### 8.248.2.6 getPatternColor()

```cpp
virtual bool IDOMTilingPatternBrush::getPatternColor ( 
    IDOMColorPtr & color ) const [pure virtual]
```

Gets the pattern color for uncolored pattern (paint type is 2)

**Parameters**

- `color` Reference to receive the color.

Returns

bool Returns true on success.

### 8.248.2.7 getPatternType()

```cpp
virtual int IDOMTilingPatternBrush::getPatternType ( ) [pure virtual]
```

Retrieves the pattern type.

Returns

int Returns pattern type, 1 for tiling pattern

### 8.248.2.8 getRenderTransform()

```cpp
virtual bool IDOMTilingPatternBrush::getRenderTransform ( 
    FMatrix & matrix ) const [pure virtual]
```

Retrieves render transform matrix.

**Parameters**

- `matrix` Render transform matrix
Concerning the Class Documentation provided:

8.248.2.9 getTilingStep()

virtual bool IDOMTilingPatternBrush::getTilingStep(
    float & xstep,
    float & ystep) [pure virtual]

Retrieves the tiling step for the pattern.

Parameters

<table>
<thead>
<tr>
<th>xstep</th>
<th>Reference to xstep</th>
</tr>
</thead>
<tbody>
<tr>
<td>ystep</td>
<td>Reference to ystep</td>
</tr>
</tbody>
</table>

Returns

bool Returns true on success.

8.248.2.10 getTilingType()

virtual bool IDOMTilingPatternBrush::getTilingType(
    int & tilingType) [pure virtual]

Gets the tiling type for the pattern.

Parameters

| tilingType | The tiling type (1 for constant spacing, 2 for no distortion, 3 for constant spacing and faster tiling) |

Returns

bool Returns true on success.

8.248.2.11 getVisual()

virtual bool IDOMTilingPatternBrush::getVisual(
    IDOMNodePtr & ptrVisual) const [pure virtual]

Retrieves smart pointer to visual (path, glyphs, canvas) node.
Parameters

| ptrVisual | Smart pointer to the visual node |

Returns

bool Returns true on success

8.248.2.12 setBBox()

virtual bool IDOMTilingPatternBrush::setBBox (float * bbox) [pure virtual]

Sets the bounding box for the pattern.

Parameters

| bbox | Reference to bbox |

Returns

bool Returns true on success.

8.248.2.13 setPaintType()

virtual bool IDOMTilingPatternBrush::setPaintType (int paintType) [pure virtual]

Sets the paint type for the pattern.

Parameters

| paintType | The paint type (1 for colored tiling pattern, 2 for uncolored) |

Returns

bool Returns true on success.

8.248.2.14 setPatternColor()

virtual bool IDOMTilingPatternBrush::setPatternColor (IDOMColorPtr & color) [pure virtual]

Generated by Doxygen
Sets the pattern color for uncolored pattern (paint type is 2)
Parameters

| color | Reference to the desired color to use. |

Returns

bool Returns true on success.

8.248.2.15  

setRenderTransform()

```
virtual bool IDOMTilingPatternBrush::setRenderTransform (  
    const FMatrix & matrix ) [pure virtual]
```

Sets render transform matrix.

Parameters

| matrix | Render transform matrix |

Returns

bool Returns true on success

Implements IDOMTransformableBrush.

8.248.2.16  

setTilingStep()

```
virtual bool IDOMTilingPatternBrush::setTilingStep (  
    float xstep,  
    float ystep ) [pure virtual]
```

Sets the tiling step for the pattern.

Parameters

<table>
<thead>
<tr>
<th>xstep</th>
<th>Reference to xstep</th>
</tr>
</thead>
<tbody>
<tr>
<td>ystep</td>
<td>Reference to ystep</td>
</tr>
</tbody>
</table>

Returns

bool Returns true on success.
8.248.2.17  setTilingType()

virtual bool IDOMTilingPatternBrush::setTilingType (  
    int tilingType  ) [pure virtual]

Sets the tiling type for the pattern.

Parameters

| tilingType | The tiling type (1 for constant spacing, 2 for no distortion, 3 for constant spacing and faster tiling) |

Returns

bool Returns true on success.

8.248.2.18  setVisual()

virtual bool IDOMTilingPatternBrush::setVisual (  
    const IDOMNodePtr & ptrVisual  ) [pure virtual]

Sets visual node.

Parameters

| ptrVisual | Smart pointer to the visual node |

Returns

bool Returns true on success

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

- idombrush.h

8.249   IDOMTransformableBrush Class Reference

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

#include <idombrush.h>
Public Member Functions

- virtual bool getRenderTransform (FMatrix &matrix) const =0
  Retrieves the render transform matrix.
- virtual bool setRenderTransform (const FMatrix &matrix)=0
  Sets the render transform matrix.

Additional Inherited Members

8.249.1 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.

8.249.2 Member Function Documentation

8.249.2.1 getRenderTransform()

virtual bool IDOMTransformableBrush::getRenderTransform (FMatrix &matrix) const [pure virtual]

Retrieves the render transform matrix.

Parameters

| matrix | Render transform matrix |
Returns

bool. Returns true on success, false if the call fails.

Implemented in IDOMTilingPatternBrush.

8.249.2.2 setRenderTransform()

virtual bool IDOMTransformableBrush::setRenderTransform (const FMatrix & matrix) [pure virtual]

Sets the render transform matrix.

Parameters

| matrix  | Render transform matrix |

Returns

bool. Returns true on success, false if the call fails.

Implemented in IDOMTilingPatternBrush.

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

- idombrush.h

8.250 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 bool setOpacity (float opacity)=0
  
  Set the group opacity.
- virtual bool getBlendMode (eBlendMode &blendMode)=0
  
  Get the blend mode to be used for compositing this group with the backdrop.
- virtual bool setBlendMode (eBlendMode blendMode)=0
  
  Set the blend mode to be used for compositing this group with the backdrop.
- virtual bool getColorSpace (IDOMColorSpacePtr &colorSpace)=0
  
  Get the group colorspace.
• virtual bool 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 bool 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 bool 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 bool getOpacityMask (IDOMBrushPtr &ptrOpacityMask) const =0
  Retrieves smart pointer to opacity mask.

• virtual bool setOpacityMask (const IDOMBrushPtr &ptrOpacityMask)=0
  Sets opacity mask.

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.

Additional Inherited Members

8.250.1 Detailed Description

IDOMTransparencyGroup interface. Analogous to PDF Transparency groups.

8.250.2 Member Function Documentation

8.250.2.1 classID()

static const CClassID & IDOMTransparencyGroup::classID () [inline], [static]

Retrieves class id of IDOMTransparencyGroup.

Returns

CClassID class id of the element
8.250.2.2  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

<table>
<thead>
<tr>
<th>pFactory</th>
<th>The EDL class factory</th>
</tr>
</thead>
<tbody>
<tr>
<td>transform</td>
<td>The desired render transform</td>
</tr>
<tr>
<td>clip</td>
<td>The desired clip</td>
</tr>
<tr>
<td>opacity</td>
<td>The desired opacity</td>
</tr>
<tr>
<td>opacityMask</td>
<td>The desired opacity mask</td>
</tr>
<tr>
<td>blendMode</td>
<td>The desired blend mode</td>
</tr>
<tr>
<td>colorSpace</td>
<td>The desired group color space</td>
</tr>
<tr>
<td>isolated</td>
<td>Group isolated parameter</td>
</tr>
<tr>
<td>knockout</td>
<td>Group knockout parameter</td>
</tr>
</tbody>
</table>

Returns

IDOMTransparencyGroupPtr A smart pointer to the new transparency group

8.250.2.3  getBlendMode()

virtual bool IDOMTransparencyGroup::getBlendMode (  
    eBlendMode & blendMode ) [pure virtual]

Get the blend mode to be used for compositing this group with the backdrop.

Parameters

| blendMode | Reference to receive the blend mode |

Returns

bool True on success
virtual bool IDOMTransparencyGroup::getColorSpace ( 
    IDOMColorSpacePtr & colorSpace ) [pure virtual]

Get the group colorspace.

Parameters

| colorSpace | Reference to receive a smart pointer to the groups color space |

Returns

**bool** True on success

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.

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.

virtual float IDOMTransparencyGroup::getOpacity ( ) const [pure virtual]

Get the group alpha/opacity.

Returns

**float** The group opacity.

virtual bool IDOMTransparencyGroup::getOpacityMask ( 
    IDOMBrushPtr & ptrOpacityMask ) const [pure virtual]

Retrieves smart pointer to opacity mask.
Parameters

| ptrOpacityMask | Smart pointer to opacity mask |

Returns

bool True on success

8.250.2.9 setBlendMode()

virtual bool IDOMTransparencyGroup::setBlendMode ( 
  eBlendMode blendMode ) [pure virtual]

Set the blend mode to be used for compositing this group with the backdrop.

Parameters

| blendMode | The blend mode for the group. |

Returns

bool True on success

8.250.2.10 setColorSpace()

virtual bool IDOMTransparencyGroup::setColorSpace ( 
  const IDOMColorSpacePtr & colorSpace ) [pure virtual]

Set the group colorspace.

This is an optional entry in most circumstances, and may not be used in all situations. This is only used for the following situations:

- If the group is isolated (optional)
- If the group is used in a SoftMask Brush, with the soft mask using the luminosity of the group.

Not all color spaces can be used. See section 7.5.5 of the PDF 1.7 specification for further information.

Parameters

| colorSpace | The desired color space |
Returns

\textbf{bool} True on success

8.250.2.11 setIsIsolated()

\texttt{virtual bool IDOMTransparencyGroup::setIsIsolated (}
\texttt{bool isolated ) \ [pure virtual]}

Set whether the group is isolated. See section 7.5.5 of the PDF 1.7 spec for details.

Parameters

\begin{tabular}{|l|}
\hline
\texttt{isolated} & New value of isolated \\
\hline
\end{tabular}

Returns

\textbf{bool} True if the group is isolated.

8.250.2.12 setIsKnockout()

\texttt{virtual bool IDOMTransparencyGroup::setIsKnockout (}
\texttt{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

\begin{tabular}{|l|}
\hline
\texttt{knockout} & New value of knockout \\
\hline
\end{tabular}

Returns

\textbf{bool} True if the group is isolated.

8.250.2.13 setOpacity()

\texttt{virtual bool IDOMTransparencyGroup::setOpacity (}
\texttt{float opacity ) \ [pure virtual]}

Set the group opacity.
Parameters

| opacity | The desired opacity. |

Returns

bool True on success.

8.250.2.14 setOpacityMask()

virtual bool IDOMTransparencyGroup::setOpacityMask {
    const IDOMBrushPtr & ptrOpacityMask ) [pure virtual]

Sets opacity mask.

Parameters

| ptrOpacityMask | Smart pointer to brush |

Returns

bool True on success

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

- idomgroup.h

8.251 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 getGlyph (EDLSysString &glyphName, IDOMGlyphPtr &glyph)=0
  
  Retrieves a glyph of the given name, if it exists.

- virtual bool getGlyph (uint32 codePoint, IDOMGlyphPtr &glyph)=0
  
  Retrieves a glyph of the given unicode codepoint, if it exists.

- virtual bool getGlyph (IDOMGlyph::GlyphID glyphID, IDOMGlyphPtr &glyph)=0
  
  Retrieves a glyph of the given GlyphID, if it exists.

- virtual bool deleteGlyphs ()=0
  
  Delete the glyph collection from the font.

- virtual bool addGlyph (IDOMGlyphPtr &glyph)=0
  
  Adds a glyph of the given name to the font.

- virtual bool getBBox (IntRect &bbox)=0
  
  Returns font's bounding box.
Static Public Member Functions

- static const CClassID & classID ()
  Retrieves class id of IDOM.

Additional Inherited Members

8.251.1 Detailed Description

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

8.251.2 Member Function Documentation

8.251.2.1 addGlyph()

virtual bool IDOMType3Font::addGlyph ( 
    IDOMGlyphPtr & glyph ) [pure virtual]

Adds a glyph of the given name to the font.

Parameters

- glyph The glyph

Returns

- bool True on success

8.251.2.2 classID()

static const CClassID& IDOMType3Font::classID ( ) [inline], [static]

Retrieves class id of IDOM.

Returns

- CClassID class id of the element
8.251.2.3 deleteGlyphs()

virtual bool IDOMType3Font::deleteGlyphs ( ) [pure virtual]

Delete the glyph collection from the font.

Returns

bool True on success

8.251.2.4 getBBox()

virtual bool IDOMType3Font::getBBox ( 
    IntRect & bBox ) [pure virtual]

Returns font's bounding box.

Parameters

<table>
<thead>
<tr>
<th>bBox</th>
<th>Bounding box</th>
</tr>
</thead>
</table>

Returns

bool True on success

8.251.2.5 getGlyph() [1/3]

virtual bool IDOMType3Font::getGlyph ( 
    EDLSysString & glyphName, 
    IDOMGlyphPtr & glyph ) [pure virtual]

Retrieves a glyph of the given name, if it exists.

Parameters

<table>
<thead>
<tr>
<th>glyphName</th>
<th>The glyph name</th>
</tr>
</thead>
<tbody>
<tr>
<td>glyph</td>
<td>The glyph</td>
</tr>
</tbody>
</table>

Returns

bool True on success
virtual bool IDOMType3Font::getGlyph (  
    uint32 codePoint,  
    IDOMGlyphPtr & glyph ) [pure virtual]

Retrieves a glyph of the given unicode codepoint, if it exists.

Parameters

<table>
<thead>
<tr>
<th>codePoint</th>
<th>The unicode codepoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>glyph</td>
<td>The glyph</td>
</tr>
</tbody>
</table>

Returns

bool True on success

8.251.2.7  getGlyph() [3/3]

virtual bool IDOMType3Font::getGlyph (  
    IDOMGlyph::GlyphID glyphID,  
    IDOMGlyphPtr & glyph ) [pure virtual]

Retrieves a glyph of the given GlyphID, if it exists.

Parameters

<table>
<thead>
<tr>
<th>glyphID</th>
<th>The glyphID</th>
</tr>
</thead>
<tbody>
<tr>
<td>glyph</td>
<td>The glyph</td>
</tr>
</tbody>
</table>

Returns

bool True on success

8.251.2.8  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

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

- idomfont.h
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 bool 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 bool setViewBoxUnits (eViewUnits vu)=0
  Sets the viewbox units value of the image brush. Currently, only absolute units are supported.
8.252 IDOMVisualBrush Class Reference

• virtual eViewUnits getViewPortUnits () const =0

   Retrieves the viewport units value of the image brush. Currently, only absolute units are supported.

• virtual bool setViewPortUnits (eViewUnits vu)=0

   Sets the viewport units used for the image brush. Currently, only absolute units are supported.

• virtual bool getViewBox (FRect &vb) const =0

   Retrieves the viewbox rectangle.

• virtual bool setViewBox (const FRect &vb)=0

   Sets viewbox rectangle.

• virtual bool getViewPort (FRect &vp) const =0

   Retrieves the viewport rectangle.

• virtual bool setViewPort (const FRect &vp)=0

   Sets the viewport rectangle.

• virtual bool getVisual (IDOMNodePtr &ptrVisual) const =0

   Retrieves smart pointer to the visual node (path, glyphs or canvas node) used to specify the source for the visual brush.

• virtual bool setVisual (const IDOMNodePtr &ptrVisual)=0

   Sets the visual node (path, glyphs or canvas node) used to specify the source for the visual brush.

• virtual bool getEquivalentTilingBrush (IEDLClassFactory ∗pFactory, IDOMTilingPatternBrushPtr &tiling)=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. An exception of type IEDLError is thrown on failure.

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

8.252.1 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

8.252.2 Member Function Documentation
8.252.2.1  classID()

static const CClassID & IDOMVisualBrush::classID ( ) [inline], [static]

Retrieves the class id of IDOMVisualBrush.

Returns

CClassID. Returns the class id of IDOMVisualBrush.

8.252.2.2  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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pFactory</td>
<td>The factory to use.</td>
</tr>
<tr>
<td>visual</td>
<td>The node to use as the visual.</td>
</tr>
<tr>
<td>viewBox</td>
<td>The desired view box.</td>
</tr>
<tr>
<td>viewPort</td>
<td>The desired view port.</td>
</tr>
<tr>
<td>renderTransform</td>
<td>The desired render transform.</td>
</tr>
<tr>
<td>opacity</td>
<td>The desired brush opacity.</td>
</tr>
<tr>
<td>tileMode</td>
<td>The desired tile mode.</td>
</tr>
</tbody>
</table>

Returns

IDOMVisualBrushPtr The new visual brush.

8.252.2.3  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. An exception of type IEDLError is thrown on failure.
Parameters

| pFactory | A pointer to an EDL class factory |

Returns

IDOMVisualBrushPtr The equivalent visual brush, or this brush if no changes are required.

8.252.2.4 getEquivalentTilingBrush()

virtual bool IDOMVisualBrush::getEquivalentTilingBrush (IEDLClassFactory * pFactory,
IDOMTilingPatternBrushPtr & tiling) [pure virtual]

Gets an equivalent IDOMTilingPattern brush. If the receiver has a tile mode of eNoTile, this call will fail.

Parameters

| pFactory | A pointer to an EDL class factory |
| tiling   | A reference to receive the equivalent tiling brushes |

Returns

bool Returns true on success.

8.252.2.5 getTileMode()

virtual eTilingMode IDOMVisualBrush::getTileMode () const [pure virtual]

Retrieves the tiling mode value of the visual brush.

Returns

eTilingMode. Returns the tiling mode

See also

eTilingMode
virtual bool IDOMVisualBrush::getViewBox(FRect & vb) 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.

Parameters

- `vb`: The viewbox rectangle

Returns

- `bool`: Returns true on success

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`: Returns the viewbox units.

virtual bool IDOMVisualBrush::getViewPort(FRect & vp) 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.
Parameters

vp | Reference parameter to receive the viewport rectangle.

Returns

bool. Returns true on success, false if the call fails.

8.252.2.9 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. Returns the viewport units.

8.252.2.10 getVisual()

virtual bool IDOMVisualBrush::getVisual ( 
    IDOMNodePtr & ptrVisual ) const [pure virtual]

Retrieves smart pointer to the visual node (path, glyphs or canvas node) used to specify the source for the visual brush.

Parameters

ptrVisual | Smart pointer to receive the visual node.

Returns

bool. Returns true on success, false if the call fails.

8.252.2.11 setTileMode()

virtual bool IDOMVisualBrush::setTileMode ( 
    eTilingMode tm ) [pure virtual]
Sets tiling mode value of the visual brush.

Parameters

\[ tm \] The tiling mode value

Returns

bool. Returns true on success, false if the call fails.

See also

eTilingMode

8.252.12 setViewBox()

virtual bool IDOMVisualBrush::setViewBox ( 
    const FRect & vb ) [pure virtual]

Sets viewbox rectangle.

The viewbox specifies the portion of a source image or visual to be rendered to the page as a tile. The tile is then used to fill the geometry specified by the parent element according to the TileMode() function. The viewbox can specify a region larger than the image itself, including negative values. The viewbox specifies the position and dimension of the brush's source content. It is specified by four comma-separated real numbers (x, y, width, height) where width and height are non-negative.

Parameters

\[ vb \] The viewbox rectangle

Returns

bool Returns true on success

8.252.13 setViewBoxUnits()

virtual bool IDOMVisualBrush::setViewBoxUnits ( 
    eViewUnits vu ) [pure virtual]

Sets the viewbox units value of the image brush. Currently, only absolute units are supported.

See also

gViewBox()
setViewBox()
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vu</td>
<td>The viewbox units value</td>
</tr>
</tbody>
</table>

Returns

bool. Returns true on success, false if the call fails.

---

### 8.252.2.14 setViewPort()

virtual bool 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vp</td>
<td>The viewport rectangle</td>
</tr>
</tbody>
</table>

Returns

bool. Returns true on success, false if the call fails.

---

### 8.252.2.15 setViewPortUnits()

virtual bool 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vu</td>
<td>The new viewport units value</td>
</tr>
</tbody>
</table>
8.252.16  setVisual()

```cpp
virtual bool IDOMVisualBrush::setVisual {
    const IDOMNodePtr & ptrVisual } [pure virtual]
```

Sets the visual node (path, glyphs or canvas node) used to specify the source for the visual brush.

**Parameters**

| ptrVisual | Pointer to the visual node (path, glyphs or canvas node) used to specify the source for the visual brush. |

**Returns**

bool. Returns true on success, false if the call fails.

The documentation for this class was generated from the following file:
- idombrush.h

**8.253  IDOMVisualRoot Class Reference**

IDOMVisualRoot interface.

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

Inheritance diagram for IDOMVisualRoot:
Static Public Member Functions

• static const CClassID & classID ()

  Retrieves the class ID of IDOMVisualRoot.

Additional Inherited Members

8.253.1 Detailed Description

IDOMVisualRoot interface.

8.253.2 Member Function Documentation

8.253.2.1 classID()

static const CClassID & IDOMVisualRoot::classID () [inline], [static]

Retrieves the class ID of IDOMVisualRoot.

Returns

  CClassID. Returns the class ID of IDOMVisualRoot.

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

• idombrush.h

8.254 IDOMWMPI mage Class Reference

IDOMWMPI mage interface.

#include <idomimageresource.h>
Static Public Member Functions

- static const CClassID & classID ()
  
  Retrieves class id of IDOMWMPImage.

Additional Inherited Members

8.254.1 Detailed Description

IDOMWMPImage interface.

8.254.2 Member Function Documentation
8.254.2.1 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

8.255 IEDLClassFactory Class Reference

EDL Class Factory.

#include <iedlfactory.h>

Inheritance diagram for IEDLClassFactory:

```
IEVLClassFactory
```

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 registred 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 getSingleton (const CClassID &id)=0
  Creates the EDL singleton (for example FontLibrary or ColorManager)
8.255.1 Detailed Description

EDL Class Factory.

8.255.2 Member Function Documentation

8.255.2.1 createInstance()

virtual IEDLObjectPtr IEDLClassFactory::createInstance ( const CClassID & id, CClassParams * pParams = NULL ) [pure virtual]

Creates the registered class by CClassID.

Parameters

<table>
<thead>
<tr>
<th>id</th>
<th>Class GUID</th>
</tr>
</thead>
<tbody>
<tr>
<td>pParams</td>
<td>Parameters for class initialization</td>
</tr>
</tbody>
</table>

Returns

IEDLObjectPtr A smart pointer to EDLObject interface to newly created class

8.255.2.2 findNamedClass()

virtual bool IEDLClassFactory::findNamedClass ( const EDLSysString & strName, CClassID & id ) [pure virtual]

Retrieve GUID registred under the string name.

Parameters

<table>
<thead>
<tr>
<th>strName</th>
<th>Name to find</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Placeholder for result</td>
</tr>
</tbody>
</table>

Returns

bool True on success
8.255.2.3  getSingleton()

virtual IEDLObjectPtr IEDLClassFactory::getSingleton (const CClassID & id) [pure virtual]

Creates the EDL singleton (for example FontLibrary or ColorManager)

Parameters

| id | Class GUID |

Returns

IEDLObjectPtr A smart pointer to the IEDLObject

8.255.2.4  registerClass()

virtual bool IEDLClassFactory::registerClass (const CClassID & id, creatorFunc f) [pure virtual]

Register a GUID with creator function.

Parameters

| id | Class GUID |
| f | Function that creates the class |

Returns

bool True on success

8.255.2.5  registerNamedClass()

virtual bool IEDLClassFactory::registerNamedClass (const EDLSysString & strName, const CClassID & id, bool overwrite = true) [pure virtual]

Register a GUID under a string name.

Parameters

| strName | Name for registration |
| id | Class GUID |
| overwrite | If overwrite is true then overwrites old name registration else returns false |
Returns

**bool** True on success

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

- iedfactory.h

### 8.256 IEDLError Class Reference

An abstract class for EDL exceptions.

```cpp
#include <edlerrors.h>
```

Inherits exception.

### 8.256.1 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:

```cpp
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

### 8.257 IEDLFontSystemFont Class Reference

Representation of fonts installed on the target system (OS dependant).

```cpp
#include <idomfont.h>
```
Inheritance diagram for IEDLFontSystemFont:

![Inheritance Diagram]

Public Types

- enum **eFontFamily**
  Type used to uniquely identify the font face-family.
- enum **eFontDecoration**
  Type used to define a set of bit flag values that describe the visual decorations designed into the font.

Public Member Functions

- virtual bool **getFontFilePath** (EDLString &fontFilePath)=0
  Gets the font file path.
- virtual bool **setFontFilePath** (EDLString &fontFilePath)=0
  Sets the font file path.
- virtual bool **getFontName** (EDLString &fontName)=0
  Gets the font name.
- virtual bool **getFontWeight** (uint16 &weight)=0
  Gets the font weight.
- virtual **eFontFamily** **getFontFamily** ()=0
  Gets the font family.
- virtual uint32 **getFontDecorationFlags** ()=0
  Gets the font decoration flags.
virtual bool getAscentDescentRatio (float &ascendDescentRatio)=0

Gets the ratio of the font ascent to descent, if available.

virtual bool getAverageCharAspectRatio (float &averageCharAspectRatio)=0

Gets the average character aspect ratio, if available.

virtual bool loadFont (IDOMFontPtr &font)=0

Loads the font from the system and makes it available as a regular DOM Font. Only TrueType and Opentype fonts are currently supported.

Static Public Member Functions

• static const CClassID & classID ()

Retrieves class id of IDOM.

Additional Inherited Members

8.257.1 Detailed Description

Representation of fonts installed on the target system (OS dependant).

8.257.2 Member Function Documentation

8.257.2.1 classID()

static const CClassID & IEDLFontSystemFont::classID ( ) [inline], [static]

Retrieves class id of IDOM.

Returns

CClassID class id of the element

8.257.2.2 getAscentDescentRatio()

virtual bool IEDLFontSystemFont::getAscentDescentRatio ( 
    float & ascendDescentRatio ) [pure virtual]

Gets the ratio of the font ascent to descent, if available.

Parameters

ascendDescentRatio  Reference parameter to return the ratio.
8.257.2.3  getAverageCharAspectRatio()

virtual bool IEDLFontSystemFont::getAverageCharAspectRatio ( 
    float & averageCharAspectRatio )  [pure virtual]

Gets the average character aspect ratio, if available.

Parameters

- **averageCharAspectRatio**
  Reference parameter to return the ratio.

Returns

  **bool** The function returns an indication of whether the ratio is available.

8.257.2.4  getFontDecorationFlags()

virtual uint32 IEDLFontSystemFont::getFontDecorationFlags ( )  [pure virtual]

Gets the font decoration flags.

Returns

  **uint32** The font decoration flags, defined in typedef eFontDecoration.

8.257.2.5  getFontFamily()

virtual eFontFamily IEDLFontSystemFont::getFontFamily ( )  [pure virtual]

Gets the font family.

Returns

  **eFontFamily** The font family.

8.257.2.6  getFontFilePath()

virtual bool IEDLFontSystemFont::getFontFilePath ( 
    EDLString & fontFilePath )  [pure virtual]

Gets the font file path.
Parameters

| fontFilePath | Reference parameter to return the font file path. |

Returns

**bool** True on success, false if the call fails.

### 8.257.2.7 getFontName()

```cpp
def getFontName(EDLString &fontName) -> bool
```

Gets the font name.

Parameters

| fontName | Reference parameter to return the font name. |

Returns

**bool** True on success, false if the call fails.

### 8.257.2.8 getFontWeight()

```cpp
def getFontWeight(uint16 &weight) -> bool
```

Gets the font weight.

'Font Weight' describes the font stroke weight. The convention of weight values is given below:

<table>
<thead>
<tr>
<th>Weight</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNKNOWN</td>
<td>0</td>
</tr>
<tr>
<td>THIN</td>
<td>100</td>
</tr>
<tr>
<td>EXTRALIGHT</td>
<td>200</td>
</tr>
<tr>
<td>ULTRALIGHT</td>
<td>200</td>
</tr>
<tr>
<td>LIGHT</td>
<td>300</td>
</tr>
<tr>
<td>NORMAL</td>
<td>400</td>
</tr>
<tr>
<td>REGULAR</td>
<td>400</td>
</tr>
<tr>
<td>MEDIUM</td>
<td>500</td>
</tr>
<tr>
<td>SEMIBOLD</td>
<td>600</td>
</tr>
<tr>
<td>DEMIBOLD</td>
<td>600</td>
</tr>
<tr>
<td>BOLD</td>
<td>700</td>
</tr>
<tr>
<td>EXTRABOLD</td>
<td>800</td>
</tr>
<tr>
<td>ULTRABOLD</td>
<td>800</td>
</tr>
<tr>
<td>HEAVY</td>
<td>900</td>
</tr>
<tr>
<td>BLACK</td>
<td>900</td>
</tr>
</tbody>
</table>
Parameters

| weight | Reference parameter to return the font weight. |

Returns

**bool** True on success, false if the call fails.

### 8.257.2.9 loadFont()

```cpp
virtual bool IEDLFontSystemFont::loadFont ( 
    IDOMFontPtr & font ) [pure virtual]
```

Loads the font from the system and makes it available as a regular DOM Font. Only TrueType and Opentype fonts are currently supported.

Parameters

| font | Reference parameter to return the loaded DOM Font. |

Returns

**bool** True on success, false if the call fails.

### 8.257.2.10 setFontFilePath()

```cpp
virtual bool IEDLFontSystemFont::setFontFilePath ( 
    EDLString & fontFilePath ) [pure virtual]
```

Sets the font file path.

Parameters

| fontFilePath | The font file path. |

Returns

**bool** True on success, false if the call fails.

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

- `idomfont.h`
8.258  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 namespace prefix.*

- virtual bool getNamespace (EDLSysString &sNamespace) const =0
  
  *Retrieves the namespace.*

- virtual bool setNamespace (const EDLSysString &sNamespace)=0
  
  *Sets the namespace.*

Static Public Member Functions

- static const CClassID & classID ()
  
  *Retrieves class id of IEDLNamespace.*
Additional Inherited Members

8.258.1 Detailed Description

Interface to EDL Namespace class.

8.258.2 Member Function Documentation

8.258.2.1 classID()

static const CClassID & IEDLNamespace::classID ( ) [inline], [static]

Retrieves class id of IEDLNamespace.

Returns

  CClassID class id of the element

8.258.2.2 getNamespace()

virtual bool IEDLNamespace::getNamespace (  
        EDLSysString & sNamespace ) const [pure virtual]

Retrieves the namespace.

Parameters

| sNamespace | Namespace |

Returns

  bool Returns true on success

8.258.2.3 getPrefix()

virtual bool IEDLNamespace::getPrefix (  
        EDLSysString & sPrefix ) const [pure virtual]

Retrieves the namespace prefix.
Parameters

\textit{\textbf{sPrefix}} Namespace prefix

Returns

\textbf{bool} Returns true on success

8.258.2.4 setNamespace()

\textbf{virtual bool IEDLNamespace::setNamespace ( const EDLSysString & sNamespace ) [pure virtual]}

Sets the namespace.

Parameters

\textit{\textbf{sNamespace}} Namespace

Returns

\textbf{bool} Returns true on success

8.258.2.5 setPrefix()

\textbf{virtual bool IEDLNamespace::setPrefix ( const EDLSysString & sPrefix ) [pure virtual]}

Sets the namespace prefix.

Parameters

\textit{\textbf{sPrefix}} Namespace prefix

Returns

\textbf{bool} Returns true on success

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

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

```cpp
#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.

Additional Inherited Members

8.259.1 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

8.259.2 Member Function Documentation

8.259.2.1 clone()

virtual bool IEDLObject::clone (  

  IEDLObjectPtr & ptrObject,
  IEDLClassFactory * pFactory ) [inline], [virtual]

Create a copy of EDLObject.

Parameters

<table>
<thead>
<tr>
<th>_ptrObject</th>
<th>Smart pointer to the source object</th>
</tr>
</thead>
<tbody>
<tr>
<td>_pFactory</td>
<td>Pointer to the EDL class factory</td>
</tr>
</tbody>
</table>

Returns

- bool Returns true on success

Reimplemented in IEDLTempStore.
8.259.2.2 getClassID()

virtual const CClassID & IEDLObject::getClassID ( ) const [pure virtual]

Returns class ID of IEDLObject.

Returns

CClassID Returns reference to class ID

8.259.2.3 init()

virtual bool IEDLObject::init ( CClassParams * pData ) [inline], [virtual]

The init() method is called to perform any post-construction initialization of an IEDLObject that has been created by the EDL class factory, before it is actually returned by the factory.

Parameters

| pData | It is up to 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

8.260 IEDLStream Class Reference

Generic stream. Abstract base class for EDL stream subsystem.

#include <edlstream.h>
Inheritance diagram for IEDLStream:

![Inheritance Diagram](image_url)

Public Member Functions

- virtual bool isValid () const =0
  
  Determine stream validity.
- virtual bool open ()=0
  
  Opens the stream.
- virtual void close ()=0
  
  Closes the stream.
- virtual int64 getPos ()=0
  
  Get current stream position.

Additional Inherited Members

8.260.1 Detailed Description

Generic stream. Abstract base class for EDL stream subsystem.

8.260.2 Member Function Documentation
8.260.2.1 getPos()

virtual int64 IEDLStream::getPos ( ) [pure virtual]

Get current stream position.

Returns

```
int64 The current stream position
```

8.260.2.2 isValid()

virtual bool IEDLStream::isValid ( ) const [pure virtual]

Determine stream validity.

Returns

```
bool True if stream is valid for operations (no fatal problems)
```

8.260.2.3 open()

virtual bool IEDLStream::open ( ) [pure virtual]

Opens the stream.

Returns

```
bool True if stream was successfully opened
```

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

- edistream.h

Generated by Doxygen
8.261 IEDLTempStore Class Reference

A self-cleaning area for storage of temporary data in the form of streams. One per session, obtainable from an ISession.

#include <iedltempstore.h>

Inheritance diagram for IEDLTempStore:

```
IROObject
/
|
|
|
|
IEDLObject
/
|
|
|
|
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. Throws an IEDLError on failure.*

- virtual IRAInputOutputStreamPtr createTemporaryReaderWriterPair (IRAInputStreamPtr &reader, IRAOutputStreamPtr &writer)=0
  
  *Create a temporary object and provide a reader/writer pair. Convenience method. Throws an IEDLError on failure.*

- virtual IRAInputOutputStreamPtr createTemporaryReaderWriter ()=0
  
  *Create a temporary object and provide a consolidated reader/writer. Convenience method. Throws an IEDLError on failure.*

- virtual bool clone (IEDLObjectPtr &ptrObject, IEDLClassFactory *pFactory)
  
  *Clone - not available for objects of this type.*
Static Public Member Functions

- static const CClassID & classID ()
  Retrieves class id of IEDLTempStore.

Additional Inherited Members

8.261.1 Detailed Description

A self-cleaning area for storage of temporary data in the form of streams. One per session, obtainable from an ISession.

8.261.2 Member Function Documentation

8.261.2.1 classID()

static const CClassID & IEDLTempStore::classID ( ) [inline], [static]

Retrieves class id of IEDLTempStore.

Returns

  CClassID class id of the object

8.261.2.2 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.

8.261.2.3 createTemporaryObjectWithContents()

virtual IEDLTempStoreObjectPtr IEDLTempStore::createTemporaryObjectWithContents ( const IInputStreamPtr & stream ) [pure virtual]

Create a temporary object with the contents of an existing input stream.
Parameters

| stream | Reference to the source stream. |

Returns

IEDLTempStoreObjectPtr The new temporary object, or null on failure.

8.261.2.4 createTemporaryReaderWriter()

virtual IRAInputOutputStreamPtr IEDLTempStore::createTemporaryReaderWriter ( ) [pure virtual]

Create a temporary object and provide a consolidated reader/writer. Convenience method. Throws an IEDLError on failure.

Returns

IRAInputOutputStreamPtr The consolidated reader/writer for the temporary object.

8.261.2.5 createTemporaryReaderWriterPair()

virtual void IEDLTempStore::createTemporaryReaderWriterPair ( IRAInputStreamPtr & reader,
                                                           IRAOutputStreamPtr & writer ) [pure virtual]

Create a temporary object and provide a reader/writer pair. Convenience method. Throws an IEDLError on failure.

Parameters

| IRAInputStreamPtr | reference to receive a reader for the temporary object. |
| IRAOutputStreamPtr | reference to receive a writer for the temporary object. |

8.261.2.6 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. Throws an IEDLError on failure.
Parameters

| stream | Reference to the source stream. |

Returns

IRAInputStreamPtr The result stream.

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

- iedltempstore.h

8.262 IEDLTempStoreObject Class Reference

A temporary, file-like object, stored with an IEDLTempStore.

#include <iedltempstore.h>

Inheritance diagram for IEDLTempStoreObject:

![Inheritance Diagram](image)

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.*
Additional Inherited Members

8.262.1 Detailed Description

A temporary, file-like object, stored with an IEDLTempStore.

8.262.2 Member Function Documentation

8.262.2.1 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

| bool | buffered - If set to false, the resulting reader will not buffer read data. Useful if a reader and writer are being used on the same IEDLTempStoreObject, as it allows changes made by a writer to be immediately available to users of the reader. |

Returns

IRAInputStreamPtr The reader.

8.262.2.2 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:

- iedltmpstore.h
# IEDLTime Class Reference

Interface to EDL date-time class.

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

Inheritance diagram for IEDLTime:

```
+-----------------+         +-----------------+         +-----------------+
| IRCObject       |         | IEDLObject      |         | IEDLTime         |
|                 v         v                 v                 
| IEDLTime         |         | Initialization data |
```

## 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.

Static Public Member Functions

- static IEDLTimePtr createNow (IEDLClassFactory ∗pFactory)
  
  Simplified creator to create the time as of now.

Additional Inherited Members

8.263.1 Detailed Description

Interface to EDL date-time class.

8.263.2 Member Function Documentation

8.263.2.1 compare()

virtual int32 IEDLTime::compare (  
    const IEDLTimePtr & ptrTime ) const [pure virtual]

Compare this time against ptrTime.

Parameters

| ptrTime | The time to compare. |
8.263.2.2 fromPDFDate()

virtual bool IEDLTime::fromPDFDate ( 
    const EDLSysString & strTime ) [pure virtual]

Fill IEDLTime value from string with PDF date format.

Parameters

<table>
<thead>
<tr>
<th>EDLSysString</th>
<th>The string</th>
</tr>
</thead>
</table>

8.263.2.3 fromW3CDTF()

virtual bool IEDLTime::fromW3CDTF ( 
    const EDLSysString & strTime ) [pure virtual]

Fill IEDLTime value from string with W3CDTF format.

Parameters

<table>
<thead>
<tr>
<th>EDLSysString</th>
<th>The string</th>
</tr>
</thead>
</table>

8.263.2.4 getDay()

virtual uint16 IEDLTime::getDay ( ) const [pure virtual]

Retrieves the day value.

Returns

   bool Returns true on success, false if the call fails.
8.263.2.5  getMonth()

```
virtual uint16 IEDLTime::getMonth ( ) const [pure virtual]
```

Retrieves the month value.

Returns

bool Returns true on success, false if the call fails.

8.263.2.6  getTime()

```
virtual void IEDLTime::getTime ( 
    uint16 & hour,
    uint16 & min,
    uint16 & sec,
    uint16 & ms,
    int16 & tgd_sign,
    uint16 & tgd_hour,
    uint16 & tgd_min ) const [pure virtual]
```

Retrieves the time.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hour</td>
<td>The hour component</td>
</tr>
<tr>
<td>min</td>
<td>The minute component</td>
</tr>
<tr>
<td>sec</td>
<td>The second component</td>
</tr>
<tr>
<td>ms</td>
<td>The fractional second component</td>
</tr>
<tr>
<td>tgd_sign</td>
<td>The timezone sign component</td>
</tr>
<tr>
<td>tgd_hour</td>
<td>The timezone hour component</td>
</tr>
<tr>
<td>tgd_min</td>
<td>The timezone minute component</td>
</tr>
</tbody>
</table>

8.263.2.7  getYear()

```
virtual uint16 IEDLTime::getYear ( ) const [pure virtual]
```

Retrieves the year value.

Returns

bool Returns true on success, false if the call fails.
8.263.2.8 isEqualTo()

virtual bool IEDLTime::isEqualTo (  
    const IEDLTimePtr & ptrTime ) const [pure virtual]

Check this IEDLTime for equality to another IEDLTime.

Parameters

    *ptrTime*  The time to compare.

Returns

    bool. Returns true if the times are the same.

8.263.2.9 setDay()

virtual bool IEDLTime::setDay (  
    uint16 day ) [pure virtual]

Sets the day.

Parameters

    *day*  The day value

8.263.2.10 setMonth()

virtual bool IEDLTime::setMonth (  
    uint16 month ) [pure virtual]

Sets the month.

Parameters

    *month*  The month value

8.263.2.11 setTime()

virtual bool IEDLTime::setTime (  
    uint16 hour = 0,
    uint16 minute = 0,
    uint16 second = 0,  
    uint16 millisecond = 0 ) [pure virtual]

Sets the hour, minute, second, and millisecond.
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

<table>
<thead>
<tr>
<th>hour</th>
<th>The hour component</th>
</tr>
</thead>
<tbody>
<tr>
<td>min</td>
<td>The minute component</td>
</tr>
<tr>
<td>sec</td>
<td>The second component</td>
</tr>
<tr>
<td>ms</td>
<td>The fractional second component</td>
</tr>
<tr>
<td>tzd_sign</td>
<td>The timezone sign component</td>
</tr>
<tr>
<td>tzd_hour</td>
<td>The timezone hour component</td>
</tr>
<tr>
<td>tzd_min</td>
<td>The timezone minute component</td>
</tr>
</tbody>
</table>

Returns

bool Returns true on success, false if the call fails.

8.263.2.12 setYear()

virtual bool IEDLTime::setYear (  
        uint16 year ) [pure virtual]

Sets the year.

Parameters

| year | The year value |

8.263.2.13 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
8.263.2.14 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

8.264 IFontHeaderWriteSegmentBlockEnumerator Class Reference

IFontHeaderWriteSegmentBlockEnumerator Enumerates over the PCLXL Font Header block items for the XL ReadFontHeader operator.

#include <idomfont.h>

Inheritance diagram for IFontHeaderWriteSegmentBlockEnumerator:

![Inheritance Diagram]

Public Types

• enum eFontHeaderWriteSegmentItem
  the enumeration of the segment types to be enumerated over
Public Member Functions

- `eFontHeaderWriteSegmentItem getSegmentItem()`  
  Returns the current segment item in the enumeration.
- `int32 getSegmentBlockItem()`  
  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(IOutputStreamPtr &outStream, eEDLEndian endian)=0`  
  Writes the current enumeration item block.
- `virtual bool haveMoreEnumerationItems()=0`  
  Indicates whether there are more items in the enumeration.
- `virtual bool nextEnumerationItem()=0`  
  Gets the next enumeration item.

Static Public Member Functions

- `static const CClassID & classID()`  
  Retrieves class id of IDOM.

Additional Inherited Members

8.264.1 Detailed Description

IFontHeaderWriteSegmentBlockEnumerator Enumerates over the PCLXL Font Header block items for the XL ReadFontHeader operator.

8.264.2 Member Function Documentation

8.264.2.1 classID()

static const CClassID& IFontHeaderWriteSegmentBlockEnumerator::classID () [inline], [static]

Retrieves class id of IDOM.

Returns

  CClassID Class id of the element
8.264.2.2  getSegmentBlockItem()

```c
int32 IFontHeaderWriteSegmentBlockEnumerator::getSegmentBlockItem() { }
```

Returns the current block item of segment item in the enumeration.

Returns

```
Int32 The segment block number
```

8.264.2.3  getSegmentBlockSize()

```c
virtual int32 IFontHeaderWriteSegmentBlockEnumerator::getSegmentBlockSize() [pure virtual]
```

Returns the current enumeration item segment block size.

Returns

```
Int32 The segment block size
```

8.264.2.4  getSegmentItem()

```c
eFontHeaderWriteSegmentItem IFontHeaderWriteSegmentBlockEnumerator::getSegmentItem() { }
```

Returns the current segment item in the enumeration.

Returns

```
eFontHeaderWriteSegmentItem The segment item
```

8.264.2.5  haveMoreEnumerationItems()

```c
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.
```
8.264.2.6 nextEnumerationItem()

virtual bool IFontHeaderWriteSegmentBlockEnumerator::nextEnumerationItem ( ) [pure virtual]

Get the next enumeration item.

Returns

   bool True on success

8.264.2.7 writeSegmentBlock()

virtual int32 IFontHeaderWriteSegmentBlockEnumerator::writeSegmentBlock ( 
   IOOutputStreamPtr & outStream,
   eEDLEndian endian  ) [pure virtual]

Writes the current enumeration item block.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>outStream</td>
<td>The output stream</td>
</tr>
<tr>
<td>endian</td>
<td>The endian of the data to write out</td>
</tr>
</tbody>
</table>

Returns

   int32 The size of data written to the output stream

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

- idomfont.h

8.265 IFontPCL5WriteSegmentBlockEnumerator Class Reference

IFontPCL5WriteSegmentBlockEnumerator Enumerates over the PCL5 font blocks.

#include <idomfont.h>
Inheritance diagram for IFontPCL5WriteSegmentBlockEnumerator:

![Inheritance Diagram]

**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 (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.*

**Static Public Member Functions**

- static const CClassID & classID ()
  
  *Retrieves class id of IDOM.*

**Additional Inherited Members**

8.265.1 Detailed Description

IFontPCL5WriteSegmentBlockEnumerator Enumerates over the PCL5 font blocks.

8.265.2 Member Function Documentation
8.265.2.1 classID()

static const CClassID & IFontPCL5WriteSegmentBlockEnumerator::classID ( ) [inline], [static]

Retrieves class id of IDOM.

Returns

CClassID Class id of the element

8.265.2.2 getEnumerationCount()

virtual int32 IFontPCL5WriteSegmentBlockEnumerator::getEnumerationCount ( ) [pure virtual]

Return the total number of enumeration blocks that will be iterated through.

Returns

int32 The enumeration count

8.265.2.3 getEnumerationItemBlockSize()

virtual int32 IFontPCL5WriteSegmentBlockEnumerator::getEnumerationItemBlockSize ( ) [pure virtual]

Return the current enumeration item block size.

Returns

int32 The segment block size

8.265.2.4 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.
8.265.2.5  nextEnumerationItem()

virtual bool IFontPCL5WriteSegmentBlockEnumerator::nextEnumerationItem ( ) [pure virtual]

Get the next enumeration item.

Returns

bool True on success

8.265.2.6  writeEnumerationItemBlock()

virtual int32 IFontPCL5WriteSegmentBlockEnumerator::writeEnumerationItemBlock ( 
  IOutputStreamPtr & outStream,
  eEDLEndian endian ) [pure virtual]

Writes the current enumeration item block.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>outStream</td>
<td>The output stream</td>
</tr>
<tr>
<td>endian</td>
<td>The endian of the data to write out</td>
</tr>
</tbody>
</table>

Returns

int32 The size of data written to the output stream

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

- idomfont.h

8.266  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 IFormField.

- 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 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 bool getNeedAppearances () const =0
  
  Get whether a viewer should construct its own appearances for the widgets in this field.

- virtual void setNeedAppearances (bool needAppearances)=0
  
  Set whether a viewer should construct its own appearances for the widgets in this field.

Additional Inherited Members

8.266.1 Detailed Description

An interface class for an interactive form, which tracks a tree of IFormField and widgets.
8.266.2 Member Function Documentation

8.266.2.1 addField()

virtual void JawsMako::IForm::addField (  
    const IFormFieldPtr & field )  [pure virtual]

Add a field at the top level. An exception will be thrown if the field to be inserted has the same partial name or id as an existing field at the top level.

Parameters

  field  Field too be added

8.266.2.2 addWidget() [1/2]

virtual void JawsMako::IForm::addWidget (  
    const IWidgetAnnotationPtr & widget )  [pure virtual]

Add a top level widget. An exception will be thrown if an annotation with the same reference is already.

Parameters

  widget  The widget to be added

8.266.2.3 addWidget() [2/2]

virtual void JawsMako::IForm::addWidget (  
    const IAnnotationReferencePtr & widgetReference )  [pure virtual]

Add a top level widget by reference. An exception will be thrown if an annotation with the same reference is already present at this level.

Parameters

  widgetReference  The widget, by reference, to be added

8.266.2.4 clone()

virtual IFormPtr JawsMako::IForm::clone ( ) const  [pure virtual]
Clone the form field, which includes a deep clone of any child IFormField.

Returns

IFormField The cloned form field

8.266.2.5 fieldInTree()

virtual bool JawsMako::IFormField::fieldInTree ( 
    const IFormFieldPtr & field ) const [pure virtual]

Search the field tree for a field that has the same id as the given field. The search recurses through the children.

Parameters

field The field to be matched

Returns

bool True if field was found, otherwise false

8.266.2.6 getFields()

virtual CFormFieldVect JawsMako::IFormField::getFields ( ) const [pure virtual]

Get the top level fields in a vector.

Returns

CFormFieldVect The vector of fields

8.266.2.7 getNeedAppearances()

virtual bool JawsMako::IFormField::getNeedAppearances ( ) const [pure virtual]

Get whether a viewer should construct its own appearances for the widgets in this field.

Returns

bool True if viewer should construct its own appearances
virtual CFormFieldVect JawsMako::IForm::getPathToField (const IFormFieldPtr & field) const [pure virtual]

Find the path to a field with the same id that is present in the field tree.

This returns a vector of IFormFields representing the branches of the field tree that lead to the given field. That is, the last entry in the returned vector is the immediate parent of the desired field, the second last entry is the grandparent of the desired field, and so forth leading back to the top level.

This is useful in particular for determining the fully qualified name of a field.

An empty vector is returned if the requested field is at the top level of the form and therefore has no parents.

An exception is thrown if the field is not present in the tree.

Parameters

- **field** The field to be matched

Returns

- **CFormFieldVect** A vector of IFormFields

virtual CFormFieldVect JawsMako::IForm::getPathToWidget (const IWidgetAnnotationPtr & widget) const [pure virtual]

Find the path to a widget with the same reference as the given widget.

This returns a vector of IFormFields representing the branches of the field tree that lead to the given widget. That is, the last entry in the returned vector is the immediate parent of the desired widget, the second last entry is the grandparent of the desired widget, and so forth leading back to the top level.

This is useful in particular for determining the fully qualified name of a field.

An empty vector is returned if the requested widget is at the top level of the form and therefore has no parents.

An exception is thrown if the widget is not present in the tree.

Parameters

- **widget** The widget to be matched

Returns

- **CFormFieldVect** A vector of IFormFields
virtual CFormFieldVect JawsMako::IForm::getPathToWidget ( 
    const IAnnotationReferencePtr & widgetReference ) const [pure virtual]

Find the path to a widget with the given reference.

This returns a vector of IFormFields representing the branches of the field tree that lead to the given widget. That is, the last entry in the returned vector is the immediate parent of the desired widget, the second last entry is the grandparent of the desired widget, and so forth leading back to the top level.

This is useful in particular for determining the fully qualified name of a field.

An empty vector is returned if the requested widget is at the top level of the form and therefore has no parents.

An exception is thrown if the widget is not present in the tree.

Parameters

| widgetReference | The widget, by reference, to be matched |

Returns

CFormFieldVect A vector of IFormFields

8.266.2.11 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

8.266.2.12 removeField()

virtual void JawsMako::IForm::removeField ( 
    const IFormFieldPtr & field ) [pure virtual]

Remove a top level field.

Parameters

| field | Field to be removed |
8.266.2.13  removeWidget() [1/2]

virtual void JawsMako::IForm::removeWidget (  
    const IWidgetAnnotationPtr & widget ) [pure virtual]

Remove the given top level widget.

Parameters

widget  The widget to be removed

8.266.2.14  removeWidget() [2/2]

virtual void JawsMako::IForm::removeWidget (  
    const IAnnotationReferencePtr & widgetReference ) [pure virtual]

Remove the given top level widget, by reference.

Parameters

widgetReference  The widget, by reference, to be removed

8.266.2.15  setNeedAppearances()

virtual void JawsMako::IForm::setNeedAppearances (  
    bool needAppearances ) [pure virtual]

Set whether a viewer should construct its own appearances for the widgets in this field.

Parameters

needAppearances  True or false

8.266.2.16  widgetInTree() [1/2]

virtual bool JawsMako::IForm::widgetInTree (  
    const IWidgetAnnotationPtr & widget ) const [pure virtual]

Search the field sub for a widget. The search recurses through the child fields.
Parameters

| widget | The widget to be matched |

Returns

**bool** True if widget was found, otherwise false

8.266.2.17 **widgetInTree()** [2/2]

virtual bool JawsMako::IForm::widgetInTree (const IAnnotationReferencePtr & widgetReference) const [pure virtual]

Search the field sub for a widget, by reference The search recurses through the child fields.

Parameters

| widgetReference | The widget, by reference, to be matched |

Returns

**bool** True if widget was found, otherwise false

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

- interactive.h

8.267 **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 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.

Additional Inherited Members

8.267.1 Detailed Description

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

Generated by Doxygen
8.267.2 Member Function Documentation

8.267.2.1 addChildField()

virtual void JawsMako::IFormField::addChildField ( const IFormFieldPtr & field ) [pure virtual]

Add a child field. An exception will be thrown if the field to be inserted has the same partial name or id as an existing field.

Parameters

| field | Field to be added |

8.267.2.2 addChildWidget() [1/2]

virtual void JawsMako::IFormField::addChildWidget ( const IWidgetAnnotationPtr & widget ) [pure virtual]

Add a child widget. An exception will be thrown if an annotation with the same reference is already present at this level.

Parameters

| widget | The widget to be added |

8.267.2.3 addChildWidget() [2/2]

virtual void JawsMako::IFormField::addChildWidget ( const IAnnotationReferencePtr & widgetReference ) [pure virtual]

Add a child widget by reference. An exception will be thrown if an annotation with the same reference is already present at this level.

Parameters

| widgetReference | The widget, by reference, to be added |
8.267.2.4  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

8.267.2.5  fieldInSubtree()

virtual bool JawsMako::IFormField::fieldInSubtree ( const IFormFieldPtr & field ) const [pure virtual]

Search the IFormField subtree for a field that has the same id as the given field. The search recurses through the children.

Parameters

   field  Field to search for

Returns

   bool  True if found, otherwise false

8.267.2.6  getChildFields()

virtual CFormFieldVect JawsMako::IFormField::getChildFields ( ) const [pure virtual]

Get the child fields in a vector.

Returns

   CFormFieldVect  The vector of child fields

8.267.2.7  getChildWidgets()

virtual CAnnotationReferenceVect JawsMako::IFormField::getChildWidgets ( ) const [pure virtual]

Get references to the child widgets in a vector.

Returns

   CAnnotationReferenceVect  The vector of widgets' references
8.267.2.8 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

8.267.2.9 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

8.267.2.10 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

8.267.2.11 removeChildField()

virtual void JawsMako::IFormField::removeChildField ( const IFormFieldPtr & field ) [pure virtual]

Remove a child field.

Parameters

| field | Field to be removed |

Generated by Doxygen
8.267.2.12  removeChildWidget() [1/2]

virtual void JawsMako::IFormField::removeChildWidget (  
    const IWidgetAnnotationPtr & widget ) [pure virtual]

Remove the given child widget from the field.

Parameters

| widget | The widget to be removed |

8.267.2.13  removeChildWidget() [2/2]

virtual void JawsMako::IFormField::removeChildWidget (  
    const IAnnotationReferencePtr & widgetReference ) [pure virtual]

Remove the given child widget from the field, by reference.

Parameters

| widgetReference | The widget, by reference, to be removed |

8.267.2.14  setPartialName()

virtual void JawsMako::IFormField::setPartialName (  
    const U8String & name ) [pure virtual]

Set the partial name of the field.

Parameters

| name | The partial name |

8.267.2.15  widgetInSubtree() [1/2]

virtual bool JawsMako::IFormField::widgetInSubtree (  
    const IWidgetAnnotationPtr & widget ) const [pure virtual]

Search the IFormField subtree for a widget The search recurses through the children.
Parameters

| widget  | The widget to search for |

Returns

bool True if found, otherwise false

8.267.2.16 widgetInSubtree()

virtual bool JawsMako::IFormField::widgetInSubtree (const IAnnotationReferencePtr & widgetReference) const [pure virtual]

Search the IFormField subtree for a widget, by reference The search recurses through the children.

Parameters

| widgetReference | The widget, by reference, to search for |

Returns

bool True if found, otherwise false

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

- interactive.h

8.268 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:

```
IRObject

JawsMako::ITransform

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.`

### Static Public Member Functions

- `static JAWSMAKO_API IFormUnpackerTransformPtr create (const IJawsMakoPtr &jawsMako)`
  
  Create the transform.

### Additional Inherited Members

#### 8.268.1 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.

#### 8.268.2 Member Function Documentation

##### 8.268.2.1 create()

```
static JAWSMAKO_API IFormUnpackerTransformPtr JawsMako::IFormUnpackerTransform::create (const IJawsMakoPtr & jawsMako ) [static]
```

Create the transform.
Parameters

| `JawsMako` | The JawsMako instance. |

Returns

The new instance.

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

- `transforms.h`

### 8.269 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.

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

Inheritance diagram for JawsMako::IFreeTextAnnotation:

```
IRCOObject

<table>
<thead>
<tr>
<th>JawsMako::IAnnotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>JawsMako::IMarkupAnnotation</td>
</tr>
<tr>
<td>JawsMako::IFreeTextAnnotation</td>
</tr>
</tbody>
</table>
```

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.*
Additional Inherited Members

8.269.1 Detailed Description

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

8.269.2 Member Function Documentation

8.269.2.1 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.

8.269.2.2 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
8.270  JawsMako::IJawsRenderer::IHalftone Class Reference

An abstract base class for communicating halftone information to the Jaws renderer, for use with `renderMonochrome()` and `renderMonochromeToFrameBuffer()`.

```cpp
#include <jawsmako.h>
```

Inheritance diagram for JawsMako::IJawsRenderer::IHalftone:

8.270.1  Detailed Description

An abstract base class for communicating halftone information to the Jaws renderer, for use with `renderMonochrome()` and `renderMonochromeToFrameBuffer()`.

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

- jawsmako.h

8.271  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.

```cpp
#include <iimagecodec.h>
```
Inheritance diagram for IImageDecoder:

```
8.272 JawsMako::IImageDownsamplerTransform Class Reference

Public Member Functions

- IImageDecoder()
  Constructor.
- virtual ~IImageDecoder()
  Destructor.
- virtual int32 getNumFrames()
  Returns the number of frames for this image file.

Additional Inherited Members

8.271.1 Detailed Description

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

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

- imagecodec.h

8.272 JawsMako::IImageDownsamplerTransform Class Reference

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

#include <transforms.h>

Generated by Doxygen
Inheritance diagram for JawsMako::IImageDownsamplerTransform:

```
IRCOBJECT

JawsMako::ITransform

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 the target resolution (dpi) for downsampling color images.*
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.

**Static Public Member Functions**

- static JAWSMAKO_API IImageDownsamplerTransformPtr create (const IJawsMakoPtr &jawsMako)
  Create the transform.

**Additional Inherited Members**

8.272.1 Detailed Description

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

8.272.2 Member Function Documentation

8.272.2.1 create()

static JAWSMAKO_API IImageDownsamplerTransformPtr JawsMako::IImageDownsamplerTransform::create
{
  const IJawsMakoPtr & jawsMako } [static]

Create the transform.
Parameters

| JawsMako | The JawsMako instance. |

Returns

The new instance.

### 8.272.2.2 setDownsampleMaskedImages()

```
virtual void JawsMako::IImageDownsamplerTransform::setDownsampleMaskedImages (
   bool downsampleMaskedImages ) [pure virtual]
```

Set whether or not to downsample masked images.

This applies to cases where an image is masked by a separate masked image, such as types of PDF masked or soft-masked images. These are represented in the DOM using `IDOMMaskedBrush`, where the sub-brush is an image.

The default is true.

### 8.272.2.3 setUseMaskResolutionForMaskedImages()

```
virtual void JawsMako::IImageDownsamplerTransform::setUseMaskResolutionForMaskedImages (
   bool useMaskResolutionSettingForMaskedImages ) [pure virtual]
```

Set whether or not the mask resolution setting should be applied to the image portion of a masked image.

This applies to cases where an image is masked by a separate masked image, such as types of PDF masked or soft-masked images. These are represented in the DOM using `IDOMMaskedBrush`, where the sub-brush is an image.

If false, then the mask and the image are evaluated separately and a downsampling resolution and threshold are chosen. For the mask, this is normally either grayscale or monochrome. The image data can be anything. In this mode it is possible for the downsampled image and mask to be downsampled to different resolutions.

If true, then whatever target resolution and threshold is applied to the mask will also be applied to the image samples. If these images have the same effective resolution before downsampling, then they will also share the same effective resolution after downsampling.

The default is false.

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

- `transforms.h`
IImageEncoder accepts IImageFrame objects and streams it out to an image format.

#include <iimagecodec.h>

Inheritance diagram for IImageEncoder:

![Inheritance Diagram](diagram.png)

Public Member Functions

- IImageEncoder ()
  Constructor.
- virtual ~IImageEncoder ()
  Destructor.

Additional Inherited Members

8.273.1 Detailed Description

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

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

- iimagecodec.h

Generated by Doxygen
8.274 JawsMako::IImageEncoderTransform Class Reference

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

#include <transforms.h>

Inheritance diagram for JawsMako::IImageEncoderTransform:

```
IRObject

JawsMako::ITransform

JawsMako::IImageEncoderTransform
```

Public Types

- `enum eEncodeFormat { eEFJPEG }`
  
  Available target image formats.

Public Member Functions

- `virtual void setReencodeAllImages (bool value)=0`
  
  Sets whether or not all images should be reencoded. The default is false; that is, only image types other than JPEG, TIFF, HDPHOTO and PNG will be re-encoded.

- `virtual void setEnableTIFFEncoding (bool value)=0`
  
  Set whether or not TIFF should ever be allowed. The default value is true. This is needed as some output formats do not support TIFF. If a TIFF is seen it will be re-encoded as something else.

- `virtual void 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 colour 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 colour should be encoded as an image with an alpha channel, rather than encoding the image separately. The default is true.

Static Public Member Functions

- static JAWSMAKO_API IImageEncoderTransformPtr create (const IJawsMakoPtr &jawsMako)
  Create the transform.

Additional Inherited Members

8.274.1 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 colour converted if they are not compatible with the desired image type.

8.274.2 Member Enumeration Documentation

8.274.2.1 eEncodeFormat

enum JawsMako::IImageEncoderTransform::eEncodeFormat

Available target image formats.
8.274.3 Member Function Documentation

8.274.3.1 create()

static JAWSMAKO_API IImageEncoderTransformPtr JawsMako::IImageEncoderTransform::create ( const IJawsMakoPtr & jawsMako ) [static]

Create the transform.

Parameters

| JawsMako | The JawsMako instance |

Returns

The new instance.

8.274.3.2 setColorTIFFCompression()

virtual void JawsMako::IImageEncoderTransform::setColorTIFFCompression ( IDOMTIFFImage::eTIFFCompression scheme ) [pure virtual]

Set the preferred TIFF compression when TIFF encoding is used for color images. Only PackBits, LZW or None may be used.

Parameters

| scheme | The preferred TIFF compression scheme. May not always be used as not all compression types can be used for all types. |

8.274.3.3 setGrayTIFFCompression()

virtual void JawsMako::IImageEncoderTransform::setGrayTIFFCompression ( IDOMTIFFImage::eTIFFCompression scheme ) [pure virtual]

Set the preferred TIFF compression when TIFF encoding is used for gray images. Only PackBits, LZW or None may be used.
Parameters

| scheme | The preferred TIFF compression scheme. May not always be used as not all compression types can be used for all types. |

8.274.3.4 setJPEGQuality()

virtual void JawsMako::IImageEncoderTransform::setJPEGQuality ( uint8 quality ) [pure virtual]

Set the JPEG encoding quality when JPEG encoding is used.

Parameters

| quality | The desired quality level, with 1 being lowest quality and 5 being highest quality. |

8.274.3.5 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

| scheme | The preferred TIFF compression scheme. May not always be used as not all compression types can be used for all types. |

8.274.3.6 setPreferredColorFormat()

virtual void JawsMako::IImageEncoderTransform::setPreferredColorFormat ( eEncodeFormat format ) [pure virtual]

Set the preferred format for colour images. The default is eEFAuto.

Parameters

| format | The preferred format. |
8.274.3.7  setPreferredFormat()

virtual void JawsMako::IImageEncoderTransform::setPreferredFormat (  
    eEncodeFormat format )  [pure virtual]

Set the preferred format for all images. This is only a preference and may not be honoured in all cases. The default is eEFAuto.

Parameters

| format | The preferred format. |

8.274.3.8  setPreferredGrayFormat()

virtual void JawsMako::IImageEncoderTransform::setPreferredGrayFormat (  
    eEncodeFormat format )  [pure virtual]

Set the preferred format for gray images. This is only a preference and may not be honoured in all cases. The default is eEFAuto.

Parameters

| format | The preferred format. |

8.274.3.9  setPreferredMonoFormat()

virtual void JawsMako::IImageEncoderTransform::setPreferredMonoFormat (  
    eEncodeFormat format )  [pure virtual]

Set the preferred format for monochrome images. This is only a preference and may not be honoured in all cases. The default is eEFAuto.

Parameters

| format | The preferred format. |

8.274.3.10  setTIFFCompression()

virtual void JawsMako::IImageEncoderTransform::setTIFFCompression (  
    IDOMTIFFImage::eTIFFCompression scheme )  [pure virtual]

Set the preferred TIFF compression when TIFF encoding is used.
Parameters

| scheme | The preferred TIFF compression scheme. May not always be used as not all compression types can be used for all types. |

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

- transforms.h

8.275  **IImageFrame Class Reference**

**IImageFrame** encapsulates an EDL image with its details.

#include <iimagecodec.h>

Inheritance diagram for IImageFrame:

![Inheritance diagram](image)

**Public Member Functions**

- **IImageFrame()** *Constructor.*
- **virtual ~IImageFrame()** *Destructor.*
- **virtual int32 getWidth() = 0** *Gets the width of the image.*
- **virtual int32 getHeight() = 0**
Gets the height of the image.

- virtual int32 getBPS ()=0
  Gets the bits per sample of the image.

- virtual bool getColorSpace (IDOMColorSpacePtr &colorSpace)=0
  Gets the color space.

- virtual ImageExtraChannel_t getExtraChannelType ()
  Gets the type of information contained in the extra channel.

- virtual int32 getNumExtraChannels ()
  Gets the number of image extra channels (for example an alpha or mask)

- virtual int32 getNumChannels ()
  Gets the total number of image channels (including extra channels)

- virtual bool getHasAlphaChannel ()
  Indicates if the image has an alpha mask.

- virtual bool getHasMask (bool &isNoisy)
  Indicates if the image has a mask (i.e. a binary alpha channel)

- virtual bool getHasICCProfile ()=0
  Indicates if the image has an ICC profile.

- virtual bool getICCProfile (IDOMICCProfilePtr &profile)=0
  Gets the ICC profile for the image.

- virtual double getXResolution ()

- virtual FMatrix *getMatrix () const =0
  Gets the transform for the image.

- virtual int32 getRawBytesPerRow ()=0
  Gets the image row size in bytes.

- virtual bool readScanLine (void *pRow)=0
  Gets an image row scanline (of size given by getRawBytesPerRow())

- virtual bool skipScanLines (uint32 skipCount)
  Skips the given number of scanline rows, which may be fast for some image types.

Additional Inherited Members

8.275.1 Detailed Description

IImageFrame encapsulates an EDL image with its details.

8.275.2 Member Function Documentation

8.275.2.1 getBPS()

virtual int32 IImageFrame::getBPS () { } [pure virtual]

Gets the bits per sample of the image.

Returns

int32. Returns the image bits from sample.
8.275.2.2 getColorSpace()

virtual bool IImageFrame::getColorSpace (IDOMColorSpacePtr & colorSpace) [pure virtual]

Gets the color space.

Parameters

| colorSpace | The colorspace to be passed back by reference. |

Returns

bool. Returns an indicator of success.

8.275.2.3 getExtraChannelType()

virtual ImageExtraChannel_t IImageFrame::getExtraChannelType ( ) [inline], [virtual]

Gets the type of information contained in the extra channel.

Returns

ImageExtraChannel_t. Returns the type of information in the extra channel.

8.275.2.4 getHasAlphaChannel()

virtual bool IImageFrame::getHasAlphaChannel ( ) [inline], [virtual]

Indicates if the image has an alpha mask.

Returns

bool. Returns an indication of alpha mask present in the image.

8.275.2.5 getHasICCProfile()

virtual bool IImageFrame::getHasICCProfile ( ) [pure virtual]

Indicates if the image has an ICC profile.

Returns

bool. Returns an indication of ICC profile present in the image.

8.275.2.6 getHasMask()

virtual bool IImageFrame::getHasMask (bool & isNoisy) [inline], [virtual]

Indicates if the image has a mask (i.e. a binary alpha channel)
Parameters

| isNoisy | An indicator that the alpha channel has binary values and so is acting as a mask. |

Returns

bool. Returns an indication of mask present in the image.

8.275.2.7 getHeight()

virtual int32 IImageFrame::getHeight ( ) [pure virtual]

Gets the height of the image.

Returns

int32. Returns the image height.

8.275.2.8 getICCProfile()

virtual bool IImageFrame::getICCProfile ( IDOMICCProfilePtr & profile ) [pure virtual]

Gets the ICC profile for the image.

Parameters

| profile | The image ICC profile to be passed back. |

Returns

bool. Returns an indication success.

8.275.2.9 getMatrix()

virtual FMatrix* IImageFrame::getMatrix ( ) const [pure virtual]

Gets the transform for the image.

Returns

FMatrix*. Returns the image transform.

Implemented in IImageFrameReader.
8.275.2.10 getNumChannels()

virtual int32 IImageFrame::getNumChannels ( ) [inline], [virtual]

Gets the total number of image channels (including extra channels)

Returns

int32. Returns the total number of image channels.

8.275.2.11 getNumExtraChannels()

virtual int32 IImageFrame::getNumExtraChannels ( ) [inline], [virtual]

Gets the number of image extra channels (for example an alpha or mask)

Returns

int32. Returns the number of image extra channels.

8.275.2.12 getRawBytesPerRow()

virtual int32 IImageFrame::getRawBytesPerRow ( ) [pure virtual]

Gets the image row size in bytes.

Returns

int32. The image row size in bytes

Implemented in IImageFrameWriter.

8.275.2.13 getWidth()

virtual int32 IImageFrame::getWidth ( ) [pure virtual]

Gets the width of the image.

Returns

int32. Returns the image width.
virtual double IImageFrame::getXResolution ( ) [inline], [virtual]

Please note: `getXResolution()` and `getYResolution()` are temporary APIs. Because it is not certain what the interpretation of resolution means for all image types. At the moment both APIs return values that are to be interpreted as pixels per inch and the default value of 96.0 (XPS resolution) is returned for any image (type) for which there is currently no other meaningful resolution value.

virtual bool IImageFrame::readScanLine ( void * pRow ) [pure virtual]

Gets an image row scanline (of size given by `getRawBytesPerRow()`).
Parameters

\texttt{pRow} \hspace{1em} \text{The buffer in which the row scanline will be copied into.}

Returns

\text{bool. An indicator of success.}

Implemented in \texttt{IImageFrameWriter}.

\subsection{8.275.2.16 skipScanLines()}

\begin{verbatim}
virtual bool IImageFrame::skipScanLines ( 
    uint32 skipCount ) [inline], [virtual]
\end{verbatim}

Skips the given number of scanline rows, which may be fast for some image types.

Parameters

\texttt{skipCount} \hspace{1em} \text{The number of scanlines to skip.}

Returns

\text{bool. An indicator of success.}

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

- \texttt{iimagecodec.h}

\section*{8.276 IImageFrameReader Class Reference}

\texttt{IImageFrameReader} reads an image into an imageframe.

\begin{verbatim}
#include <iimagecodec.h>
\end{verbatim}
Inheritance diagram for IImageFrameReader:

```
IRCOObject

IEDLObject

IImageFrame

IImageFrameReader
```

Public Member Functions

- **IImageFrameReader()**
  Constructor.
- **virtual ~IImageFrameReader()**
  Destructor.
- **virtual bool writeScanLine(void *pScanLine)**
  Writes an image row into the image frame.
- **virtual FMatrix * getMatrix() const**
  Gets the transform for the image.

Additional Inherited Members

8.276.1 Detailed Description

**IImageFrameReader** reads an image into an image frame.

8.276.2 Member Function Documentation
8.276.1 getMatrix()

virtual FMatrix* IImageFrameReader::getMatrix() const [inline], [virtual]

Gets the transform for the image.

Returns

FMatrix*. Returns the image transform.

Implements IImageFrame.

8.276.2 writeScanLine()

virtual bool IImageFrameReader::writeScanLine(
    void* pScanLine) [inline], [virtual]

Writes an image row into the image frame.

Parameters

pScanLine A buffer that contains the scanline.

Returns

bool. An indicator of success.

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

- iimagecodec.h

8.277 IImageFrameWriter Class Reference

IImageFrameWriter writes an image from an imageframe.

#include <iimagecodec.h>
Inheritance diagram for IImageFrameWriter:

![Inheritance Diagram](image.png)

**Public Member Functions**

- `IImageFrameWriter()`  
  Constructor.
- `virtual ~IImageFrameWriter()`  
  Destructor.
- `virtual int32 getRawBytesPerRow()`  
  Gets the image row size in bytes.
- `virtual bool readScanLine(void *pScanLine)`  
  reads a scanline

**Additional Inherited Members**

8.277.1 Detailed Description

*IImageFrameWriter* writes an image from an imageframe.

8.277.2 Member Function Documentation
8.278 JawsMako::IImageMergerTransform Class Reference

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

#include <transforms.h>

Inheritance diagram for JawsMako::IImageMergerTransform:

```
IRCOObject
```
```
JawsMako::ITransform
```
```
JawsMako::IImageMergerTransform
```

Public Member Functions

- virtual void setAllowMaskedResults (bool allowMasked)=0
  
  Sets whether or a mask channel is allowed in the merged result.
Static Public Member Functions

- static JAWSMAKO_API IImageMergerTransformPtr create (const IJawsMakoPtr &jawsMako)
  Create the transform.

Additional Inherited Members

8.278.1 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.

8.278.2 Member Function Documentation

8.278.2.1 create()

static JAWSMAKO_API IImageMergerTransformPtr JawsMako::IImageMergerTransform::create (const IJawsMakoPtr & jawsMako) [static]

Create the transform.

Parameters

JawsMako The JawsMako instance.

Returns

The new instance.

8.278.2.2 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
A generic interface class for a ink annotation. It is intended that future releases of JawsMako will extend this interface.

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

Inheritance diagram for JawsMako::IInkAnnotation:

```
IROObject
  JawsMako::IAnnotation
    JawsMako::IMarkupAnnotation
      JawsMako::IInkAnnotation
```

Public Member Functions

- `virtual CFPointVecVec getInkList () const =0`
  
  Get the annotation's ink list. All coordinates are relative to the annotation rectangle.

- `virtual void setInkList (const CFPointVecVec &inkList)=0`
  
  Get the annotation's ink list. All coordinates are relative to the annotation rectangle.

Static Public Member Functions

- `static JAWSMAKO_API IInkAnnotationPtr create (const IJawsMakoPtr &jawsMako, const FRect &rect, const IDOMColorPtr &color=IDOMColorPtr(), const CFPointVecVec &inkList=CFPointVecVec())`

  Create an ink annotation.

Additional Inherited Members

8.279.1 Detailed Description

A generic interface class for a ink annotation. It is intended that future releases of JawsMako will extend this interface.
8.279.2 Member Function Documentation

8.279.2.1 create()

```cpp
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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>jawsMako</td>
<td>The JawsMako instance</td>
</tr>
<tr>
<td>rect</td>
<td>The annotations bounds. Must not be empty</td>
</tr>
<tr>
<td>color</td>
<td>Optional; The desired annotation color. If NULL, RGB yellow will be chosen. If provided, the color space must be DeviceRGB, DeviceCMYK or DeviceGray</td>
</tr>
<tr>
<td>inkList</td>
<td>Optional; the ink list to use</td>
</tr>
</tbody>
</table>

**Returns**

`IInkAnnotationPtr` A smart pointer to the new ink annotation

8.279.2.2 getInkList()

```cpp
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)

8.279.2.3 setInkList()

```cpp
virtual void JawsMako::IInkAnnotation::setInkList (const CFPointVectVect & inkList) [pure virtual]
```

Get the annotation's ink list. All coordinates are relative to the annotation rectangle.
Parameters

| inkList | The ink list (a collection of quadpoint data) |

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

- interactive.h

### 8.280 JawsMako::IInput Class Reference

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

```cpp
#include <jawsmako.h>
```

Inheritance diagram for JawsMako::IInput:

![Inheritance Diagram](image)

**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.

**Static Public Member Functions**

- static JAWSMAKO_API IInputPtr `create` (const IJawsMakoPtr &jawsMako, eFileFormat format)
  
  Create an input for reading source documents in the given format. The following formats are currently supported:

**Additional Inherited Members**

### 8.280.1 Detailed Description

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

8.280.2.1 create()

static JAWSMAKO_API IInputPtr JawsMako::IInput::create (  
    const IJawsMakoPtr & jawsMako,  
    eFileFormat format ) [static]

Create an input for reading source documents in the given format. The following formats are currently supported:

- PDF
- XPS
- PCL5
- PCL/XL

Returns
IInputPtr the input

8.280.2.2 open() [1/3]

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.

8.280.2.3 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.
8.280.2.4 `open()` [3/3]

```cpp
virtual IDocumentAssemblyPtr JawsMako::IInput::open (const IInputStreamPtr & inputStream) [pure virtual]
```

Open a stream, returning the IDocumentAssembly representing the contents.

Returns

IDocumentAssemblyPtr the document assembly.

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

- jawsmako.h

---

8.281 `IInputEnum< typename T >` Class Reference

Iterator template class to allow iteration over a collection of instances of type `<T>`

```cpp
#include <iedlenum.h>
```

8.281.1 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

---

8.282 `IInputEnumRC< typename T >` Class Reference

Reference-counted iterator template class to allow iteration over a collection of instances of type `<T>`

```cpp
#include <iedlenum.h>
```

8.282.1 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
Input Stream with pushback support.

#include <edlstream.h>

Inheritance diagram for IInputPushbackStream:

---

Additional Inherited Members

8.283.1 Detailed Description

Input Stream with pushback support.

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

- edlstream.h
Generic input stream. Abstract base class for all input streams.
#include <edlstream.h>

Inheritance diagram for IInputStream:

```
<table>
<thead>
<tr>
<th>IRCObject</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEDLObject</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>IDOMHashable</td>
</tr>
<tr>
<td>IEDLStream</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>InputStream</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>IRAPushbackStream</td>
</tr>
<tr>
<td>IRAInputStream</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>IRAInputPushbackStream</td>
</tr>
</tbody>
</table>
```

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 completeRead (void *buffer, int32 count)
  Perform a complete read.
- 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.
Static Public Member Functions

- static EDL_API IRAInputStreamPtr createFromFile (IEDLClassFactory *pFactory, const EDLSysString &path)
  
  Creation function for an InputStream 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 InputStream 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 InputStream 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 InputStream 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 createFromMemory (IEDLClassFactory *pFactory, const void *mem, uint32 length, bool copy=false, bool free=true)
  
  Creation function for an InputStream for data in memory. Throws an IEDLError exception on failure.

- static EDL_API IInputStreamPtr createFromUserFunc (IEDLClassFactory *pFactory, UserStreamReadFunc readFunc, void *priv)
  
  Creation function for an InputStream from a user function that provides data. Throws an IEDLError exception on failure.

- static EDL_API IRAInputStreamPtr createFromRAUserFunc (IEDLClassFactory *pFactory, int64 length, UserRAReadFunc readFunc, void *priv)
  
  Creation function for an IRAInputStream from a user function that provides random access. Throws an IEDLError exception on failure.

- static EDL_API IRAInputStreamPtr createFromNewFileWithContents (IEDLClassFactory *pFactory, const EDLSysString &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 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 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 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.
8.284 IInputStream Class Reference

Additional Inherited Members

8.284.1 Detailed Description

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

8.284.2 Member Function Documentation

8.284.2.1 completeRead()

virtual bool IInputStream::completeRead ( void * buffer, int32 count ) [virtual]

Perform a complete read.

Parameters

| buffer | Buffer to accept the data |
| count  | Number of bytes to be read into the buffer |

Returns

bool True if the read operation was successful, or false if the read could not be completely fulfilled

8.284.2.2 createCompositeStream()

static EDL_API IInputStreamPtr IInputStream::createCompositeStream ( IEDLClassFactory * pFactory, const CIInputStreamVect & streams ) [static]

Creation routine for creating a composite input stream representing the concatenation of a series of streams.

Parameters

| pFactory | The class factory. |
| streams  | The vector of streams to use. |

Returns

IInputStreamPtr The new input stream.
Creation function for an `IInputStream` for a file on disk. Throws an `IEDLError` exception on failure.

**Parameters**

- `pFactory` The class factory
- `path` Path to the file

**Returns**

- `IRAInputStreamPtr` The new input stream

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

<table>
<thead>
<tr>
<th>pFactory</th>
<th>The class factory</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>Path to the file</td>
</tr>
</tbody>
</table>

Returns

IRAInputStreamPtr The new input stream

---

8.284.2.6 `createFromFileShared()` [2/2]

```cpp
static EDL_API IRAInputStreamPtr IInputStream::createFromFileShared (  
    IEDLClassFactory * pFactory,  
    const EDLString & path ) [static]
```

Creation function for an InputStream 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

<table>
<thead>
<tr>
<th>pFactory</th>
<th>The class factory</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>Path to the file</td>
</tr>
</tbody>
</table>

Returns

IRAInputStreamPtr The new input stream

---

8.284.2.7 `createFromFlateCompressed()`

```cpp
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

<table>
<thead>
<tr>
<th>pFactory</th>
<th>The class factory</th>
</tr>
</thead>
<tbody>
<tr>
<td>stream</td>
<td>The compressed stream.</td>
</tr>
<tr>
<td>raw</td>
<td>Pass to true if the flate stream has no zlib header.</td>
</tr>
<tr>
<td>ignoreChecksums</td>
<td>Pass true if checksum errors should be ignored.</td>
</tr>
</tbody>
</table>
Returns

**IInputStreamPtr** The new input stream

### 8.284.2.8 createFromLz4Compressed()

```cpp
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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pFactory</td>
<td>The class factory</td>
</tr>
<tr>
<td>stream</td>
<td>The compressed stream.</td>
</tr>
</tbody>
</table>

Returns

**IInputStreamPtr** The new input stream

### 8.284.2.9 createFromMemory()

```cpp
static EDL_API IRAInputStreamPtr IInputStream::createFromMemory (  
    IEDLClassFactory * pFactory,
    const void * mem,  
    uint32 length,  
    bool copy = false,  
    bool free = true ) [static]
```

Creation function for an **IInputStream** for data in memory. Throws an **IEDLError** exception on failure.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pFactory</td>
<td>The class factory</td>
</tr>
<tr>
<td>mem</td>
<td>A pointer to the memory to be used as the source data for the stream.</td>
</tr>
<tr>
<td>length</td>
<td>The length of the buffer pointed to by mem.</td>
</tr>
<tr>
<td>copy</td>
<td>If true, a copy of the memory pointed to by mem will be made and will be managed and released by the stream.</td>
</tr>
<tr>
<td>free</td>
<td>If copy is false and free is true, then when the stream is destroyed, mem will be deallocated using stdlib free(). Ignored if copy is true.</td>
</tr>
</tbody>
</table>
Returns

**InputStreamPtr** The new input stream

### 8.284.2.10 `createFromNewFileWithContents()` [1/2]

```c
static EDL_API IRAInputStreamPtr IInputStream::createFromNewFileWithContents (　　IEDLClassFactory ∗ pFactory,　　const EDLSysString & 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**

<table>
<thead>
<tr>
<th><strong>pFactory</strong></th>
<th>The class factory.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>path</strong></td>
<td>Path to the file</td>
</tr>
<tr>
<td><strong>stream</strong></td>
<td>The stream to populate the new file with.</td>
</tr>
</tbody>
</table>

Returns

**InputStreamPtr** The new input stream

### 8.284.2.11 `createFromNewFileWithContents()` [2/2]

```c
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**

<table>
<thead>
<tr>
<th><strong>pFactory</strong></th>
<th>The class factory.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>path</strong></td>
<td>Path to the file</td>
</tr>
<tr>
<td><strong>stream</strong></td>
<td>The stream to populate the new file with.</td>
</tr>
</tbody>
</table>

Returns

**InputStreamPtr** The new input stream
8.284.2.12 createFromRAUserFunc()

static EDL_API IRAInputStreamPtr IInputStream::createFromRAUserFunc ( 
    IEDLClassFactory * pFactory, 
    int64 length, 
    UserRAReadFunc readFunc, 
    void * priv ) [static]

Creation function for an IRAInputStream from a user function that provides random access. Throws an IEDLError exception on failure.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pFactory</code></td>
<td>The class factory.</td>
</tr>
<tr>
<td><code>length</code></td>
<td>The stream length</td>
</tr>
<tr>
<td><code>readFunc</code></td>
<td>A function that when called, will provide the data for the stream.</td>
</tr>
<tr>
<td><code>priv</code></td>
<td>An opaque private pointer that is passed to the readFunc on each call.</td>
</tr>
</tbody>
</table>

Returns

`IInputStreamPtr` The new input stream.

8.284.2.13 createFromUserFunc()

static EDL_API IInputStreamPtr IInputStream::createFromUserFunc ( 
    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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pFactory</code></td>
<td>The class factory.</td>
</tr>
<tr>
<td><code>readFunc</code></td>
<td>A function that when called, will provide the data for the stream.</td>
</tr>
<tr>
<td><code>priv</code></td>
<td>An opaque private pointer that is passed to the readFunc on each call.</td>
</tr>
</tbody>
</table>

Returns

`IInputStreamPtr` The new input stream.

8.284.2.14 createPushbackStream()

static EDL_API IInputPushbackStreamPtr IInputStream::createPushbackStream ( 
    IEDLClassFactory * pFactory, 
    UserPushbackReadFunc readFunc, 
    void * priv ) [static]

Creation function for an IInputPushbackStream from a user function that provides data. Throws an IEDLError exception on failure.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pFactory</code></td>
<td>The class factory.</td>
</tr>
<tr>
<td><code>readFunc</code></td>
<td>A function that when called, will provide the data for the stream.</td>
</tr>
<tr>
<td><code>priv</code></td>
<td>An opaque private pointer that is passed to the readFunc on each call.</td>
</tr>
</tbody>
</table>

Returns

`IInputPushbackStreamPtr` The new input stream.
8.284 IInputStream Class Reference

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

<table>
<thead>
<tr>
<th>pFactory</th>
<th>The class factory.</th>
</tr>
</thead>
<tbody>
<tr>
<td>sourceStream</td>
<td>The source stream to use.</td>
</tr>
<tr>
<td>clone</td>
<td>If true, the input stream is cloned. If false, the existing stream is used in place.</td>
</tr>
</tbody>
</table>

Returns

IInputStreamPtr The new input stream.

8.284.2.15 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

<table>
<thead>
<tr>
<th>pFactory</th>
<th>The class factory</th>
</tr>
</thead>
<tbody>
<tr>
<td>stream</td>
<td>The stream to use for source.</td>
</tr>
<tr>
<td>offset</td>
<td>The offset within the source stream of the start of the data to use.</td>
</tr>
<tr>
<td>length</td>
<td>The length of source data to use.</td>
</tr>
</tbody>
</table>

Returns

IInputStreamPtr The new input stream

8.284.2.16 eof()

virtual bool IInputStream::eof ( ) const [pure virtual]

Determine if the stream has exhausted.

Returns

bool True if the stream has exhausted
8.284.2.17 hash()

virtual bool IInputStream::hash (  
    uint64 & hash ) [virtual]

Obtain a 64-bit hash of the stream. Please note that this requires reading the stream and is therefore not thread safe. If thread safety is desired, make a clone of the stream first.

Parameters

| hash | A reference to receive the hash value |

Returns

bool True if method succeeded

Implements IDOMHashable.

8.284.2.18 read() [1/2]

virtual int32 IInputStream::read (  
    void * buffer,  
    int32 count ) [pure virtual]

Read specified number of bytes from a stream into buffer.

Parameters

| buffer | Buffer to accept the data |
| count  | Number of bytes to read  |

Returns

int32 The number of bytes actually read

8.284.2.19 read() [2/2]

virtual int8 IInputStream::read ( ) [pure virtual]

Read single byte from a stream.

Returns

int8 Number of bytes actually read
8.284.2.20  skip()

virtual int64 IInputStream::skip (  
    int64 count ) [inline], [virtual]  

Skip a specified number of bytes.

Parameters  

| count | Number of bytes to skip |

Returns  

    int64 The number of bytes actually skipped

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

- edlstream.h

8.285  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 &fontIndex)=0
  Find a font from the system fonts, and the font index if applicable.

Static Public Member Functions

- static JAWSMAKO_API IJawsMakoPtr create (const U8String &libDir="", const U8String &tempDir="", const U8String &cacheDir="", const CTemporaryStoreParameters &tempStoreParams = CTemporaryStoreParameters())
  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. Not available on iOS or Android.
- static JAWSMAKO_API void enablePDFOutput (const IJawsMakoPtr &jawsMako)
  Enable PDF Output for JawsMako. Requires the following additional libraries to be linked on iOS.
- static JAWSMAKO_API void enablePSOutput (const IJawsMakoPtr &jawsMako)
  Enable PostScript Output for JawsMako. Requires the following additional libraries to be linked on iOS.
- static JAWSMAKO_API void enableAllFeatures (const IJawsMakoPtr &jawsMako)
  Enable all JawsMako features.

Additional Inherited Members

8.285.1 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.

8.285.2 Member Function Documentation

8.285.2.1 create()

static JAWSMAKO_API IJawsMakoPtr JawsMako::IJawsMako::create {
    const U8String & libDir = U8String(""),
    const U8String & tempDir = U8String(""),
    const U8String & cacheDir = U8String(""),
    const CTemporaryStoreParameters & tempStoreParams = CTemporaryStoreParameters{} }

[static]

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>libDir</td>
<td>the absolute path to the directory containing the support libraries (ie the corerip and edlpspdfout libraries and executables). May be set to an empty string if these components are not used (such as on iOS or Android).</td>
</tr>
<tr>
<td>tempDir</td>
<td>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.</td>
</tr>
<tr>
<td>cacheDir</td>
<td>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.</td>
</tr>
<tr>
<td>tempStoreParams</td>
<td>The desired temporary store parameters. See #CTemporaryStoreParams for details.</td>
</tr>
</tbody>
</table>

Returns

IJawsMakoPtr the new Instance.

8.285.2.2 enablePDFOutput()

static JAWSMAKO_API void JawsMako::IJawsMako::enablePDFOutput ( const IJawsMakoPtr & jawsMako ) [static]

Enable PDF Output for JawsMako. Requires the following additional libraries to be linked on iOS.

- libedlpspdfout (iOS only)

8.285.2.3 enablePSOutput()

static JAWSMAKO_API void JawsMako::IJawsMako::enablePSOutput ( const IJawsMakoPtr & jawsMako ) [static]

Enable PostScript Output for JawsMako. Requires the following additional libraries to be linked on iOS.

- libedlpspdfout (iOS only)

8.285.2.4 findFont()

virtual IDOMFontPtr JawsMako::IJawsMako::findFont ( const U8String & fontName, uint32 & 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.

Generated by Doxygen
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fontName</td>
<td>The font name of the desired font.</td>
</tr>
<tr>
<td>fontIndex</td>
<td>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.</td>
</tr>
</tbody>
</table>

Returns

IDOMFontPtr The found font.

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

- jawsmako.h

8.286 JawsMako::IJawsRenderer Class Reference

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

#include <jawsmako.h>

Inheritance diagram for JawsMako::IJawsRenderer:

```
IRObj
```

Classes

- class CSpotHalftone
  
  Description of a simple spot halftone, at 45 degrees, using Jaws's default spot function.

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

- class CThresholdHalftone
  
  A halftone representing a simple threshold.

- class IHalftone
  
  An abstract base class for communicating halftone information to the Jaws renderer, for use with renderMonochrome() and renderMonochromeToFrameBuffer()
Public Member Functions

- virtual IDOMImagePtr render (const IDOMNodePtr &node, uint32 dpi=72, const IDOMColorSpacePtr &colorSpace=IDOMColorSpacePtr(NULL), 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 IDOMColorSpacePtr &colorSpace=IDOMColorSpacePtr(NULL), bool antiAliased=false, const FRect &bounds=FRect(), uint32 destWidth=0, uint32 destHeight=0, const CSpotColorNames &retainedSpotColors=CSpotColorNames(), const IOptionalContentPtr &optionalContent=IOptionalContentPtr(), eOptionalContentEvent optionalContentUsage=eOCEView)=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 renderToFrameBuffer (const IDOMNodePtr &node, void *buffer, uint32 stride, uint32 destWidth, uint32 destHeight, const IDOMColorSpacePtr &colorSpace=IDOMColorSpacePtr(NULL), 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(NULL), const FRect &bounds=FRect(), const IOptionalContentPtr &optionalContent=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 void renderAntiAliasedToFrameBuffer (const IDOMNodePtr &node, void *frameBuffer, int32 stride, uint32 width, uint32 height, uint32 dpi=72, uint8 quality=3, const FRect &bounds=FRect())=0

  EXPERIMENTAL: Render a node to an antialiased, 8-bit RGBA (DeviceRGB) frame buffer, suitable for display. Does not use full supersampling as renderAntiAliased(), but uses a faster method.

- virtual IDOMImagePtr renderMonochrome (const IDOMNodePtr &node, uint32 dpi=72, const IHalftone *halftone=NULL, const IDOMColorSpacePtr &colorSpace=IDOMColorSpacePtr(NULL), const FRect &bounds=FRect())=0

  Render a node to a 1-bit monochrome image.

- virtual void renderMonochromeToFrameBuffer (const IDOMNodePtr &node, void *buffer, uint32 stride, uint32 destWidth, uint32 destHeight, const IHalftone *halftone, const IDOMColorSpacePtr colorSpace=IDOMColorSpacePtr(NULL), const FRect &bounds=FRect(), const IOptionalContentPtr &optionalContent=IOptionalContentPtr(), eOptionalContentEvent optionalContentUsage=eOCEView)=0

  Render a node to a 1-bit monochrome frame buffer.

Static Public Member Functions

- static JAWSMAKO_API IJawsRendererPtr create (const IJawsMakoPtr &jawsMako)

  Create a Jaws renderer instance.

Additional Inherited Members

8.286.1 Detailed Description

A renderer that uses the Jaws RIP to create images from arbitrary DOM.
8.286.2 Member Function Documentation

8.286.2.1 create()

```cpp
static JAWSMAKO_API IJawsRendererPtr JawsMako::IJawsRenderer::create (
    const IJawsMakoPtr & jawsMako ) [static]
```

Create a Jaws renderer instance.

Parameters

- **JawsMako** The JawsMako instance.

Returns

IJawsRendererPtr A jaws renderer instance.

8.286.2.2 render()

```cpp
virtual IDOMImagePtr JawsMako::IJawsRenderer::render (
    const IDOMNodePtr & node,
    uint32 dpi = 72,
    const IDOMColorSpacePtr & colorSpace = IDOMColorSpacePtr(NULL),
    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

<table>
<thead>
<tr>
<th>node</th>
<th>The node to render.</th>
</tr>
</thead>
<tbody>
<tr>
<td>dpi</td>
<td>The desired resolution.</td>
</tr>
<tr>
<td>colorSpace</td>
<td>The desired colour space. If NULL, DeviceRGB will be used. If provided, this must be an instance of one of the following colorspace types:</td>
</tr>
<tr>
<td></td>
<td>• DeviceRGB,</td>
</tr>
<tr>
<td></td>
<td>• sRGB,</td>
</tr>
<tr>
<td></td>
<td>• DeviceGray,</td>
</tr>
<tr>
<td></td>
<td>• sGray,</td>
</tr>
<tr>
<td></td>
<td>• DeviceCMYK,</td>
</tr>
<tr>
<td></td>
<td>• ICCBased,</td>
</tr>
</tbody>
</table>
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bounds</td>
<td>The desired rendering area. If the rect is empty an appropriate size is chosen based on the input node.</td>
</tr>
<tr>
<td>retainedSpotColorNames</td>
<td>Used only if the color space is a three or four channel type (i.e., 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.</td>
</tr>
<tr>
<td>optionalContent</td>
<td>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.</td>
</tr>
<tr>
<td>optionalContentUsage</td>
<td>The desired usage for optional content. Ignored if #optionalContent is NULL. eOCEUnknown must not be used.</td>
</tr>
</tbody>
</table>

Returns

IDOMImagePtr the rendered image.

8.286.2.3 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>node</td>
<td>The node to render.</td>
</tr>
<tr>
<td>dpi</td>
<td>The desired resolution.</td>
</tr>
<tr>
<td>quality</td>
<td>The desired anti-aliasing quality, from 0 (no antialiasing) to 15 (maximum).</td>
</tr>
</tbody>
</table>

Returns

IDOMImagePtr the rendered image.

8.286.2.4 renderAntiAliasedToFrameBuffer()

virtual void JawsMako::IJawsRenderer::renderAntiAliasedToFrameBuffer ( const IDOMNodePtr & node, void * frameBuffer, int32 stride,
EXPERIMENTAL: Render a node to an antialiased, 8-bit RGBA (DeviceRGB) frame buffer, suitable for display. Does not use full supersampling as `renderAntiAliased()`, but uses a faster method.

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>node</td>
<td>The node to render.</td>
</tr>
<tr>
<td>frameBuffer</td>
<td>The destination frame buffer</td>
</tr>
<tr>
<td>stride</td>
<td>The offset in bytes from one frame buffer scanline to the next, which may be negative</td>
</tr>
<tr>
<td>width</td>
<td>The frame buffer width, in pixels</td>
</tr>
<tr>
<td>height</td>
<td>The frame buffer height, in pixels</td>
</tr>
<tr>
<td>dpi</td>
<td>The desired resolution.</td>
</tr>
<tr>
<td>quality</td>
<td>The desired anti-aliasing quality, from 1 (low quality) to 15 (maximum).</td>
</tr>
<tr>
<td>bounds</td>
<td>The desired rendering area. If the rect is empty an appropriate size is chosen based on the input node.</td>
</tr>
</tbody>
</table>

---

### 8.286.2.5 renderMonochrome()

```cpp
def renderMonochrome(self, node: IDOMNodePtr, dpi: int = 72, halftone: IHalftone = NULL, colorSpace: IDOMColorSpacePtr = IDOMColorSpacePtr(NULL), bounds: FRect = FRect()) -> IDOMImagePtr:
```

Render a node to a 1-bit monochrome image.

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>node</td>
<td>The node to render.</td>
</tr>
<tr>
<td>dpi</td>
<td>The desired resolution.</td>
</tr>
<tr>
<td>halftone</td>
<td>The Halftone to use. If NULL, a 60lpi spot function will be used.</td>
</tr>
<tr>
<td>colorSpace</td>
<td>The desired colour space. If NULL, DeviceGray will be used. If provided, only DeviceGray or sGray may be used.</td>
</tr>
<tr>
<td>bounds</td>
<td>The desired rendering area. If the rect is empty an appropriate size is chosen based on the input node.</td>
</tr>
</tbody>
</table>

### Returns

`IDOMImagePtr` the rendered image.
8.286.2.6 renderMonochromeToFrameBuffer()

virtual void JawsMako::IJawsRenderer::renderMonochromeToFrameBuffer (  
    const IDOMNodePtr & node,  
    void * buffer,  
    uint32 stride,  
    uint32 destWidth,  
    uint32 destHeight,  
    const IHalftone & halftone,  
    const IDOMColorSpacePtr colorSpace = IDOMColorSpacePtr(NULL),  
    const FRect & bounds = FRect(),  
    const IOptionalContentPtr & optionalContent = IOptionalContentPtr(),  
    eOptionalContentEvent optionalContentUsage = eOCEView ) [pure virtual]

Render a node to a 1-bit monochrome frame buffer.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>node</td>
<td>The node to render.</td>
</tr>
<tr>
<td>buffer</td>
<td>The target frame buffer.</td>
</tr>
<tr>
<td>stride</td>
<td>The distance in bytes between the start of one scanline in the buffer to the next scanline in y.</td>
</tr>
<tr>
<td>halftone</td>
<td>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).</td>
</tr>
<tr>
<td>colorSpace</td>
<td>The desired colour space. If NULL, DeviceGray will be used. If provided, only DeviceGray or sGray may be used.</td>
</tr>
<tr>
<td>bounds</td>
<td>The desired rendering area. If the rect is empty an appropriate size is chosen based on the input node.</td>
</tr>
<tr>
<td>optionalContentUsage</td>
<td>The desired usage for optional content. Ignored if optionalContent is NULL. eOCEUnknown must not be used.</td>
</tr>
</tbody>
</table>

8.286.2.7 renderSeparations()

virtual CEDLVector<IDOMImagePtr> JawsMako::IJawsRenderer::renderSeparations (  
    const IDOMNodePtr & node,  
    const IDOMColorSpacePtr & colorSpace = IDOMColorSpacePtr(NULL),  
    bool antiAliased = false,  
    const FRect & bounds = FRect(),  
    uint32 destWidth = 0,  
    uint32 destHeight = 0,  
    const CSpotColorNames & retainedSpotColors = CSpotColorNames(),  
    const IOptionalContentPtr & optionalContent = IOptionalContentPtr(),  
    eOptionalContentEvent optionalContentUsage = eOCEView ) [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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>node</td>
<td>The node to render.</td>
</tr>
</tbody>
</table>
Parameters

<table>
<thead>
<tr>
<th><code>colorSpace</code></th>
<th>The desired colour space. If NULL, DeviceRGB will be used. If provided, this must be an instance of one of the following colorspace types:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• DeviceRGB,</td>
</tr>
<tr>
<td></td>
<td>• sRGB,</td>
</tr>
<tr>
<td></td>
<td>• DeviceGray,</td>
</tr>
<tr>
<td></td>
<td>• sGray,</td>
</tr>
<tr>
<td></td>
<td>• DeviceCMYK,</td>
</tr>
<tr>
<td></td>
<td>• ICCBased,</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><code>bounds</code></th>
<th>The desired rendering area. If the rect is empty an appropriate size is chosen based on the input node.</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>destWidth</code></td>
<td>The desired output width, in pixels. If zero, the input size is used.</td>
</tr>
<tr>
<td><code>destHeight</code></td>
<td>The desired output height, in pixels. If zero, the input size is used.</td>
</tr>
<tr>
<td><code>retainedSpotColorNames</code></td>
<td>Used only if the color space is a three or four channel type (i.e. some form of RGB or CMYK color space). If provided, the renderer will produce separate images for these colorants.</td>
</tr>
<tr>
<td><code>optionalContent</code></td>
<td>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.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><code>optionalContentUsage</code></th>
<th>The desired usage for optional content. Ignored if <code>#optionalContent</code> is NULL.</th>
</tr>
</thead>
</table>

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.

8.286.2.8 renderToFrameBuffer()

virtual void JawsMako::IJawsRenderer::renderToFrameBuffer (  
const IDOMNodePtr & node,  
void * buffer,  
uint32 stride,  
uint32 destWidth,  
uint32 destHeight,  
const IDOMColorSpacePtr & colorSpace = IDOMColorSpacePtr(NULL),  
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

<table>
<thead>
<tr>
<th><code>node</code></th>
<th>The node to render.</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>buffer</code></td>
<td>The target frame buffer.</td>
</tr>
</tbody>
</table>
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>destWidth</td>
<td>The desired output width in pixels. The input content will be scaled to fit this size.</td>
</tr>
<tr>
<td>destHeight</td>
<td>The desired output height in pixels. The input content will be scaled to fit this size.</td>
</tr>
<tr>
<td>stride</td>
<td>The distance in bytes between the start of one scanline in the buffer to the next scanline in y.</td>
</tr>
<tr>
<td>colorSpace</td>
<td>The desired colour space. If NULL, DeviceRGB will be used. If provided, this must be an instance of one of the following colorspace types:</td>
</tr>
<tr>
<td></td>
<td>• DeviceRGB,</td>
</tr>
<tr>
<td></td>
<td>• sRGB,</td>
</tr>
<tr>
<td></td>
<td>• DeviceGray,</td>
</tr>
<tr>
<td></td>
<td>• sGray,</td>
</tr>
<tr>
<td></td>
<td>• DeviceCMYK,</td>
</tr>
<tr>
<td></td>
<td>• ICCBased,</td>
</tr>
<tr>
<td>bounds</td>
<td>The desired rendering area. If the rect is empty an appropriate size is chosen based on the input node.</td>
</tr>
<tr>
<td>optionalContent</td>
<td>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.</td>
</tr>
<tr>
<td>optionalContentUsage</td>
<td>The desired usage for optional content. Ignored if #optionalContent is NULL. eOCEUnknown must not be used.</td>
</tr>
</tbody>
</table>

Returns

IDOMImagePtr the rendered image.

8.286.2.9 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(NULL), 
    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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>node</td>
<td>The node to render.</td>
</tr>
<tr>
<td>buffer</td>
<td>The target frame buffer.</td>
</tr>
<tr>
<td>destWidth</td>
<td>The desired output width in pixels. The input content will be scaled to fit this size.</td>
</tr>
<tr>
<td>destHeight</td>
<td>The desired output height in pixels. The input content will be scaled to fit this size.</td>
</tr>
<tr>
<td>stride</td>
<td>The distance in bytes between the start of one scanline in the buffer to the next scanline in y.</td>
</tr>
<tr>
<td>pixelStride</td>
<td>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 colour space.</td>
</tr>
<tr>
<td>reverseComponents</td>
<td>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.</td>
</tr>
<tr>
<td>colorSpace</td>
<td>The desired colour space. If NULL, DeviceRGB will be used. If provided, this must be an instance of one of the following colorspace types:</td>
</tr>
<tr>
<td></td>
<td>• DeviceRGB,</td>
</tr>
<tr>
<td></td>
<td>• sRGB,</td>
</tr>
<tr>
<td></td>
<td>• DeviceGray,</td>
</tr>
<tr>
<td></td>
<td>• sGray,</td>
</tr>
<tr>
<td></td>
<td>• DeviceCMYK,</td>
</tr>
<tr>
<td></td>
<td>• ICCBased,</td>
</tr>
<tr>
<td>bounds</td>
<td>The desired rendering area. If the rect is empty an appropriate size is chosen based on the input node.</td>
</tr>
<tr>
<td>optionalContent</td>
<td>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.</td>
</tr>
<tr>
<td>optionalContentUsage</td>
<td>The desired usage for optional content. Ignored if #optionalContent is NULL. eOCEUnknown must not be used.</td>
</tr>
</tbody>
</table>

**Returns**

IDOMImagePtr the rendered image.

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

- jawsmako.h

8.287 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:

![Inheritance Diagram](image)

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 setInteriorsColor (const IDOMColorPtr &color)=0
  
  Set the interior color to be used to fill the line endings.

Additional Inherited Members

8.287.1 Detailed Description

An interface class for a line annotation. It is intended that future releases of JawsMako will extend this interface.
8.287.2 Member Function Documentation

8.287.2.1 getCaptionOffset()

virtual void JawsMako::ILineAnnotation::getCaptionOffset ( 
    float & xOffset, 
    float & yOffset ) const [pure virtual]

Get the caption offset.

Parameters

| xOffset | The xOffset value |
| yOffset | The yOffset value |

8.287.2.2 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

8.287.2.3 getLeaderLineExtensionsLength()

virtual float JawsMako::ILineAnnotation::getLeaderLineExtensionsLength ( ) const [pure virtual]

Get the length of the leader line extensions.

Returns

    float The leader line extensions length
8.287.2.4 getLeaderLineLength()

virtual float JawsMako::ILineAnnotation::getLeaderLineLength ( ) const [pure virtual]

Get the length of the leader line.

Returns

float The leader line length

8.287.2.5 getLeaderLineOffset()

virtual float JawsMako::ILineAnnotation::getLeaderLineOffset ( ) const [pure virtual]

Get the leader line offset.

Returns

float The leader line offset

8.287.2.6 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

<table>
<thead>
<tr>
<th>start</th>
<th>Start point</th>
</tr>
</thead>
<tbody>
<tr>
<td>end</td>
<td>End point</td>
</tr>
</tbody>
</table>

8.287.2.7 setInteriorColor()

virtual void JawsMako::ILineAnnotation::setInteriorColor ( 
    const IDOMColorPtr & color ) [pure virtual]

Set the interior color to be used to fill the line endings.
8.287.2.8 setLineEndpoints()

`virtual void JawsMako::ILineAnnotation::setLineEndpoints (
    const FPoint & start,
    const FPoint & end ) [pure virtual]`

Set the line end points. The points are relative to the annotation rect.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>start</code></td>
<td>Start point</td>
</tr>
<tr>
<td><code>end</code></td>
<td>End point</td>
</tr>
</tbody>
</table>

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

- interactive.h

8.288 JawsMako::ILinkAnnotation Class Reference

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

```cpp
#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.

Additional Inherited Members

8.288.1 Detailed Description

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

8.288.2 Member Function Documentation

8.288.2.1 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

8.288.2.2 setQuadPoints()

virtual void JawsMako::ILinkAnnotation::setQuadPoints ( 
    const CQuadPointVect &quadPoints ) [pure virtual]

Set the link annotation's quad points.

Parameters

quadPoints The link annotation's quad points

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

- interactive.h

Generated by Doxygen
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.

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.

Additional Inherited Members

8.289.1 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
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 IRCObjectPtr getProperties () const =0
  
  Obtain the properties, if present. This is currently a private structure.

- virtual bool getIsPoint () const =0
  
  Do the marked content details represent a single point?

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

8.290.1 Detailed Description

Details of Marked Content applied to an IDOMGroup.

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

- structure.h
8.291  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:

![Inheritance Diagram](image)

Public Member Functions

- virtual IStructureElementReferencePtr getStructureElementReference () const =0
  
  Obtain a reference for the structure element that this content refers to. Will never return NULL.

Static Public Member Functions

- static IMarkedContentStructureDetailsPtr create (const IJawsMakoPtr &jawsMako, const U8String &tag, const IRCObjectPtr &properties, bool isPoint, const IStructureElementReferencePtr &elementReference)
  
  Create structure marked content details.

Additional Inherited Members

8.291.1  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
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.
Additional Inherited Members

8.292.1 Detailed Description

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

8.292.2 Member Function Documentation

8.292.2.1 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.

8.292.2.2 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.

8.292.2.3 getOpacity()

virtual float JawsMako::IMarkupAnnotation::getOpacity ( ) const [pure virtual]

Get the opacity of the markup annotation.

Returns

float The opacity value
8.292.2.4 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.

8.292.2.5 setAuthor()

virtual void JawsMako::IMarkupAnnotation::setAuthor(const U8String & author) [pure virtual]

Set the Author of the markup annotation. This is the "T" entry in the annotation, which by convention is the author.

Parameters

| author | The author to set. An empty string will cause the author to be removed entirely. |

8.292.2.6 setCreationTime()

virtual void JawsMako::IMarkupAnnotation::setCreationTime(const IEDLTimePtr & creationTime) [pure virtual]

Set the creation date and time of the annotation.

Parameters

| creationTime | The creation date and time. Must not be NULL. |

8.292.2.7 setOpacity()

virtual void JawsMako::IMarkupAnnotation::setOpacity(float opacity) [pure virtual]

Set the opacity of the markup annotation.
Parameters

| opacity | A value between 0.0 (fully transparent) and 1.0 (fully opaque) |

8.292.2.8  setPopup() [1/2]

```
virtual void JawsMako::IMarkupAnnotation::setPopup (  
   const IAnnotationReferencePtr & popupReference  )  [pure virtual]
```

Set or clear the popup, by reference.

Parameters

| popupReference | A reference to the popup. Set to NULL to remove the current popup. |

For full functionality, the popup must be added to the current page’s annotations.

8.292.2.9  setPopup() [2/2]

```
virtual void JawsMako::IMarkupAnnotation::setPopup (  
   const IPopupAnnotationPtr & popup  )  [pure virtual]
```

Set a reference to a popup, if present.

Parameters

| popup | The popup. Set to NULL to remove the current popup. |

For full functionality, the popup must be added to the current page’s annotations.

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

- interactive.h

8.293  JawsMako::INamedDestination Class Reference

A named destination in a PDF Document.

```
#include <interactive.h>
```
Inheritance diagram for JawsMako::INamedDestination:

```
IRCOBJECT

JawsMako::INamedDestination
```

Public Member Functions

- virtual INamedDestinationPtr clone () const =0
  
  Clone the named destination.

- virtual const U8String & getName () const =0
  
  Get the name of this named destination.

- virtual IDOMPageRectTargetPtr getTarget () const =0
  
  Get the destination.

Static Public Member Functions

- static JAWSMAKO_API INamedDestinationPtr create (IJawsMakoPtr &jawsMako, const U8String &name, const IDOMPageRectTargetPtr &target)
  
  Create a named destination with the given name to the given target.

Additional Inherited Members

8.293.1 Detailed Description

A named destination in a PDF Document.

8.293.2 Member Function Documentation

8.293.2.1 clone()

```
virtual INamedDestinationPtr JawsMako::INamedDestination::clone () const [pure virtual]
```

Clone the named destination.

Returns

INamedDestinationPtr A smart pointer to the cloned named destination
8.293.2.2 create()

```cpp
static JAWSMAKO_API INamedDestinationPtr JawsMako::INamedDestination::create ( 
    IJawsMakoPtr & jawsMako,
    const U8String & name,
    const IDOMPageRectTargetPtr & target ) [static]
```

Create a named destination with the given name to the given target.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>jawsMako</code></td>
<td>The JawsMako instance</td>
</tr>
<tr>
<td><code>name</code></td>
<td>The name of the destination</td>
</tr>
<tr>
<td><code>target</code></td>
<td>The target of the destination</td>
</tr>
</tbody>
</table>

**Returns**

INamedDestinationPtr A smart pointer to the new named destination

8.293.2.3 getName()

```cpp
virtual const U8String & JawsMako::INamedDestination::getName ( ) const [pure virtual]
```

Get the name of this named destination.

**Returns**

U8String The name of this named destination

8.293.2.4 getTarget()

```cpp
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`

Generated by Doxygen
JawsMako::IOptionalContent Class Reference

Root level optional content information for an entire document.

#include <optionalcontent.h>

Inheritance diagram for JawsMako::IOptionalContent:

```
IRObject

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, bool visible)
  * Make the given DOM node optional with a new optional content group.
- virtual bool groupIsVisible (const IOptionalContentGroupPtr &group, eOptionalContentEvent event=eOCEView, IOptionalContentConfigurationPtr configuration=IOptionalContentConfigurationPtr())=0
  * Determine if the given optional content group is visible in the given circumstances.
- virtual bool groupIsVisible (const IOptionalContentGroupReferencePtr &groupRef, eOptionalContentEvent event=eOCEView, IOptionalContentConfigurationPtr configuration=IOptionalContentConfigurationPtr())=0
  * Determine if the given optional content group is visible in the given circumstances, using the group's reference.
- virtual bool groupIsPresent (const IOptionalContentGroupReferencePtr &groupRef) const =0
  * Is a group with the given reference present?
- virtual IOptionalContentGroupPtr getGroup (const IOptionalContentGroupReferencePtr &groupRef) const =0
  * Get the optional content group with the given reference. An exception will be thrown if the group is not found.
- virtual void forceGroupState (const IOptionalContentGroupPtr &group, bool visible)=0
  * Force the given group to be always on or off in the default configuration. Convenience.
- 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.
• 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.

**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**

**8.294.1 Detailed Description**

Root level optional content information for an entire document.

**8.294.2 Member Function Documentation**

**8.294.2.1 addGroup()**

virtual void JawsMako::IOptionalContent::addGroup {
  const IOptionalContentGroupPtr & group,
  const IDocumentPtr & sourceDocument ) [pure virtual]

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 behaviour as closely as possible.
8.294.2.2 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.
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, and remove any usage information from the group.

8.294.2.3 groupIsVisible() [1/2]

virtual bool JawsMako::IOptionalContent::groupIsVisible (  
    const IOptionalContentGroupPtr & group,  
    eOptionalContentEvent event = eOCEView,  
    IOptionalContentConfigurationPtr configuration = IOptionalContentConfigurationPtr()  
) [pure virtual]

Determine if the given optional content group is visible in the given circumstances.

**Parameters**

<table>
<thead>
<tr>
<th>group</th>
<th>The group to check.</th>
</tr>
</thead>
<tbody>
<tr>
<td>event</td>
<td>The event representing the intended use. Pass eOCEUnknown if the automatic state IOptionalContentGroupUsageApplication should not be checked.</td>
</tr>
<tr>
<td>configuration</td>
<td>The configuration to use. Pass NULL to use the default configuration.</td>
</tr>
</tbody>
</table>

8.294.2.4 groupIsVisible() [2/2]

virtual bool JawsMako::IOptionalContent::groupIsVisible (  
    const IOptionalContentGroupReferencePtr & groupRef,  
    eOptionalContentEvent event = eOCEView,  
    IOptionalContentConfigurationPtr configuration = IOptionalContentConfigurationPtr()  
) [pure virtual]

Determine if the given optional content group is visible in the given circumstances, using the group's reference.

**Parameters**

| groupRef | Reference to the group to check. |
| event | The event representing the intended use. Pass eOCEUnknown if the automatic state IOptionalContentGroupUsageApplication should not be checked. |
| configuration | The configuration to use. Pass NULL to use the default configuration. |
8.294.2.5 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 its 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.

8.294.2.6 makeNodeOptionalWithNewGroup()

```
IDOMNodePtr JawsMako::IOptionalContent::makeNodeOptionalWithNewGroup ( 
    const IDOMNodePtr & node,
    const U8String & newGroupName,
    bool visible ) [inline]
```

Make the given DOM node optional with a new optional content group.

Convenience.

8.294.2.7 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`
A configuration for optional content.

#include <optionalcontent.h>

Inheritance diagram for JawsMako::IOptionalContentConfiguration:

- 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.

### Static Public Member Functions

• static JAWSMAKO_API IOptionalContentConfigurationPtr create (const IJawsMakoPtr &jawsMako)  
  Create a new configuration.
Additional Inherited Members

8.295.1 Detailed Description

A configuration for optional content.

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

- optionalcontent.h

8.296 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:

![Inheritance Diagram](image)

Public Types

- enum eVisibilityPolicy { eVPAllOn, eVPAnyOn, eVPAnyOff, eVPAllOff }

  The visibility policy controlling how the dependent optional content groups affect the visibility of this content.

Public Member Functions

- virtual IOptionalContentDetailsPtr clone () const =0

  Clone.

- virtual bool isVisible (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.

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

8.296.1 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.

8.296.2 Member Enumeration Documentation

8.296.2.1 eVisibilityPolicy

enum JawsMako::IOptionalContentDetails::eVisibilityPolicy

The visibility policy controlling how the dependent optional content groups affect the visibility of this content.
### Enumerator

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eVPAllOn</td>
<td>This content is visible if all the dependent groups are visible.</td>
</tr>
<tr>
<td>eVPAnyOn</td>
<td>This content is visible if any dependent group is visible.</td>
</tr>
<tr>
<td>eVPAnyOff</td>
<td>This content is visible if any dependent group is invisible.</td>
</tr>
<tr>
<td>eVPAllOff</td>
<td>This content is visible if all the dependent groups are invisible.</td>
</tr>
</tbody>
</table>

### 8.296.3 Member Function Documentation

#### 8.296.3.1 getIsVisible()

```cpp
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`

### 8.297 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.

```cpp
#include <transforms.h>
```
Inheritance diagram for JawsMako::IOptionalContentFixerTransform:

```
IRCOObject

JawsMako::ITransform

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.*

**Static Public Member Functions**

- static JAWSMAKO_API IOptionalContentFixerTransformPtr create (const IJawsMakoPtr &jawsMako)
  
  *Create the transform.*

**Additional Inherited Members**

8.297.1 Detailed Description

A simple transform that strips the DOM of any PDF optional content that is not visible for the given document use.

8.297.2 Member Function Documentation

8.297.2.1 create()

```cpp
static JAWSMAKO_API IOptionalContentFixerTransformPtr JawsMako::IOptionalContentFixerTransform::create (const IJawsMakoPtr &jawsMako) [static]
```

Create the transform.
Parameters

| JawsMako | The JawsMako instance. |

Returns

The new instance.

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

- transforms.h

8.298 JawsMako::IOptionalContentGroup Class Reference

Interface for an optional content group.

#include <optionalcontent.h>

Inheritance diagram for JawsMako::IOptionalContentGroup:

8.298 JawsMako::IOptionalContentGroup Class Reference

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.
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

8.298.1 Detailed Description

Interface for an optional content group.

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

- optionalcontent.h

8.299 JawsMako::IOptionalContentGroupReference Class Reference

A reference to an optional content group.

#include <optionalcontent.h>

Inheritance diagram for JawsMako::IOptionalContentGroupReference:

```
IRCObject

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?
Additional Inherited Members

8.299.1 Detailed Description

A reference to an optional content group.

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

- optionalcontent.h

8.300 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:

![Inheritance Diagram](image)

Public Member Functions

- virtual IOptionalContentGroupUsagePtr clone () const =0
  
  Clone the usage information.

- virtual void setLanguage (const U8String &language, bool preferred=false)=0
  
  Set the language of the content controlled by this optional content group, and whether or not it should be considered "preferred" if there is no exact match for the language used by the viewer's.

- virtual U8String getLanguage () const =0
  
  Get the language for the group, which may be an empty string if no language has been set.

- virtual bool 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

Generated by Doxygen
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:

Static Public Member Functions

- static JAWSMAKO_API IOptionalContentGroupUsagePtr create (const IJawsMakoPtr &jawsMako)
  Create usage information.

Additional Inherited Members

8.300.1 Detailed Description

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

These are only consulted when making visibility decisions if the document's IOptionalContent specifies that the usage for this group should be used, and even then only some attributes may be consulted depending on how the usages are configured. See #IOptionalContentGroupUsageApplication below.

8.300.2 Member Function Documentation
8.300.2.1 getUsers()

virtual CU8StringVect JawsMako::IOptionalContentGroupUsage::getUsers ( U8String & type ) const [pure virtual]

Get the type and users for which this content is primarily intended.

Parameters

| type | Reference to receive the type of user. |

Returns

The user names, or an empty array if none are present.

8.300.2.2 recommendedVisibilityForCategories()

virtual eOptionalContentVisibility JawsMako::IOptionalContentGroupUsage::recommendedVisibilityForCategories ( 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.

8.300.2.3 setLanguage()

virtual void JawsMako::IOptionalContentGroupUsage::setLanguage ( const U8String & language, bool preferred = false ) [pure virtual]

Set the language of the content controlled by this optional content group, and whether or not it should be considered "preferred" if there is no exact match for the language used by the viewer's.

Parameters

| language | A "Natural Language Specification" string representing the language and possible a locale, such as "en-US". Pass an empty string to specify no particular language. |
| preferred | Whether this should be considered preferred if no exact match for the user's viewing language is found. |
### 8.300.2.4 setPageElement()

```cpp
virtual void JawsMako::IOptionalContentGroupUsage::setPageElement ( const U8String & type ) [pure virtual]
```

Sets a description of the page element that is represented by the optional content group.

**Parameters**

| type | Should be either "HF" (Header or footer), "FG" (foreground content), "BG" (background content), "L" (Logo) or an empty string indicating no particular page element is represented. |

### 8.300.2.5 setUsers()

```cpp
virtual void JawsMako::IOptionalContentGroupUsage::setUsers ( const U8String & type, const CU8StringVect & users ) [pure virtual]
```

Get the type and users for which this content is primarily intended.

**Parameters**

| type | The type of user, which should be either "Ind" (Individual), "Ttl" (Title), or "Org" (Organisation). Must not be empty unless the users vector is also empty (which will cause the user information to be ignored for visibility determination) |
| users | The user names. |

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

- `optionalcontent.h`

### 8.301 JawsMako::IOptionalContentGroupUsageApplication Class Reference

Interface for controlling how `IOptionalContentGroupUsage` is applied, and for what groups.

```cpp
#include <optionalcontent.h>
```
Inheritance diagram for JawsMako::IOptionalContentGroupUsageApplication:

```
+--------------------------+
| IRCObject               |
|                         |
|                         |  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 optional content usage should be applied. Must not be eOCEUnknown.
- virtual eOptionalContentEvent getEvent () const =0
  Get the event for which the optional content usage should be applied.
- virtual void setAffectedOptionalContentGroups (const COptionalContentGroupReferenceVect &groups)=0
  Set the optional content groups (by reference) for which the usage information should be applied.
- virtual COptionalContentGroupReferenceVect getAffectedOptionalContentGroups () const =0
  Get the optional content groups for which the usage information should be applied.
- virtual void addGroup (const IOptionalContentGroupReferencePtr &groupRef)=0
  Add the given group to the list of affected groups. It should not already be present.
- virtual bool groupIsAffected (const IOptionalContentGroupReferencePtr &groupRef) const =0
  Is the given group affected by this usage application?
- virtual void setCategories (const COCCategoryVect &categories)=0
  Set the categories from the optional content groups that should be consulted to determine visibility.
- virtual COCCategoryVect getCategories () const =0
  Get the categories from the optional content groups that should be consulted to determine visibility.
- virtual bool categoryInCategories (eOptionalContentCategory category) const =0
  Does the usage categories include the given category?
- virtual bool categoriesInCategories (const COCCategoryVect &categories) const =0
  Does the usage categories include all the given categories? Convenience.

Static Public Member Functions

- static JAWSMAKO_API IOptionalContentGroupUsageApplicationPtr create (const IJawsMakoPtr &jawsMako, eOptionalContentEvent event=eOCEView)
  Create usage application information.
Additional Inherited Members

8.301.1 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

8.302 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:

![Inheritance Diagram](image)

**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 bool evaluate (const IOptionalContentPtr &optionalContent, eOptionalContentEvent event) const =0
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

8.302.1 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.

8.302.2 Member Enumeration Documentation

8.302.2.1 eVisibilityExpressionOperation

enum JawsMako::IOptionalContentVisibilityExpression::eVisibilityExpressionOperation

The operation of this expression.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eVEOGroupState</td>
<td>The result of this expression is the state of the single group.</td>
</tr>
<tr>
<td>eVEONot</td>
<td>The result of this expression is the inverse of the sub-expression.</td>
</tr>
<tr>
<td>eVEOAnd</td>
<td>The result of this expression is the logical and of the result of the sub-expressions.</td>
</tr>
<tr>
<td>eVEOOr</td>
<td>The result of this expression is the logical or of the result of the sub-expressions.</td>
</tr>
</tbody>
</table>

8.302.3 Member Function Documentation

Generated by Doxygen
8.302.3.1 evaluate()

virtual bool 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

8.303 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 Types

- typedef void(* ProgressCallbackFunc) (void *priv, float progress)
  
  A callback type for receiving progress information from 0.0 (starting) through 1.0 (complete)
Public Member Functions

- virtual void setPreset (const U8String &preset)=0
  
  Configure the output according to a general preset. Please see the supplied documentation for details of these presets. The default is “Preserve” which will attempt to produce output as close to the input as possible for the output format. A string value can be used for any parameter and will be converted as necessary.

- virtual void setParameter (const U8String &param, const U8String &value)=0
  
  Apply a key value pair output parameter with a string value. The parameter name is case insensitive. Please refer to the supplied documentation for the details of the available parameters and their ranges.

- virtual void setAllowedPermissionsFlags (uint32 allowedPermissions)=0
  
  Control whether or not assemblies with certain security permission flags are allowed to be written by this output.

- virtual void writeAssembly (const IDocumentAssemblyPtr &assembly, const U8String &pathToFile, ProgressCallbackFunc progressFunc=NULL, void *progressPriv=NULL, const IOutputAbortPtr &abort=IOutputAbortPtr())=0

  Write the given document assembly to a file on disk.

- virtual void writeAssembly (const IDocumentAssemblyPtr &assembly, const String &pathToFile, ProgressCallbackFunc progressFunc=NULL, void *progressPriv=NULL, const IOutputAbortPtr &abort=IOutputAbortPtr())=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, ProgressCallbackFunc progressFunc=NULL, void *progressPriv=NULL, const IOutputAbortPtr &abort=IOutputAbortPtr())=0

  Write the given document assembly to a stream.

- virtual IOutputWriterPtr openWriter (const IDocumentAssemblyPtr &assembly, const U8String &pathToFile, const IOutputAbortPtr &abort=IOutputAbortPtr())=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, const IOutputAbortPtr &abort=IOutputAbortPtr())=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, const IOutputAbortPtr &abort=IOutputAbortPtr())=0

  Create an output writer for the given assembly, targeting a stream.

Static Public Member Functions

- static JAWSMAKO_API IOutputPtr create (const IJawsMakoPtr &jawsMako, eFileFormat format)

  Create an output for writing source in the given format.

Additional Inherited Members

8.303.1 Detailed Description

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

8.303.2 Member Function Documentation
8.303.2.1 create()

```cpp
static JAWSMAKO_API IOutputPtr JawsMako::IOutput::create (
    const IJawsMakoPtr & jawsMako,
    eFileFormat format ) [static]
```

Create an output for writing source in the given format.

The following formats are currently supported:

- PDF
- XPS
- PostScript
- PCL/XL
- PCL5e
- PCL5c

Returns

IOutputPtr the new output

8.303.2.2 openWriter() [1/3]

```cpp
virtual IOutputWriterPtr JawsMako::IOutput::openWriter ( 
    const IDocumentAssemblyPtr & assembly,
    const U8String & pathToFile,
    const IOutputAbortPtr & abort = IOutputAbortPtr() ) [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

<table>
<thead>
<tr>
<th>parameter</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>assembly</td>
<td>The document assembly for output. The assembly need not be populated with documents.</td>
</tr>
<tr>
<td>pathToFile</td>
<td>The path to the output file on disk.</td>
</tr>
<tr>
<td>abort</td>
<td>An optional reference to an IOutputAbort instance that, when, signalled will cause the output operation to abort. When an abort is signalled, writeAssembly() will abort and an IError exception with code EDL_ERR_ABORTED will be thrown. Not all outputs will be able to abort immediately.</td>
</tr>
</tbody>
</table>

Returns

IOutputWriterPtr The new output writer.
8.303.2.3 openWriter() [2/3]

virtual IOutputWriterPtr JawsMako::IOutput::openWriter (  
    const IDocumentAssemblyPtr & assembly,  
    const String & pathToFile,  
    const IOutputAbortPtr & abort = IOutputAbortPtr() ) [pure virtual]

Create an output writer for the given assembly, targeting a file on disk. As above, but with the file specified in a wide character string.

Parameters

| assembly | The document assembly for output. The assembly need not be populated with documents. |
| pathToFile | The path to the output file on disk. |
| abort | An optional reference to an IOutputAbort instance that, when, signalled will cause the output operation to abort. When an abort is signalled, writeAssembly() will abort and an IError exception with code EDL_ERR_ABORTED will be thrown. Not all outputs will be able to abort immediately. |

Returns

IOutputWriterPtr The new output writer.

8.303.2.4 openWriter() [3/3]

virtual IOutputWriterPtr JawsMako::IOutput::openWriter (  
    const IDocumentAssemblyPtr & assembly,  
    const IOutputStreamPtr & stream,  
    const IOutputAbortPtr & abort = IOutputAbortPtr() ) [pure virtual]

Create an output writer for the given assembly, targeting a stream.

Parameters

| assembly | The document assembly for output. The assembly need not be populated with documents. |
| stream | The stream for the output. |
| abort | An optional reference to an IOutputAbort instance that, when, signalled will cause the output operation to abort. When an abort is signalled, writeAssembly() will abort and an IError exception with code EDL_ERR_ABORTED will be thrown. Not all outputs will be able to abort immediately. |

Returns

IOutputWriterPtr The new output writer.

8.303.2.5 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".

8.303.2.6 writeAssembly() [1/3]

virtual void JawsMako::IOutput::writeAssembly (  
    const IDocumentAssemblyPtr & assembly,  
    const U8String & pathToFile,  
    ProgressCallbackFunc progressFunc = NULL,  
    void * progressPriv = NULL,  
    const IOutputAbortPtr & abort = IOutputAbortPtr() ) [pure virtual]

Write the given document assembly to a file on disk.

Parameters

<table>
<thead>
<tr>
<th>pathToFile</th>
<th>The path to the output file on disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>progressCallback</td>
<td>An optional function pointer to be called after each page has been written providing progress information.</td>
</tr>
<tr>
<td>progressPriv</td>
<td>A private pointer passed to the progress callback for each call.</td>
</tr>
<tr>
<td>abort</td>
<td>An optional reference to an IOutputAbort instance that, when, signalled will cause the output operation to abort. When an abort is signalled, writeAssembly() will abort and an IError exception with code EDL_ERR_ABORTED will be thrown.</td>
</tr>
</tbody>
</table>

8.303.2.7 writeAssembly() [2/3]

virtual void JawsMako::IOutput::writeAssembly (  
    const IDocumentAssemblyPtr & assembly,  
    const U8String & pathToFile,  
    ProgressCallbackFunc progressFunc = NULL,  
    void * progressPriv = NULL,  
    const IOutputAbortPtr & abort = IOutputAbortPtr() ) [pure virtual]
const String & pathToFile,
ProgressCallbackFunc progressFunc = NULL,
void * progressPriv = NULL,
const IOutputAbortPtr & abort = IOutputAbortPtr() ) [pure virtual]

Write the given document assembly to a file on disk, specified by a wide character string.

Parameters

<table>
<thead>
<tr>
<th>pathToFile</th>
<th>The path to the output file on disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>progressCallback</td>
<td>An optional function pointer to be called after each page has been written providing progress information.</td>
</tr>
<tr>
<td>progressPriv</td>
<td>A private pointer passed to the progress callback for each call.</td>
</tr>
<tr>
<td>abort</td>
<td>An optional reference to an IOutputAbort instance that, when, signalled will cause the output operation to abort. When an abort is signalled, writeAssembly() will abort and an IError exception with code EDL_ERR_ABORTED will be thrown.</td>
</tr>
</tbody>
</table>

8.303.2.8  writeAssembly()  [3/3]

virtual void JawsMako::IOutput::writeAssembly {
    const IDocumentAssemblyPtr & assembly,
    const IOutputStreamPtr & stream,
    ProgressCallbackFunc progressFunc = NULL,
    void * progressPriv = NULL,
    const IOutputAbortPtr & abort = IOutputAbortPtr() ) [pure virtual]

Write the given document assembly to a stream.

Parameters

<table>
<thead>
<tr>
<th>stream</th>
<th>The stream to write to.</th>
</tr>
</thead>
<tbody>
<tr>
<td>progressCallback</td>
<td>An optional function pointer to be called after each page has been written providing progress information.</td>
</tr>
<tr>
<td>progressPriv</td>
<td>A private pointer passed to the progress callback for each call.</td>
</tr>
<tr>
<td>abort</td>
<td>An optional reference to an IOutputAbort instance that, when, signalled will cause the output operation to abort. When an abort is signalled, writeAssembly() will abort and an IError exception with code EDL_ERR_ABORTED will be thrown.</td>
</tr>
</tbody>
</table>

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

- jawsmako.h

8.304  JawsMako::IOutputAbort Class Reference

A simple class, usable with IOutput and IOutputWriter to signal an abort of output.

#include <jawsmako.h>
Inheritance diagram for JawsMako::IOutputAbort:

![Inheritance diagram for JawsMako::IOutputAbort](image)

**Public Member Functions**

- virtual void `signalAbort ()=0`
  
  *Signal that the output operation should be aborted.*

- virtual bool `abortSignalled ()=0`
  
  *Determine if the abort has been signalled.*

**Static Public Member Functions**

- static `IOutputAbortPtr create ()`

  *Create an abort signal.*

**Additional Inherited Members**

**8.304.1 Detailed Description**

A simple class, usable with `IOutput` and `IOutputWriter` to signal an abort of output.

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

- `jawsmako.h`
Interface class representing a PDF output intent.

```cpp
#include <outputintent.h>
```

Inheritance diagram for JawsMako::IOutputIntent:

![Inheritance Diagram](image)

**Public Member Functions**

- virtual IOutputIntentPtr `clone ()=0`
  
  Clone the output intent.

- virtual `U8String getSubtype () const =0`
  
  Retrieve the subtype.

- virtual `U8String getOutputCondition () const =0`
  
  Retrieve the output condition, if present. An empty string is returned if there is no output condition provided.

- virtual `U8String getOutputConditionIdentifier () const =0`
  
  Retrieve the output condition identifier.

- virtual `U8String getRegistryName () const =0`
  
  Retrieve the registry name, if present. An empty string is returned if there is no registry name provided.

- virtual `U8String getInfo () const =0`
  
  Retrieve the output intent information string, if present. An empty string is returned if there is no info string provided.

- virtual `IDOMICCProfilePtr getProfile () const =0`
  
  Retrieve the output ICC profile if provided. An empty string is returned if there is no output profile provided.

- virtual `RawString getCheckSum () const =0`
  
  Retrieve the 16-byte MD5 checksum of the profile, if provided. An empty string is returned if there is no checksum.

- virtual `CU8StringVect getColorantTable () const =0`
  
  Retrieve the colorant table for the output intent, if provided. An empty string is returned if there is no colorant table.

- virtual `uint32 getProfileVersion () const =0`
  
  Retrieve the ICC profile version, if provided, as a 32 bit value. 0 will be returned if the version is not provided.

- virtual `RawString getProfileColorSpaceSignature () const =0`
  
  Retrieve the four-byte color space signature of the IFF profile, if provided. Otherwise, an empty string will be returned.

- virtual `U8String getProfileName () const =0`
  
  Retrieve the profile name, if provided. Otherwise, an empty string will be returned.

- virtual `CFileSpecVect getProfileFileSpecifications () const =0`
  
  Retrieve any file specifications referring to external profiles (IFileSpecAsUrl) or embedded profiles (IFileSpecAsEmbeddedData).
Static Public Member Functions

- static JAWSMAKO_API IOutputIntentPtr create (const IJawsMakoPtr &jawsMako, const U8String &subtype, const U8String &outputCondition, const U8String &outputConditionIdentifier, const U8String &registryName, const U8String &info, const IDOMICCProfilePtr &profile=IDOMICCProfilePtr(), const RawString &checkSum=RawString(), const CU8StringVect &colorantTable=CU8StringVect(), uint32 profileVersion=0, const RawString &profileCS=RawString(), const U8String &profileName=U8String(), const CFileSpecVect &fileSpecs=CFileSpecVect())

Create an IOutputIntent.

Additional Inherited Members

8.305.1 Detailed Description

Interface class representing a PDF output intent.

8.305.2 Member Function Documentation

8.305.2.1 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

8.305.2.2 getProfileFileSpecifications()

virtual CFFileSpecVect 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
8.306  IOutputStream Class Reference

Generic output stream. Abstract base class for output streams.

#include <edlstream.h>

Inheritance diagram for IOutputStream:

```
IRCOObject
   |
   v
IEdLObject
   |
   v
IEdlStream
   |
   v
IOutputStream
   |
   v
IraOutputStream
```

Public Member Functions

- virtual int32 write (const char *str)
  
  Perform a write.

- virtual EDL_API int32 writeFormatted (const char *fmt,...)

  Perform a formatted write as per fprintf()

- virtual bool completeWrite (const void *buffer, int32 count)

  Perform a complete write.

Static Public Member Functions

- static EDL_API IOutputStreamPtr 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 IOutputStreamPtr 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 createFromUserFunc (IEDLClassFactory *pFactory, UserStreamWriteFunc 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 IOutputStream &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 IOutputStream &stream)

  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

8.306.1 Detailed Description

Generic output stream. Abstract base class for output streams.

8.306.2 Member Function Documentation

8.306.2.1 completeWrite()

virtual bool IOutputStream::completeWrite (  
    const void * buffer, 
    int32 count ) [virtual]

Perform a complete write.

Parameters

<table>
<thead>
<tr>
<th>buffer</th>
<th>Address of buffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>Number of bytes to be written</td>
</tr>
</tbody>
</table>

Returns

bool True if the write operation was successful, or false if the write could not be completely fulfilled
8.306.2.2  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

<table>
<thead>
<tr>
<th>inStream</th>
<th>The source stream</th>
</tr>
</thead>
<tbody>
<tr>
<td>outStream</td>
<td>The destination stream</td>
</tr>
</tbody>
</table>

Returns

int64 The number of bytes copied

8.306.2.3  createFromUserFunc()

static EDL_API IOutputStreamPtr IOutputStream::createFromUserFunc (  
    IEDLClassFactory * pFactory,  
    UserStreamWriteFunc writeFunc,  
    void * priv ) [static]

Creation function for an IOutputStream from a user function that provides data. Throws an IEDLError exception on failure.

Parameters

| pFactory | The class factory. |
| writeFunc | A function that when called, will write the data for the stream. |
| priv | An opaque private pointer that is passed to the readFunc on each call. |

Returns

IOutputStreamPtr The new input stream

8.306.2.4  createToFile() [1/2]

static EDL_API IOutputStreamPtr 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

<table>
<thead>
<tr>
<th>pFactory</th>
<th>The EDL class factory</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>Path to the file</td>
</tr>
<tr>
<td>append</td>
<td>Specify what to do if the file exists. Optional</td>
</tr>
<tr>
<td></td>
<td>• If false the file will be overwritten by new content. Default</td>
</tr>
</tbody>
</table>

• If true new content will be added to the end of the file

Returns

IOutputStreamPtr The new output stream

8.306.2.5 createToFile() [2/2]

static EDL_API IOutputStreamPtr 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

<table>
<thead>
<tr>
<th>pFactory</th>
<th>The EDL class factory</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>Path to the file</td>
</tr>
<tr>
<td>append</td>
<td>Specify what to do if the file exists. Optional</td>
</tr>
<tr>
<td></td>
<td>• If false the file will be overwritten by new content. Default</td>
</tr>
</tbody>
</table>

• If true new content will be added to the end of the file

Returns

IOutputStreamPtr The new output stream

8.306.2.6 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pFactory</td>
<td>The class factory</td>
</tr>
<tr>
<td>stream</td>
<td>The compressed stream.</td>
</tr>
<tr>
<td>compressionLevel</td>
<td>Values are in the range 1 (fastest, least compression) through 9 (slowest, most compression)</td>
</tr>
<tr>
<td>raw</td>
<td>Pass true if the flat stream should have no zlib header.</td>
</tr>
</tbody>
</table>

Returns

IOutputStreamPtr The new output stream

8.306.2.7 createToLz4Compressed()

static EDL_API IOutputStreamPtr IOutputStream::createToLz4Compressed (  
    IEDLClassFactory ∗pFactory,
    const IOutputStreamPtr &stream) [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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pFactory</td>
<td>The class factory</td>
</tr>
<tr>
<td>stream</td>
<td>The compressed stream.</td>
</tr>
</tbody>
</table>

Returns

IOutputStreamPtr The new output stream

8.306.2.8 write()

virtual int32 IOutputStream::write (  
    const char ∗str) [inline], [virtual]

Perform a write.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>str</td>
<td>C-style string that contains the text to be written to the output</td>
</tr>
</tbody>
</table>

Returns

int32 On success, the total number of characters written
8.306.2.9  writeFormatted()

virtual EDL_API int writeFormatted ( const char * fmt, ... ) [virtual]

Perform a formatted write as per fprintf()

Parameters

<table>
<thead>
<tr>
<th>fmt</th>
<th>C-style string that contains the text to be written to the output</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>Additional arguments as specified by fmt</td>
</tr>
</tbody>
</table>

Returns

int32 On success, the total number of characters written

8.306.2.10  writeStream()

static EDL_API int64 writeStream ( const IInputStreamPtr & inStream, const IOutputStreamPtr & outStream ) [static]

Write the contents of the given stream to an output stream. Opens and closes the input, but does not open or close the output. Throws an IEDLError exception on failure.

Parameters

<table>
<thead>
<tr>
<th>inStream</th>
<th>The source stream</th>
</tr>
</thead>
<tbody>
<tr>
<td>outStream</td>
<td>The destination stream</td>
</tr>
</tbody>
</table>

Returns

int64 The number of bytes written.

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

- edstream.h

8.307  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:

![Inheritance diagram]

**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.

**Additional Inherited Members**

8.307.1 Detailed Description

A writer for writing individual pages and documents to an output in piecemeal fashion.

8.307.2 Member Function Documentation

8.307.2.1 **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()**.

8.307.2.2 **writePage()**

virtual void JawsMako::IOutputWriter::writePage (  
  const IPagePtr & page ) [pure virtual]

Write a page to the output.

**beginDocument()** must have been called to begin the document before any pages may be written. The page may be discarded after this is called.
Parameters

| page | The page to write. |

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

- jawsmako.h

### 8.308 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.

```cpp
#include <transforms.h>
```

Inheritance diagram for JawsMako::IOverprintSimulationTransform:

```
IRCOBJECT

<table>
<thead>
<tr>
<th>JawsMako::ITransform</th>
</tr>
</thead>
<tbody>
<tr>
<td>JawsMako::IOverprintSimulationTransform</td>
</tr>
</tbody>
</table>
```

### 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 **setSimulateBlackDeviceGrayTextOverprint** (bool simulate)=0

  Set whether or not overprint simulation should be performed for 100% black DeviceGray text.
Additional Inherited Members

8.308.1 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 colours will be converted to DeviceCMYK colour space. Use an IColorConverterTransform after this transform if a different output colour space is required.

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.

8.308.2 Member Function Documentation

8.308.2.1 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.

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

- transforms.h

8.309 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 IFixedPage, loading it from the source document if necessary.

- virtual IDOMFixedPagePtr edit ()=0
  Mark the page as edited, returning the editable IFixedPage. Do this before attempting to edit the page contents. The markup DOM will be set to incomplete and may be edited in arbitrary ways. This also prevents the fixed page from being purged in low memory conditions. Note: This member may return a different instance than returned by previous invocations of getContent(), however once this routine has been called, subsequent calls to getContent() will return the same instance as that returned here.

- virtual void revert ()=0
  Revert any edits to a page and mark the page as non-edited if possible.

- virtual bool isLoaded ()=0
  Has the page been loaded? That is, has the DOM been fetched?

- virtual bool isEdited ()=0
  Has the page been edited? That is, has the DOM been fetched?

- virtual void setMetadataChanged () const =0
  Let the page know that the content metadata has changed. For example if either of the MediaBox, CropBox, TrimBox or ArtBox has changed.

- virtual void setContent (const IDOMFixedPagePtr &content)=0
  Replace the content with the given fixed page.

- virtual void release ()=0
  Release the reference to the page content, if possible. If a page is not edited, and the original source is available, the IPage's reference to the page content will be released. This is useful to release the memory associated with the page content if a page is not to be accessed again. The content may be safely requested again using getContent().

- virtual CAnnotationVect getAnnotations ()=0
  Get the annotations for the page, as a vector.

- virtual void addAnnotation (const IAnnotationPtr &annotation)=0
  Add the given annotation.

- virtual IAnnotationPtr findAnnotation (const IAnnotationReferencePtr &reference)=0
  Find the annotation with the given annotation reference within the page. Throws an IError if the target could not be found.

- virtual void removeAnnotation (uint32 index)=0
  Remove the annotation at the given index.

- virtual void removeAnnotation (const IAnnotationPtr &annotation)=0
  Remove the given annotation from the page. If the page is not present, an exception will be thrown.

- virtual void removeAnnotations ()=0
Remove all the annotations from the page.

- virtual void getTextRuns (CTextRunVect &runs)=0
  
  Get the page's text runs. Text present in Tiling brushes will not be returned.

- virtual IDOMCatalogPtr getCatalog () const =0
  
  Get the IDOMCatalog associated with this page.

- virtual DOMid getPageId ()=0
  
  Get the Id of the page.

- virtual COutputIntentVect getOutputIntents ()=0
  
  Get the output intents, if present.

- virtual IDOMJobTkPtr getJobTicket () const =0
  
  Get the page job ticket, if present.

- virtual void setJobTicket (const IDOMJobTkPtr &jobTicket)=0
  
  Set the page job ticket.

- virtual IPagePtr clone ()=0
  
  Clone an IPage. For edited pages, this will clone the DOM tree.

### Static Public Member Functions

- static JAWSMAKO_API IPagePtr create (const IJawsMakoPtr &jawsMako)

  Create an empty page.

### Additional Inherited Members

### 8.309.1 Detailed Description

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

### 8.309.2 Member Function Documentation

#### 8.309.2.1 clone()

```cpp
virtual IPagePtr JawsMako::IPage::clone () [pure virtual]
```

Clone an IPage. For edited pages, this will clone the DOM tree.

Returns

IPagePtr the fixed page.
### 8.309.2.2 create()

```cpp
class JAWSMAKO_API IPagePtr JawsMako::IPage::create (
    const IJawsMakoPtr & jawsMako ) [static]
```

Create an empty page.

Returns

- `IPagePtr` the new page.

### 8.309.2.3 getContent()

```cpp
class virtual IDOMFixedPagePtr JawsMako::IPage::getContent ( ) [pure virtual]
```

Return a smart pointer to the `IFixedPage`, loading it from the source document if necessary.

Returns

- `IDOMFixedPagePtr` the fixed page.

### 8.309.2.4 getJobTicket()

```cpp
class 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

### 8.309.2.5 release()

```cpp
class 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.

### 8.309.2.6 removeAnnotation()

```cpp
class virtual void JawsMako::IPage::removeAnnotation ( uint32 index ) [pure virtual]
```

Remove the annotation at the given index.
Parameters

| index | The index of the annotation to be removed (0 being the first document). |

8.309.2.7 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.

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

• jawsmako.h

8.310 JawsMako::IPageCropperTransform Class Reference

Very simple transform for cropping pages to one of the standard boxes.

#include <transforms.h>

Inheritance diagram for JawsMako::IPageCropperTransform:

![Inheritance Diagram](image)

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.
Static Public Member Functions

- static IPageCropperTransformPtr create (const IJawsMakoPtr &jawsMako)

Create the transform.

Additional Inherited Members

8.310.1 Detailed Description

Very simple transform for cropping pages to one of the standard boxes.

8.310.2 Member Function Documentation

8.310.2.1 create()

static IPageCropperTransformPtr JawsMako::IPageCropperTransform::create ( const IJawsMakoPtr & jawsMako ) [static]

Create the transform.

Parameters

| JawsMako | The JawsMako instance. |

Returns

The new instance.

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

- transforms.h

8.311 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:

```
IRCOBJECT

JawsMako::IPageLayout
```

Public Member Functions

- virtual IDOMFixedPagePtr getFixedPage () const =0
  
  Get the FixedPage being processed.

- virtual void analyze (ePageAnalysis analysisToPerform=ePAAll)=0
  
  Process the page find the blocks of text. Can optionally perform each analysis phase independently (which can be useful when debugging).

- virtual String getLayoutInfo () const =0
  
  Get a textual description of the page content, useful for debugging purposes.

- virtual IPageLayoutDataPtr getLayoutData () const =0
  
  Get a processed representation of the page content.

- virtual IPageLayoutNodeCollection getLayoutNodeCollection () const =0
  
  Get a flat collection of page content nodes.

- virtual String getPageText () const =0
  
  Return all page text.

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

8.311.1 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.
8.311.2 Member Function Documentation

8.311.2.1 create() [1/2]

```cpp
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

<table>
<thead>
<tr>
<th>pFactory</th>
<th>The factory to use.</th>
</tr>
</thead>
<tbody>
<tr>
<td>page</td>
<td>The fixed page to be analysed</td>
</tr>
</tbody>
</table>

Returns

`IPageLayoutPtr` The new page layout analyser

8.311.2.2 create() [2/2]

```cpp
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

<table>
<thead>
<tr>
<th>pFactory</th>
<th>The factory to use</th>
</tr>
</thead>
<tbody>
<tr>
<td>runs</td>
<td>The array of text runs to be analysed</td>
</tr>
</tbody>
</table>

Returns

`IPageLayoutPtr` The new page layout analyser

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

- `text.h`

8.312 JawsMako::IPageLayoutData Class Reference

Provides a representation of the analyzed page layout by organizing and allowing access to collections of `IPageLayoutNodes`. 

Generated by Doxygen
#include <text.h>

Inheritance diagram for JawsMako::IPageLayoutData:

```
IRObject

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 does not include the root item)

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

8.312.1 Detailed Description

Provides a representation of the analyzed page layout by organizing and allowing access to collections of IPageLayoutNodes.

8.312.2 Member Function Documentation

8.312.2.1 create()

static JAWSMAKO_API IPay not shown
### Parameters

| data  | The collection of IPageLayoutNodes |

### Returns

IPageLayoutDataPtr The new representation

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

- text.h

### 8.313 JawsMako::IPageLayoutNode Class Reference

Simple data type representing a part of an analyzed page.

```c
#include <text.h>
```

Inheritance diagram for JawsMako::IPageLayoutNode:

```
IRCObject
```

#### 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 `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.*

#### Static Public Member Functions

- static `JAWSMAKO_API IPageLayoutNodePtr create (FRect rect, ePageLayoutType type, String content, uint32 columnNumber)`
  
  *Creation function for an IPageLayoutNode, a simple data type representing a part of an analyzed page.*
8.313.1 Detailed Description

Simple data type representing a part of an analyzed page.

8.313.2 Member Function Documentation

8.313.2.1 create()

static JAWSMAKO_API IPageLayoutNodePtr JawsMako::IPageLayoutNode::create (FRect rect, ePageLayoutType type, String content, uint32 columnNumber) [static]

Creation function for an IPageLayoutNode, a simple data type representing a part of an analyzed page.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rect</td>
<td>The bounds of the text area</td>
</tr>
<tr>
<td>type</td>
<td>The layout node type</td>
</tr>
<tr>
<td>content</td>
<td>The Unicode string content</td>
</tr>
<tr>
<td>columnNumber</td>
<td>The zero-indexed column number</td>
</tr>
</tbody>
</table>

Returns

IPageLayoutNodePtr The new node

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

- text.h

8.314 JawsMako::IPCL5Input Class Reference

An instance of the JawsMako PCL5 input class.

#include <pcl5input.h>
Public Member Functions

- virtual void setRopResolution (uint32 resolution)=0
  Set the resolution to be used when flattening ROPs.
- virtual void setDefaultPaperSize (const U8String &paperSize)=0
  Set the default paper size.
- virtual void setDefaultLandscape (bool landscape)=0
  Set the default orientation.
- virtual void setDefaultCopies (uint32 copies)=0
  Set the default number of per-page copies.
- virtual void setDefaultDuplex (bool duplex)=0
  Set whether or not duplex should be set by default.
- virtual void setDefaultDuplexBindingMode (IPJLParser::eDuplexBindingMode bindingMode)=0
  Set the default binding edge for duplexing.
- virtual void setDefaultManualFeed (bool manualFeed)=0
  Set whether or not manual feed should be set by default.
- virtual void setDefaultsFromPjl (IPJLParserPtr &pjlParser)=0
  Take initialisation data from the given PJL parser.
- virtual void enableUnencapsulatedMode (bool unencapsulated)=0
  Enable or disable unencapsulated mode.

Static Public Member Functions

- static JAWSMAKO_API IPCL5InputPtr create (const IJawsMakoPtr &jawsMako)
  Create an input for reading source documents in PCL5 format.

Additional Inherited Members

8.314.1 Detailed Description

An instance of the JawsMako PCL5 input class.
8.314.2 Member Function Documentation

8.314.2.1 create()

static JAWSMAKO_API IPCL5InputPtr JawsMako::IPCL5Input::create (const IJawsMakoPtr & jawsMako) [static]

Create an input for reading source documents in PCL5 format.

Returns
IPCL5InputPtr the PCL5 input

8.314.2.2 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.

8.314.2.3 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.
8.314.2.4 setDefaultDuplex()

virtual void JawsMako::IPCL5Input::setDefaultDuplex (  
   bool duplex ) [pure virtual]

Set whether or not duplex should be set by default.

The default is false.

8.314.2.5 setDefaultDuplexBindingMode()

virtual void JawsMako::IPCL5Input::setDefaultDuplexBindingMode (  
   IPJLParser::eDuplexBindingMode bindingMode ) [pure virtual]

Set the default binding edge for duplexing.

The default is long edge.

8.314.2.6 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.

8.314.2.7 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.
8.314.2.8  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
- jsb4
- jsb5
- a5
- a6
- tabloid
- ofuku

The matching is case-insensitive.

The default page size is a4

8.314.2.9  setDefaultsFromPjl()

virtual void JawsMako::IPCL5Input::setDefaultsFromPjl (  
    IPJLParserPtr & pjlParser ) [pure virtual]

Take initialisation data from the given PJL parser.

Uses the environment in the given PJL parser to set defaults in the PCL5 input. Any previously set defaults will be ignored.

This is especially useful with unencapsulated mode, where PJL parsing happens externally to the input.

8.314.2.10  setRopResolution()

virtual void JawsMako::IPCL5Input::setRopResolution (  
    uint32 resolution ) [pure virtual]

Set the resolution to be used when flattening ROPs.

PCL5, as PCL/XL features a method of using bitwise operations to composite pixels from patterns and images with content already on the page. These are known as ROPs (Raster OPerations). These are not compatible with the DOM and most other PDLs, and in many situations require rendering at the input stage.

This parameter controls the rendering resolution. The default is 600dpi.
Parameters

| resolution | The ROP rendering resolution to use. |

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

- pcl5input.h

8.315 JawsMako::IPCL5Output Class Reference

Interface for the PCL5 IOutput class.

```
#include <pcl5output.h>
```

Inheritance diagram for JawsMako::IPCL5Output:

```
+------------------+
| IRCObject        |
|                  |
+------------------+
|                  |
| JawsMako::IOutput|
+------------------+
|                  |
+------------------+
|                  |
| JawsMako::IPCL5Output |
+------------------+
```

Public Types

- enum ePCL5Version
  
  Supported versions.
- enum ePCL5ImageCompression
  
  Available image compression schemes.
Public Member Functions

- virtual void setVersion (ePCL5Version version)=0
  
  Set the PCL version. The default is ePCL5c. Equivalent to calling setParameter() with the parameter name "PCL5 Version" using the string value "pcl5e" or "pcl5c".

- virtual void 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 "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.

Static Public Member Functions

- static IPCL5OutputPtr create (const IJawsMakoPtr &jawsMako)
  
  Create a PCL5 Output instance.

Additional Inherited Members

8.315.1 Detailed Description

Interface for the PCL5 IOutput class.

8.315.2 Member Function Documentation

8.315.2.1 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".

8.315.2.2 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".

Generated by Doxygen
Parameters

- **compression**: The compression method to use.

8.315.2.3 `setMediaSource()`

```cpp
template 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 "MediaSource".

Parameters

- **source**: The media source to use.

8.315.2.4 `setOpenStream()`

```cpp
template 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".

8.315.2.5 `setResolution()`

```cpp
template 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

- **version**: The version to use.
8.315.2.6  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 "PC5L5Version" using the string value "pcl5e" or "pcl5c".

Parameters

version       The version to use.

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

• pcl5output.h

8.316  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 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 (IPJLParse::eDuplexBindingMode bindingMode)=0
  Set the default binding edge for duplexing.
• virtual void setDefaultManualFeed (bool manualFeed)=0
  Set whether or not manual feed should be set by default.
• virtual void setDefaultsFromPjl (IPJLParsePtr &pjlParser)=0
  Take initialisation data from the given PJL parser.
• virtual void enableUnencapsulatedMode (bool unencapsulated)=0
  Enable or disable unencapsulated mode.

Static Public Member Functions

• static JAWSMAKO_API IPCLXLInputPtr create (const IJawsMakoPtr &jawsMako)
  Create an input for reading source documents in PCL/XL format.

Additional Inherited Members

8.316.1 Detailed Description

An instance of the JawsMako PCL/XL input class.

8.316.2 Member Function Documentation

8.316.2.1 create()

static JAWSMAKO_API IPCLXLInputPtr JawsMako::IPCLXLInput::create (const IJawsMakoPtr &jawsMako) [static]

Create an input for reading source documents in PCL/XL format.

Returns

IPCLXLInputPtr the PCL/XL input
8.316.2.2 enableUnencapsulatedMode()

```cpp
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.

8.316.2.3 setDefaultCopies()

```cpp
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.

8.316.2.4 setDefaultDuplex()

```cpp
virtual void JawsMako::IPCLXLInput::setDefaultDuplex ( bool duplex ) [pure virtual]
```

Set whether or not duplex should be set by default.

The default is false.
8.316.2.5 setDefaultDuplexBindingMode()

virtual void JawsMako::IPCLXLInput::setDefaultDuplexBindingMode ( 
    IPJLParse::eDuplexBindingMode bindingMode ) [pure virtual]

Set the default binding edge for duplexing.
The default is long edge.

8.316.2.6 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.

8.316.2.7 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.

8.316.2.8 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
The matching is case-insensitive.

The default page size is a4

8.316.2.9 setDefaultsFromPjl()

virtual void JawsMako::IPCLXLInput::setDefaultsFromPjl (IPJLParserPtr & pjlParser) [pure virtual]

Take initialisation data from the given PJL parser.

Uses the environment in the given PJL parser to set defaults in the PCL/XL input. Any previously set defaults will be ignored.

This is especially useful with unencapsulated mode, where PJL parsing happens externally to the input.

8.316.2.10 setRopResolution()

virtual void JawsMako::IPCLXLInput::setRopResolution (uint32 resolution) [pure virtual]

Set the resolution to be used when flattening ROPs.

PCL/XL features a method of using bitwise operations to composite pixels from patterns and images with content already on the page. These are known as ROPs (Raster OPerations). These are not compatible with the DOM and most other PDLs, and in many situations require rendering at the input stage.

This parameter controls the rendering resolution. The default is 600dpi.

Parameters

| resolution | The ROP rendering resolution to use. |

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

Generated by Doxygen
8.317  JawsMako::IPCLXLOutput Class Reference

Interface for the PCLXL IOutput class.

#include <pclxloutput.h>

Inheritance diagram for JawsMako::IPCLXLOutput:

```
<table>
<thead>
<tr>
<th>IRCObject</th>
</tr>
</thead>
<tbody>
<tr>
<td>JawsMako::IOutput</td>
</tr>
<tr>
<td>JawsMako::IPCLXLOutput</td>
</tr>
</tbody>
</table>
```

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.

Static Public Member Functions

- static IPCLXLOutputPtr create (const IJawsMakoPtr &jawsMako)
  
  Create a PCL5 Output instance.

Additional Inherited Members

8.317.1  Detailed Description

Interface for the PCLXL IOutput class.
8.317.2 Member Function Documentation

8.317.2.1 setEmitPjI()

virtual void JawsMako::IPCLXLOutput::setEmitPjl (bool emitPjl) [pure virtual]

Set whether or not a PJL header or end of job should be emitted.
The default is true.
Equivalent to calling setParameter() with the parameter name "EmitPJL", the value "true" or "false".

8.317.2.2 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

8.318 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 Types

- typedef void(*)(ProgressCallbackFunc) void(priv, uint32 currentPage, uint32 numPages)
  A callback type for receiving progress information in terms of page numbers within a document.

Public Member Functions

- virtual void setPassword(const U8String &password)=0
  Set the password that should be used when attempting to open PDF files using this input. The default is no password.
- 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 CPdfFontInfoVect scanPdfForFonts(const U8String &pathToFile, ProgressCallbackFunc progressFunc=NULL, void *progressPriv=NULL)=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, ProgressCallbackFunc progressFunc=NULL, void *progressPriv=NULL)=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, ProgressCallbackFunc progressFunc=NULL, void *progressPriv=NULL)=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)=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)=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)=0
  Quickly scan the given PDF stream and find any mentions of inks in color spaces.

Static Public Member Functions

- static JAWSMAKO_API IPDFInputPtr create(const IJawsMakoPtr &jawsMako)
  Create an input for reading source documents in PDF format.

Additional Inherited Members

8.318.1 Detailed Description

An instance of the JawsMako PDF input class.
8.318.2 Member Function Documentation

8.318.2.1 create()

```cpp
class JawsMako::IPDFInput

static JAWSMAKO_API IPDFInputPtr JawsMako::IPDFInput::create {
    const IJawsMakoPtr & jawsMako } [static]
```

Create an input for reading source documents in PDF format.

Returns

IPDFInputPtr the PDF input

8.318.2.2 scanPdfForFonts() [1/3]

```cpp
virtual CPdfFontInfoVect JawsMako::IPDFInput::scanPdfForFonts {
    const U8String & pathToFile,
    ProgressCallbackFunc progressFunc = NULL,
    void * progressPriv = NULL } [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.

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.

8.318.2.3 scanPdfForFonts() [2/3]

```cpp
virtual CPdfFontInfoVect JawsMako::IPDFInput::scanPdfForFonts {
    const String & pathToFile,
    ProgressCallbackFunc progressFunc = NULL,
    void * progressPriv = NULL } [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.

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.
8.318.2.4 scanPdfForFonts()

```cpp
virtual CPdfFontInfoVect JawsMako::IPDFInput::scanPdfForFonts (
    const IInputStreamPtr & pdfStream,
    ProgressCallbackFunc progressFunc = NULL,
    void * progressPriv = NULL ) [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.

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.

8.318.2.5 setFailOnFontFallback()

```cpp
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.

8.318.2.6 setPassword()

```cpp
virtual void JawsMako::IPDFInput::setPassword (
    const U8String & password ) [pure virtual]
```

Set the password that should be used when attempting to open PDF files using this input. The default is no password.

Parameters

| password | The password to use. Assumed to be UTF8. |

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

- pdfinput.h
Interface for the PDF IOutput class.

```cpp
#include <pdfoutput.h>
```

Inheritance diagram for JawsMako::IPDFOutput:

```
JawsMako::IOutput
    |   
    |   ↓
    |   JawsMako::IPDFOutput
```

Public Types

- `enum ePDFVersion`:
  - `ePDF1_3 = 3`, `ePDF1_4 = 4`, `ePDF1_5 = 5`, `ePDF1_6 = 6`,
  - `ePDF1_7 = 7`, `ePDF2_0 = 8`, `ePDF1a = 9`, `ePDF1b = 10`,
  - `ePDFX4 = 11` — Supported versions.

- `enum eImageCompression` — Enumeration for image compression formats.

- `enum ePdfXDeviceNHandling` — Flags controlling PDF/X-4 DeviceN handling
  - Allows some control on what the PDF/X writer is allowed to invent, or what inconsistencies are considered safe to ignore.
  - `ePXDNHDefault = 0`, `ePXDNHFailOnMissingColorantInformation = 0x1`,
  - `ePXDNHFailOnInconsistentColorantInformation = 0x2`, `ePXDNHFailIfBetterColorantInformationFoundAfterFirstUse = 0x4`,
  - `ePXDNHFailIfDotGainFound = 0x8`, `ePXDNHFailOnMismatchedDeviceNProcessSpace = 0x10` — Flags controlling PDF/X-4 DeviceN handling.

- `enum ePdfXOptionalContentHandling` — Flags controlling PDF/X-4 Optional content
  - Allows some control on what the PDF/X writer is allowed to invent, or what inconsistencies are considered safe to ignore.
  - `ePXOCHDefault = 0`, `ePXOCHFailOnInconsistentConfigurationName = 0x1`,
  - `ePXOCHFailOnInvalidOrder = 0x2`, `ePXOCHFailIfAutoStateFound = 0x4` — Flags controlling PDF/X-4 Optional content.

- `enum ePdfXExtendedGraphicsStateHandling` — Flags controlling PDF/X-4 Extended Graphics State information
  - Allows some control on what the PDF/X writer is allowed to invent, or what inconsistencies are considered safe to ignore.
  - `ePXEGSHDefault = 0`, `ePXEGSHFailOnTransferFunction = 0x1` — Flags controlling PDF/X-4 Extended Graphics State information.

- `enum eFormCJKCharacterSet` — A CJK character set.
Public Member Functions

• virtual void setVersion (ePDFVersion version)=0
  Set the PDF version to generate.

• virtual void setProducer (const U8String &producer)=0
  Set the Producer name for the output PDF.

• virtual void setEnableIncrementalOutput (bool enable=true)=0
  Set whether or not incremental output should be enabled.

• virtual void 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 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 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::eDownsamplingMethod method=IDOMImageDownsamplerFilter::eBicubic)=0
  Set the desired maximum resolution, threshold and downsampling method for colour images.

• virtual void setGrayImageMaxResolution (float resolution, float threshold=0.0f, IDOMImageDownsamplerFilter::eDownsamplingMethod 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::eDownsamplingMethod 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. The default is 5 - highest quality.

• 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 IDOMFont<OpenType>. The standard 14 fonts are not affected by this; use setEmbedBase14Fonts() 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 setPdfXDeviceNErrorHandling (uint32 flags)=0
  Set flags controlling PDF/X-4 DeviceN handling. See ePdfXDeviceNHandling# for details.
• virtual void setPdfXOptionalContentErrorHandling (uint32 flags)=0
  Set flags controlling PDF/X-4 Optional Content handling. See ePdfXOptionalContentHandling# for details.
• virtual void setPdfXExtendedGraphicsStateErrorHandling (uint32 flags)=0
  Set flags controlling PDF/X-4 Extended Graphics State handling. See ePdfXExtendedGraphicsStateHandling# for details.
• 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.

Static Public Member Functions

• static JAWSMAKO_API IPDFOutputPtr create (const IJawsMakoPtr &jawsMako)
  Create a PDF Output instance.

Additional Inherited Members

8.319.1 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.
• "PDF 1.3" 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.

• "PDF/A-1b" - A preset for generating PDF/A-1b, RGB, with sensible defaults for resolution and downsampling.

• "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 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

8.319.2 Member Enumeration Documentation

8.319.2.1 ePDFVersion

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ePDF1--_3</td>
<td>PDF 1.3.</td>
</tr>
<tr>
<td>ePDF1--_4</td>
<td>PDF 1.4.</td>
</tr>
<tr>
<td>ePDF1--_5</td>
<td>PDF 1.5.</td>
</tr>
<tr>
<td>ePDF1--_6</td>
<td>PDF 1.6.</td>
</tr>
<tr>
<td>ePDF1--_7</td>
<td>PDF 1.7.</td>
</tr>
<tr>
<td>ePDF2--_0</td>
<td>PDF 2.0 (EXPERIMENTAL)</td>
</tr>
<tr>
<td>ePDFA1b</td>
<td>PDF/A-1b.</td>
</tr>
<tr>
<td>ePDFX1a</td>
<td>PDF/X-1a.</td>
</tr>
<tr>
<td>ePDFX4</td>
<td>PDF/X-4.</td>
</tr>
</tbody>
</table>

Generated by Doxygen
8.319.2.2 ePdfXDeviceNHandling

enum JawsMako::IPDFOutput::ePdfXDeviceNHandling

Flags controlling PDF/X-4 DeviceN handling Allows some control on what the PDF/X 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 will be thrown, where the error description will contain a short message outlining the problem that was encountered.

Enumerator

<table>
<thead>
<tr>
<th>ePXDNHDefault</th>
<th>Default; make a best effort attempt to produce.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ePXDNHFailOnMissingColorantInformation</td>
<td>Throw an exception if a spot colorant has.</td>
</tr>
<tr>
<td>ePXDNHFailOnInconsistentColorantInformation</td>
<td>Throw an exception if inconsistent spot colorant.</td>
</tr>
<tr>
<td>ePXDNHFailIfBetterColorantInformationFoundAfterFirstUse</td>
<td>Throw an exception if better spot colorant.</td>
</tr>
<tr>
<td>ePXDNHFailIfDotGainFound</td>
<td>Throw an exception if a dotgain is encountered.</td>
</tr>
<tr>
<td>ePXDNHFailOnMismatchedDeviceNProcessSpace</td>
<td>Throw an exception if a DeviceN Process Property does.</td>
</tr>
</tbody>
</table>

8.319.2.3 ePdfXExtendedGraphicsStateHandling

enum JawsMako::IPDFOutput::ePdfXExtendedGraphicsStateHandling

Flags controlling PDF/X-4 Extended Graphics State information Allows some control on what the PDF/X 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 will be thrown, where the error description will contain a short message outlining the problem that was encountered.

Enumerator

<table>
<thead>
<tr>
<th>ePXEGSHDefault</th>
<th>Default; make a best effort attempt to produce.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ePXEGSHFailOnTransferFunction</td>
<td>Throw an exception if a non-default transfer function is encountered.</td>
</tr>
</tbody>
</table>

8.319.2.4 ePdfXOptionalContentHandling

enum JawsMako::IPDFOutput::ePdfXOptionalContentHandling

Flags controlling PDF/X-4 Optional content Allows some control on what the PDF/X 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 will be thrown, where the error description will contain a short message outlining the problem that was encountered.
Enumerators

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ePXOCHDefault</td>
<td>Default; make a best effort attempt to produce.</td>
</tr>
<tr>
<td>ePXOCHFailOnInconsistentConfigurationName</td>
<td>Throw an exception if an optional content configuration.</td>
</tr>
<tr>
<td>ePXOCHFailOnInvalidOrder</td>
<td>Throw if the Optional Content order information, if present.</td>
</tr>
<tr>
<td>ePXOCHFailIfAutoStateFound</td>
<td>Throw an exception if an optional content configuration contains.</td>
</tr>
</tbody>
</table>

### 8.319.3 Member Function Documentation

#### 8.319.3.1 setAllowRestrictedFonts()

```cpp
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 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".

#### 8.319.3.2 setAlwaysEmbedFonts()

```cpp
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).
8.319.3.3 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.

8.319.3.4 setColorImageMaxResolution()

virtual void JawsMako::IPDFOutput::setColorImageMaxResolution ( 
  float resolution, 
  float threshold = 0.0f, 
  IDOMImageDownsamplerFilter::eDownsamplingMethod method = IDOMImageDownsamplerFilter::eBicubic ) [pure virtual]

Set the desired maximum resolution, threshold and downsampling method for colour images.

Note that if incremental output is used, the content of unedited pages is not affected.

The default behaviour is leave the image resolution unchanged. Equivalent to calling setParameter() with the
param names "ColorImageDownsamplingResolution", "ColorImageDownsamplingThreshold" and "ColorImage
DownsamplingMethod" with the respective values.

Parameters

| resolution | The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default. |
| threshold  | The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution. |
| method     | The method to use when downsampling. The default is bicubic for colour images. |

8.319.3.5 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.

8.319.3.6 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").

8.319.3.7 setConvertAllColors()

virtual void JawsMako::IPDFOutput::setConvertAllColors (  
    bool convert ) [pure virtual]

Set whether or not all content should be color converted to the target space set by setTargetColorSpace().

The default is false.

This parameter is ignored and assumed true when writing to PDF/A-1b or PDF/X-1a format output.

Note that if incremental output is used, the content of unedited pages is not colour converted.

Equivalent to calling setParameter() with the param name "ConvertAllColors" with a boolean string ("true" or "false").

8.319.3.8 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 colorspace.

The default is true.

This parameter is ignored unless setConvertAllColors() is true.

For PDF/X output, this parameter only applies to objects using the DeviceGray colour space. All other gray spaces are subject to the setting of setConvertAllColors().

Note that if incremental output is used, the content of unedited pages is not colour converted.

Equivalent to calling setParameter() with the param name "ConvertGray" with a boolean string ("true" or "false").
8.319.3.9  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".

8.319.3.10  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").

8.319.3.11  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").
8.319.3.12 setEnableIncrementalOutput()

virtual void JawsMako::IPDFOutput::setEnableIncrementalOutput (bool enable = true) [pure virtual]

Set whether or not incremental output should be enabled. The default is false (that is, a full save is performed). Incremental output is only possible when all of the following are true:

- **writeAssembly()** is used (rather than the IOutputWriter interface,
- The assembly has only one document,
- The assembly was opened from a PDF, and
- The PDF was either a file or opened from a random-access stream.
- The PDF version specified is the same version or higher than the input PDF.

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.

8.319.3.13 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

<table>
<thead>
<tr>
<th>keyLength</th>
<th>The key length to use. Must be 0, 40, 128 or 256. 0 indicates that encryption will not be performed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>permissions</td>
<td>The permissions bit mask for the output PDF. (see IDOMStandardPDFSecurityInfo::ePermissionsFlags).</td>
</tr>
<tr>
<td>encryptMetadata</td>
<td>Whether or not to encrypt metadata. The default is false. Must be set to true if a 40 bit key length is specified.</td>
</tr>
<tr>
<td>ownerPassword</td>
<td>The owner password to use.</td>
</tr>
<tr>
<td>userPassword</td>
<td>The user password to use.</td>
</tr>
</tbody>
</table>

8.319.3.14 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 "GrayImageDownsamplingMethod" with the respective values.

Parameters

| resolution | The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default. |
| threshold  | The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution. |
| method     | The method to use when downsampling. The default is bicubic for gray images. |

8.319.3.15 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.

Parameters

| quality | The desired quality level, with 1 being lowest quality and 5 being highest quality. |
8.319.3.16  setMonoImageMaxResolution()

v3rtual 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 "MonoImageDownsamplingMethod" with the respective values.

Parameters

| resolution | The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default. |
| threshold | The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution. |
| method | The method to use when downsampling. The default is subsample for monochrome images; using any other method will result in grayscale output. |

8.319.3.17  setNeverEmbedFonts()

v3rtual 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 "AlwaysEmbed" and a string with the fonts delimited with semicolons (backslashes are used as an escape character).

8.319.3.18  setOutputIntent()

v3rtual 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 output. Please note that as a result of this, PDF 2.0 features are ignored.

For PDF/X-4 the output intent must provide a profile. The target color space is set to this profile.
**setPdfXDeviceNErrorHandling()**

```cpp
virtual void JawsMako::IPDFOutput::setPdfXDeviceNErrorHandling (uint32 flags) [pure virtual]
```

Set flags controlling PDF/X-4 DeviceN handling. See `ePdfXDeviceNHandling` for details.

Equivalent to calling `setParameter()` with the parameter name "PdfXDeviceNErrorHandling" and an unsigned integer string containing the flags value.

**setPdfXExtendedGraphicsStateErrorHandling()**

```cpp
virtual void JawsMako::IPDFOutput::setPdfXExtendedGraphicsStateErrorHandling (uint32 flags) [pure virtual]
```

Set flags controlling PDF/X-4 Extended Graphics State handling. See `ePdfXExtendedGraphicsStateHandling` for details.

Equivalent to calling `setParameter()` with the parameter name "PdfXExtendedGraphicsStateErrorHandling" and an unsigned integer string containing the flags value.

**setPdfXOptionalContentErrorHandling()**

```cpp
virtual void JawsMako::IPDFOutput::setPdfXOptionalContentErrorHandling (uint32 flags) [pure virtual]
```

Set flags controlling PDF/X-4 Optional Content handling. See `ePdfXOptionalContentHandling` for details.

Equivalent to calling `setParameter()` with the parameter name "PdfXOptionalContentErrorHandling" and an unsigned integer string containing the flags value.

**setPreferredColorImageCompression()**

```cpp
virtual void JawsMako::IPDFOutput::setPreferredColorImageCompression (eImageCompression compression) [pure virtual]
```

Set the desired image compression for color images that need to be reencoded. The default is eICAuto.

CCITT is not allowed for colour images.

Note: this is advisory only and may not be honoured in all cases.

Note also that if incremental output is used, the content of unedited pages is not affected. For example, PDF/A does not allow LZW.

Equivalent to calling `setParameter()` with the parameter name "ColorImageCompression" with appropriate values.

**Parameters**

| format | The desired format. |
8.319.3.23 setPreferredGrayImageCompression()

```cpp
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**

```
format  The desired format.
```

8.319.3.24 setPreferredMonoImageCompression()

```cpp
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**

```
format  The desired format.
```

8.319.3.25 setPreferredRenderedImageCompression()

```cpp
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**

| format | The desired format. |

---

### 8.319.3.26 setProducer()

```cpp
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.

### 8.319.3.27 setReencodeImages()

```cpp
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.

Note that if incremental output is used, the content of unedited pages is not affected.

### 8.319.3.28 setRenderResolution()

```cpp
virtual void JawsMako::IPDFOutput::setRenderResolution (uint32 resolution) [pure virtual]
```

Set the resolution to use if page content requires rendering in order to be output. The default is 300dpi. This is affected also by the maximum image resolution parameters. Equivalent to calling `setParameter()` with param name "RenderResolution" with the value as the desired resolution as value.
Parameters

| resolution | The desired resolution in dpi. |

8.319.3.29  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").

8.319.3.30  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 output, an OutputIntents dictionary will be created for this color space. LAB colour spaces may not be used as a destination color space for PDF/X-4. Please note that content using a device space with the same number of components as the output intent will be assumed to be compatible with this space. For example, if a CMYK ICCBased colour space is used for the target space, then any DeviceCMYK content will remain as DeviceCMYK in the output even if the current DeviceCMYK intercept is set to some other profile. Note also that if the output intent is a CMYK color space, then DeviceGray similarly will also be considered equivalent to the K channel of the output intent.

Parameters

| colorSpace | The desired color space. Must be a simple or ICC space. scRGB is not supported for output. |
8.319.3.31  setTargetProfile()

virtual void JawsMako::IPDFOutput::setTargetProfile (  
    const IDOMICCProfilePtr & profile )  [pure virtual]

Set the target color space for the output using an ICC profile.

By default, the target color space is used if anything must be rendered or color-converted. The default behaviour is  
to apply this only when required. Use convertAllColors() if converting everything is required.

Equivalent to calling setParameter() with the param name “TargetProfile” with the value as the path to the profile.

For PDF/A-1b, PDF/X-1a and PDF/X-4 output, the restrictions described in setTargetColorSpace() above apply.

Parameters

profile  The desired profile.

8.319.3.32  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”.

Generated by Doxygen
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
- 1.6
- 1.7
- PDF/A-1b
- PDF/X-1a
- 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.

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

- pdfoutput.h

8.320  JawsMako::IPJLPParser Class Reference

An instance of the Mako PJL Parser.

#include <pj1.h>

Inheritance diagram for JawsMako::IPJLPParser:
Classes

- class CPjlAttributeValue
  A captured PJL attribute.

Public Types

- enum ePjlResult
  Result code for PJL parsing.
- enum eDuplexBindingMode
  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 = <P→ ORTTRAIT 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 BINDER = <LON← GEDGE or SHORTEDGE>".
- virtual void setDefaultManualFeed (bool defaultManualFeed)=0
  Set the default manual feed mode in the PJL Environment. Equivalent to having parsed 
  "@PJL SET MANUALFEED = <ON or OFF>".

Static Public Member Functions

- static JAWSMAKO_API IPJLParserPtr create (const IJawsMakoPtr &jawsMako)
  Create a PJL parser instance.

Additional Inherited Members

8.320.1 Detailed Description

An instance of the Mako PJL Parser.
8.320.2 Member Function Documentation

8.320.2.1 create()

```cpp
static JAWSMAKO_API IPJLParserPtr JawsMako::IPJLParser::create(
    const IJawsMakoPtr & jawsMako) [static]
```

Create a PJL parser instance.

Returns

IPJLParserPtr the PCL/XL input

8.320.2.2 getAttributes()

```cpp
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.

8.320.2.3 parse()

```cpp
virtual ePjlResult JawsMako::IPJLParser::parse(
    const IInputPushbackStreamPtr & stream) [pure virtual]
```

Parse from the given stream, until another language is encountered or the end of the stream is reached.

Exceptions of type IError are thrown on error.

Parameters

| stream | The stream to parse. This must be a pushback-capable stream, as the parser must be able to sniff ahead. The stream must be open; the parser will neither open nor close the stream. |

Returns

ePjlResult The result of parsing.

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

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

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

Inheritance diagram for JawsMako::IPolyAnnotation:

```
IRCOBJECT

JawsMako::IAnnotation

JawsMako::IMarkupAnnotation

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.*

### Additional Inherited Members

#### 8.321.1 Detailed Description

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

#### 8.321.2 Member Function Documentation
8.321.2.1 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

8.321.2.2 setPoints()

virtual void JawsMako::IPolyAnnotation::setPoints ( const CFPointVect & points ) [pure virtual]

Set the points comprising the vertices of the polygon. The points are relative to the annotation rect.

Parameters

  points | The vector of points

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

- interactive.h

8.322  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:

![Inheritance diagram for JawsMako::IPopupAnnotation](image)

**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.

**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**

**8.322.1 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.

**8.322.2 Member Function Documentation**

**8.322.2.1 create()**

```cpp
static JAWSMAKO_API IPopupAnnotationPtr JawsMako::IPopupAnnotation::create (const IJawsMakoPtr &jawsMako, const FRect &rect, bool open = false) [static]
```

Create a popup annotation.
Parameters

<table>
<thead>
<tr>
<th>jawsMako</th>
<th>The JawsMako instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>rect</td>
<td>The annotations bounds. Must not be empty.</td>
</tr>
<tr>
<td>open</td>
<td>Optional; whether or not the note is open. Default is false.</td>
</tr>
</tbody>
</table>

Returns

`IPopupAnnotationPtr` A smart pointer to the new popup annotation

8.322.2.2 `getOpen()`

```cpp
virtual bool JawsMako::IPopupAnnotation::getOpen( ) const [pure virtual]
```

Get the annotation's open status.

Returns

`bool` True if open, false if closed

8.322.2.3 `setOpen()`

```cpp
virtual void JawsMako::IPopupAnnotation::setOpen( 
    bool open ) [pure virtual]
```

Set the annotation's open status.

Parameters

| open | Set to true for open, false for closed |

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

- `interactive.h`

8.323 `JawsMako::IPSInjector 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.

```cpp
#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) [inline], [virtual]
  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.

8.323.1 Detailed Description

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

8.323.2 Member Function Documentation

8.323.2.1 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pageNumber</td>
<td>The page number being prepared, with 0 indicating the first page.</td>
</tr>
<tr>
<td>psStream</td>
<td>The PostScript stream. Do not retain this stream after your handler exits, and only write during the execution of your handler.</td>
</tr>
</tbody>
</table>
8.323.2.2 afterBeginSetup()

virtual void JawsMako::IPSInjector::afterBeginSetup (const IOutputStreamPtr & psStream) [inline], [virtual]

Invoked by the PostScript writer just after %BeginSetup is written to the stream.

Parameters

| psStream | The PostScript stream. Do not retain this stream after your handler exits, and only write during the execution of your handler. |

8.323.2.3 afterLastByte()

virtual void JawsMako::IPSInjector::afterLastByte (const IOutputStreamPtr & psStream) [inline], [virtual]

Invoked by the PostScript writer after the last byte of normal PostScript data is written to the stream.

Parameters

| psStream | The PostScript stream. Do not retain this stream after your handler exits, and only write during the execution of your handler. |

8.323.2.4 beforeEndPageSetup()

virtual void JawsMako::IPSInjector::beforeEndPageSetup (uint32 pageNumber, const IOutputStreamPtr & psStream) [inline], [virtual]

Invoked by the PostScript writer just before %EndPageSetup is written to the stream.

Parameters

| pageNumber | The page number being prepared, with 0 indicating the first page. |
| psStream | The PostScript stream. Do not retain this stream after your handler exits, and only write during the execution of your handler. |

8.323.2.5 beforeEndSetup()

virtual void JawsMako::IPSInjector::beforeEndSetup (const IOutputStreamPtr & psStream) [inline], [virtual]
Invoked by the PostScript writer just before %EndSetup is written to the stream.
### 8.3.2.6 beforeFirstByte()

```cpp
virtual void JawsMako::IPSInjector::beforeFirstByte (  
    const IOutputStreamPtr & psStream ) [inline], [virtual]
```

Invoked by the PostScript writer after the output PostScript stream is opened, but before any data is written to the output stream.

**Parameters**

| psStream | The PostScript stream. Do not retain this stream after your handler exits, and only write during the execution of your handler. |

### 8.3.2.7 beforePsHeader()

```cpp
virtual void JawsMako::IPSInjector::beforePsHeader (  
    const IOutputStreamPtr & psStream ) [inline], [virtual]
```

Invoked by the PostScript writer just before the PostScript header is written to the stream. Currently, this is effectively the same as `beforeFirstByte()` above as Mako will not currently write anything at the top of the PostScript stream before the PostScript header.

**Parameters**

| psStream | The PostScript stream. Do not retain this stream after your handler exits, and only write during the execution of your handler. |

### 8.3.2.8 beforeShowpage()

```cpp
virtual void JawsMako::IPSInjector::beforeShowpage (  
    uint32 pageNumber,  
    const IOutputStreamPtr & psStream ) [inline], [virtual]
```

Invoked by the PostScript writer just before "showpage" is written to the stream.

**Parameters**

| pageNumber | The page number being output, with 0 indicating the first page. |
| psStream   | The PostScript stream. Do not retain this stream after your handler exits, and only write during the execution of your handler. |
The documentation for this class was generated from the following file:

- `psoutput.h`

### 8.324 JawsMako::IPSOutput Class Reference

Interface for the PS IOutput class.

```cpp
#include <psoutput.h>
```

Inheritance diagram for JawsMako::IPSOutput:

```
IRObject

JawsMako::IOutput

JawsMako::IPSOutput
```

#### Public Member Functions

- **virtual void `setResolution`(float resolution)=0**
  
  Set the target resolution for the output, in dots per inch. The default is 600. Equivalent to calling `setParameter` with the parameter name "TargetResolution".

- **virtual void `setStreamingOutput`(bool enable=true)=0**
  
  Set whether or not streaming output should be enabled.

- **virtual void `setTargetColorSpace`(const IDOMColorSpacePtr &targetSpace)=0**
  
  Set the target colour space for the intended output device. The default is DeviceCMYK. Equivalent to calling `setParameter()` with the param name "TargetColorSpace" with appropriate values (please refer to documentation).

- **virtual void `setTargetProfile`(const IDOMICCProfilePtr &profile)=0**
  
  Set the target color space for the intended output device using an ICC profile. Equivalent to calling `setParameter()` with the param name "TargetProfile" with the value as the path to the profile.

- **virtual void `setConvertAllObjectsToTargetColorSpace`(bool convert=true)=0**
  
  Sets whether or not all objects should be converted to the target color space. If false, only objects that require rendering, or use a color space that cannot will represented in PostScript will be forcibly converted to the target colour space. Equivalent to calling `setParameter()` with the param name "ConvertAllObjectsToTargetColorSpace".

- **virtual void `setIgnoreDeviceGrayDuringColorConversion`(bool ignore=true)=0**
  
  Sets whether to ignore DeviceGray objects when color conversion is to be performed. The default is true. Equivalent to calling `setParameter()` with the param name "IgnoreDeviceGrayDuringColorConversion".

- **virtual void `setInjector`(IPSInjector *injector)=0**
  
  Set the IPSInjector instance to use to inject PostScript directly into the output PostScript stream at strategic points in the output process. Note that this routine does not take ownership of the passed pointer, which must remain allocated until after the output is complete.
Static Public Member Functions

- static IPSOutputPtr create (const IJawsMakoPtr &jawsMako)

  Create a PS Output instance.

Additional Inherited Members

8.324.1 Detailed Description

Interface for the PS IOutput class.

Note that there is a limit on the number of PDF and PS output operations that may be in progress simultaneously. These are:

- iOS - 1 PDF or PS output operation at a time
- Android - 1 PDF or PS output operation at a time
- Windows/MacOS/X 32 bit - 8 PDF or PS output operations at a time, subject to available memory
- Windows/MacOS/X 64 bit - 96 PDF or PS output operations at a time for most tool chains
  - VS2015 Static builds are currently limited to 48
- Windows UWP - 1 PDF or PS output operation at a time

8.324.2 Member Function Documentation

8.324.2.1 setConvertAllObjectsToTargetColorSpace()

virtual void JawsMako::IPSOutput::setConvertAllObjectsToTargetColorSpace (bool convert = true) [pure virtual]

Sets whether or not all objects should be converted to the target color space. If false, only objects that require rendering, or use a color space that cannot will represented in PostScript will be forcibly converted to the target colour space. Exquivalent to calling setParameter() with the param name "ConvertAllObjectsToTargetColorSpace". The default is true.

8.324.2.2 setStreamingOutput()

virtual void JawsMako::IPSOutput::setStreamingOutput (bool enable = true) [pure virtual]

Set whether or not streaming output should be enabled.

Streaming output provides output as soon as it is ready, sending it to the output file or stream. Fonts are defined incrementally and a number of standard document structure comments are deferred to the end job trailer. This mode is suitable for use with streaming consumers, such as printers.

If streaming output is set to false, then the output is withheld until the entire assembly has been processed. This enables providing all document resources (such as fonts and color spaces up front), and may be more useful for applications that further process the PostScript. The default is true.

Equivalent to calling setParameter with the parameter name "StreamingOutput".
8.324.3 setTargetColorSpace()

virtual void JawsMako::IPSOutput::setTargetColorSpace (  
  const IDOMColorSpacePtr & targetSpace ) [pure virtual]  

Set the target colour space for the intended output device. The default is DeviceCMYK. Equivalent to calling setParameter() with the param name "TargetColorSpace" with appropriate values (please refer to documentation).

Parameters

| colorSpace  | The desired color space. Must not be LAB, Indexed, DeviceN or scRGB. Any ICC space must have one, three, or four components. |

8.324.4 setTargetProfile()

virtual void JawsMako::IPSOutput::setTargetProfile (  
  const IDOMICCProfilePtr & profile ) [pure virtual]  

Set the target color space for the intended output device using an ICC profile. Equivalent to calling setParameter() with the param name "TargetProfile" with the value as the path to the profile.

Parameters

| profile  | The desired profile. |

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

- psoutput.h

8.325 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 bool pushBack (uint8 byte)=0
  Push back a byte.
- virtual bool pushBack (const void *buffer, int32 count)=0
  Push back from a buffer.

8.325.1 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().

8.325.2 Member Function Documentation

8.325.2.1 pushBack() [1/2]

virtual bool IPushbackStream::pushBack (   
    uint8 byte ) [pure virtual]

Push back a byte.

Parameters

| byte | Byte to be pushed back |

Returns

bool True if method succeeded
virtual bool IPushbackStream::pushBack (const void * buffer, int32 count) [pure virtual]

Push back from a buffer.

<table>
<thead>
<tr>
<th>buffer</th>
<th>Buffer to push back from</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>Number of bytes to push back</td>
</tr>
</tbody>
</table>

Returns

bool True if method succeeded

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

- edlstream.h
Inheritance diagram for IRAInputPushbackStream:

![Inheritance Diagram](image)

### Additional Inherited Members

#### 8.326.1 Detailed Description

Random-access Input Stream with pushback support.

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

- edstream.h

#### 8.327 IRAInputStream Class Reference

Random Access Input Stream.

#include <edlstream.h>
Inheritance diagram forIRAInputStream:

```
IRCOBJECT

IEDLObject

IDOMHashable  IEDLStream

InputStream  IRAStream

IRAInputStream

IRAInputPushbackStream
```

Additional Inherited Members

8.327.1 Detailed Description

Random Access Input Stream.

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

- edlstream.h

8.328 IRAOutputStream Class Reference

Random Access Output Stream.

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

Generated by Doxygen
Inheritance diagram for IRAOutputStream:

```
+----------------+      +----------------+
| IRCObject      |      | IEDLObject      |
|                +----------------+
|                |      | IEDLSStream      |
|                +----------------+
|                |      | IInputStream     |
|                +----------------+
|                |      | IRAStream        |
|                +----------------+
|                |      | IRAOutputStream  |
```

Additional Inherited Members

8.328.1 Detailed Description

Random Access Output Stream.

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

- edlstream.h

8.329 IRAStream Class Reference

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

```cpp
#include <edlstream.h>
```
Inheritance diagram for IRAStream:

```
Inheritance diagram for IRAStream:

IRAStream

IRAInputStream
IRAOutputStream
IRAInputPushbackStream

Public Member Functions

• virtual int64 length ()=0
  Get length of the stream.
• virtual bool setPos (int64 newPos)=0
  Set stream position.

8.329.1 Detailed Description

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

8.329.2 Member Function Documentation

8.329.2.1 length()

virtual int64 IRAStream::length ( ) [pure virtual]
Get length of the stream.
Returns

  int64 The length of the stream

8.329.2.2 setPos()

virtual bool IRAStream::setPos (int64 newPos) [pure virtual]
Set stream position.
Parameters

| newPos | The desired new position |

Returns

```
bool True if method succeeded
```

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

- edistream.h

8.330 IRCObject Class Reference

Base class Interface for all Reference Counted objects.

```
#include <ircobject.h>
```
Inheritance diagram for IRCObject:

- Public Member Functions
  - virtual void addRef ()=0
    
    Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
  - virtual bool decRef ()=0
    
    Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to.
Protected Member Functions

- virtual ~IRCObject ()
  
  Virtual destructor.

8.330.1 Detailed Description

Base class Interface for all Reference Counted objects.

8.330.2 Member Function Documentation

8.330.2.1 decRef()

virtual bool IRCObject::decRef () [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

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

- ircobject.h

8.331 JawsMako::IRedactionAnnotation Class Reference

A generic interface class for a redaction annotation.

#include <interactive.h>
Inheritance diagram for JawsMako::IRedactionAnnotation:

![Inheritance Diagram]

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.*

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

8.331.1 Detailed Description

A generic interface class for a redaction annotation.
8.331.2 Member Function Documentation

8.331.2.1 create()

```cpp
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>jawsMako</code></td>
<td>The JawsMako instance</td>
</tr>
<tr>
<td><code>rect</code></td>
<td>The annotations bounds. Must not be empty.</td>
</tr>
<tr>
<td><code>quadPoints</code></td>
<td>The quadrilaterals defining the redaction area. May be empty, in which case the annotation rect is used.</td>
</tr>
<tr>
<td><code>color</code></td>
<td>The outline color for the annotation when viewed in the user interface. Pass NULL for the default black</td>
</tr>
<tr>
<td><code>interiorColor</code></td>
<td>The color used to fill the annotation when it is applied. An empty colour will result in white being used. The color, if provided, must use the DeviceRGB color space.</td>
</tr>
</tbody>
</table>

Returns

`IRedactionAnnotationPtr` A smart pointer to the new redaction annotation

8.331.2.2 getInteriorColor()

```cpp
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

8.331.2.3 getQuadPoints()

```cpp
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
8.331.2.4 setInteriorColor()

virtual void JawsMako::IRedactionAnnotation::setInteriorColor (const IDOMColorPtr & color) [pure virtual]

Set the interior color to be used to fill the line endings.

Parameters

| color  | The desired color. Pass NULL to remove the color (default is white). If a color is passed, it must use the DeviceRGB color space. |

8.331.2.5 setQuadPoints()

virtual void JawsMako::IRedactionAnnotation::setQuadPoints (const CQuadPointVect & quadPoints) [pure virtual]

Set the redaction annotation's quad points.

Parameters

| quadPoints | The annotation's desired quad points |

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

- interactive.h

8.332 JawsMako::IRedactorTransform Class Reference

A transform for applying redaction redactions.

#include <transforms.h>
Inheritance diagram for JawsMako::IRedactorTransform:

```
IRObject
   
JawsMako::ITransform
   
JawsMako::IRedactorTransform
```

Public Member Functions

- virtual void `setTargetSpace` (const IDOMColorSpacePtr &targetSpace)=0
  
  *Set the colour space to use when rendering redacted sections. The default is DeviceRGB. The colour 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.*

Static Public Member Functions

- static JAWSMAKO_API IRedactorTransformPtr `create` (const IJawsMakoPtr &jawsMako)
  
  *Create the transform.*

Additional Inherited Members

8.332.1 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.
8.332.2 Member Function Documentation

8.332.2.1 create()

static JAWSMAKO_API IRedactorTransformPtr JawsMako::IRedactorTransform::create (const IJawsMakoPtr & jawsMako) [static]

Create the transform.

Parameters

- **JawsMako**: The JawsMako instance.

Returns

- The new instance.

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

- `transforms.h`

8.333 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.

```cpp
#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 colour 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 colour 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 colour space for rendered content, using a profile.

- 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 setGenerateMasks (bool generate)=0
  
  Set whether or not to generated 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.

- 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 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 IDOMMShadingPatternType3Brush::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.
Static Public Member Functions

- static JAWSMAKO_API IRendererTransformPtr create (const IJawsMakoPtr &jawsMako)
  Create the transform.
- static JAWSMAKO_API ClnkInfoVect findLinks (const IJawsMakoPtr &jawsMako, const IDOMNodePtr &nodeTree)
  Find all the inks used in the given DOM node tree. Utility. Also makes an attempt to determine the colour value of the colourant.

Additional Inherited Members

8.333.1 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.

8.333.2 Member Function Documentation

8.333.2.1 create()

static JAWSMAKO_API IRendererTransformPtr JawsMako::IRendererTransform::create (const IJawsMakoPtr & jawsMako) [static]

Create the transform.

Parameters

- **JawsMako** The JawsMako instance.

Returns

- The new instance.

8.333.2.2 setGenerateMasks()

virtual void JawsMako::IRendererTransform::setGenerateMasks (bool generate) [pure virtual]

Set whether or not to generated masked images as the rendererd result. If true, the individual images generated as part of rendering will have a bilevel alpha channel providing a mask to just the silhouette of the dom nodes targetted
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.

8.333.2.3 setMonochromeMode()

```cpp
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 colour 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 colour space is not DeviceGray or sGray, sGray will be set as the target colour 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 colour preservation will result in an exception being thrown.

8.333.2.4 setRenderTransparentNodesOnPageGroupMismatch()

```cpp
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 colour 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.
8.333.2.5  setSpotHalftone()

virtual void JawsMako::IRendererTransform::setSpotHalftone (  
    float frequency,  
    bool useFullResolutionForFlattening = false ) [pure virtual]

Set the simple spot haltone to be used when monochrome mode is enabled.

The spot halftone is always at 45 degrees. The default is 60.0f. The halftone is not used if monochrome mode is not enabled. The halftone frequency must be greater than 0. Normally, the resolution at which compositing for transparent images occurs is capped at twice the halftone frequency. For some content, this may not be appropriate. Set useFullResolutionForFlattening for such cases.

8.333.2.6  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

<table>
<thead>
<tr>
<th>width</th>
<th>The width of the halftone cell, in pixels</th>
</tr>
</thead>
<tbody>
<tr>
<td>height</td>
<td>The height of the halftone cell, in pixels</td>
</tr>
<tr>
<td>thresholdArray</td>
<td>The threshold array. Must be width * height in length, and no longer than 65535 entries.</td>
</tr>
</tbody>
</table>

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

- transforms.h

8.334  IRunnable Class Reference

Interface to filter's runnable classes.

#include <isession.h>
Inheritance diagram for IRunnable:

```
IRCOBJECT

IRunnable

IEDLOBJECT
```

Public Member Functions

- virtual RunCode run ()=0
  
  Run runnable object.

Additional Inherited Members

8.334.1 Detailed Description

Interface to filter's runnable classes.

8.334.2 Member Function Documentation

8.334.2.1 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

---

Generated by Doxygen
8.335 ISession Class Reference

EDL session class.

```cpp
#include <isession.h>
```

Inheritance diagram for ISession:

```
IRCOobject
   |
   V
IEDLObject
   |
   V
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 bool getTempStore (IEDLTempStorePtr &tempStore)=0
  Get the temporary store for this session. The temporary directory must be set before calling this member.
- virtual IEDLTempStorePtr getTempStore ()=0
  Convenience method to get the temporary store. 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.
Additional Inherited Members

8.335.1 Detailed Description

EDL session class.

8.335.2 Member Function Documentation

8.335.2.1 getFactory()

virtual IEDLClassFactory* ISession::getFactory () [pure virtual]

EDL Class Factory getter method.

Returns

IEDLClassFactory* Returns pointer to EDL class factory

8.335.2.2 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.

8.335.2.3 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.

8.335.2.4 getStartupDirectory()

virtual bool ISession::getStartupDirectory ( 
    EDLSysString & sStartupDirectory ) [pure virtual]

Get the startup directory.
Parameters

- **sStartupDirectory**: The path of the directory that the executable started.

Returns

- **bool**: Returns true on success.

### 8.3.3.5.2.5 getTemporaryDirectory()

```
virtual bool ISession::getTemporaryDirectory (
    EDLSysString & sTempDirectory ) [pure virtual]
```

Get the temporary directory to be used by EDL and filters for this session.

Parameters

- **sTempDirectory**: A reference to receive path to the temp directory.

Returns

- **bool**: Returns true on success.

### 8.3.3.5.2.6 getTempStore() [1/2]

```
virtual bool ISession::getTempStore ( 
    IEDLTempStorePtr & tempStore ) [pure virtual]
```

Get the temporary store for this session. The temporary directory must be set before calling this member.

Parameters

- **tempStore**: A reference to receive the temp store.

Returns

- **bool**: Returns true on success.

### 8.3.3.5.2.7 getTempStore() [2/2]

```
virtual IEDLTempStorePtr ISession::getTempStore ( ) [pure virtual]
```

Convenience method to get the temporary store. Throws an **IEDLError** on failure.
Returns

IEDLTempStorePtr The temp store.

8.335.2.8 setFactory()

virtual bool ISession::setFactory (  
        IEDLClassFactory * pFactory ) [pure virtual]

initializes Session by setting EDL class Factory

Parameters

| pFactory | Ptr to EDL Class Factory |

Returns

bool Returns true on success

8.335.2.9 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

| sStartupDirectory | The path of the directory that the executable started |

Returns

bool Returns true on success.

8.335.2.10 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sTempDirectory</td>
<td>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.</td>
</tr>
</tbody>
</table>

Returns

bool Returns true on success.

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

- isession.h

8.336 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:

```
IRObject

JawsMako::IAnnotation

JawsMako::IMarkupAnnotation

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 line endings.
Additional Inherited Members

8.336.1 Detailed Description

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

8.336.2 Member Function Documentation

8.336.2.1 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

8.336.2.2 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

8.336.2.3 setInteriorColor()

virtual void JawsMako::IShapeAnnotation::setInteriorColor ( const IDOMColorPtr & color ) [pure virtual]

Set the interior color to be used to fill the line endings.

Parameters

color The color definition, or pass NULL to remove the color
The documentation for this class was generated from the following file:

- interactive.h

8.337 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.

Static Public Member Functions

- static JAWSMAKO_API ISkiaRendererPtr create (const IJawsMakoPtr &jawsMako)
  
  Create a Skia Renderer Instance.

Additional Inherited Members

8.337.1 Detailed Description

A renderer that can paint XPS compatible DOM into a Skia canvas using the Skia API.

8.337.2 Member Function Documentation
8.337.2.1 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.

8.337.2.2 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.

8.337.2.3 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

8.338 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:

```
IRLObject
  |
  V
JawsMako::IAnnotation
  |
  V
JawsMako::IMarkupAnnotation
  |
  V
JawsMako::ISoundAnnotation
```

Public Member Functions

- virtual IInputStreamPtr `getSoundAsWav()` const =0
  
  Get the embedded sound (if present) as a WAV stream.

Additional Inherited Members

8.338.1 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.

8.338.2 Member Function Documentation

8.338.2.1 `getSoundAsWav()`

```cpp
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`
A generic interface class for a stamp annotation.

#include <interactive.h>

Inheritance diagram for JawsMako::IStampAnnotation:

```
IRCOBJECT

JawsMako::IAnnotation

JawsMako::IMarkupAnnotation

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.

**Static Public Member Functions**

- static JAWSMAKO_API IStampAnnotationPtr create (const IJawsMakoPtr &jawsMako, const FRect &rect, const U8String &name=U8String("Draft"))
  
  Create a stamp annotation.
A generic interface class for a stamp annotation.

8.339.2.1 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>jawsMako</code></td>
<td>The JawsMako instance</td>
</tr>
<tr>
<td><code>rect</code></td>
<td>The annotations bounds. Must not be empty</td>
</tr>
<tr>
<td><code>name</code></td>
<td>The &quot;name&quot; of the stamp</td>
</tr>
</tbody>
</table>

### Returns

*IStampAnnotationPtr* A smart pointer to the new stamp annotation

8.339.2.2 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*
8.339.2.3 getName()

virtual U8String JawsMako::IStampAnnotation::getName () const [pure virtual]

Get the stamp annotation's name.

Returns

U8String The stamp annotation's name

8.339.2.4 setIntent()

virtual void JawsMako::IStampAnnotation::setIntent ( const U8String & intent ) [pure virtual]

Set the stamp annotation's intent. Note: This is a PDF 2.0 feature. If intent is set to anything other than Stamp (the default), then the name of the stamp annotation will be dropped.

Parameters

| intent | The intent |

8.339.2.5 setName()

virtual void JawsMako::IStampAnnotation::setName ( const U8String & name ) [pure virtual]

Set the stamp annotation's name.

Parameters

| name   | The desired name |

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

- interactive.h

8.340 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:

[Diagram showing inheritance relationships]

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 (IDOMPathNode::eStrokeMiterLimitTreatment treatment, bool convert=true)=0
  
  Sets whether or not the stroker should convert paths with the given miter treatment mode. Note that if this is set the stroker will not convert paths where the miters would not be seen, such as if none of the corners in the path are acute enough to generate miters longer than the threshold of the path. The default is false in all cases, but this is ignored if setConvertAllStrokes() has been set to true.

- virtual void setMiterErrorThreshold (float maxError)=0
  
  Sets the approximate allowable amount of error, in pixels at the target resolution allowed for miter joins. This does not affect the result of converting a stroke to a filled path, but does affect whether or not to convert if setConvertForMiterTreatment() has been set to true for the applicable miter treatment method. It is designed to allow thin paths where miters would not be easily discernable to be left as is. The default is 0.0.

- virtual void setConvertForLineCap (IDOMPathNode::eStrokeLineCap cap, bool convert=true)=0
  
  Sets whether or not strokes containing the given line cap type should be converted. In all cases the default is false, but this is ignored if setConvertAllStrokes() has been set to true.

- virtual void setConvertForDashCap (IDOMPathNode::eStrokeLineCap cap, bool convert=true)=0
Sets whether or not strokes containing the given dash cap type should be converted. In all cases the default is false, but this is ignored if setConvertAllStrokes() has been set to true.

- virtual void 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 caps. The default is false but this is 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.

Static Public Member Functions

- static JAWSMAKO_API IStrokerTransformPtr create (const IJawsMakoPtr &jawsMako)

Create the transform.

Additional Inherited Members

8.340.1 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.

8.340.2 Member Function Documentation

8.340.2.1 create()

static JAWSMAKO_API IStrokerTransformPtr JawsMako::IStrokerTransform::create ( const IJawsMakoPtr & jawsMako ) [static]

Create the transform.
Parameters

- **JawsMako**: The JawsMako instance.

Returns

The new instance.

### 8.340.2.2 setupForPDFStyleOutput()

```cpp
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);`

### 8.340.2.3 setupForXPSStyleOutput()

```cpp
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`

### 8.341 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.

Static Public Member Functions

- static JAWSMAKO_API IStructurePtr create (const IJawsMakoPtr &jawsMako)
  Create a new Structure.

Additional Inherited Members

8.341.1 Detailed Description

Top level tracking structure describing the logical structure of the document.

8.341.2 Member Function Documentation

8.341.2.1 tagNode()

virtual IDOMNodePtr JawsMako::IStructure::tagNode (=
    const IStructureElementPtr & element,
    const IPagePtr & page,
    const IDOMNodePtr & nodeToTag ) [pure virtual]

Tag an IDOMNode from the given page with the given element.

If the node is an IDOMGroup (which isn’t already optional content or tagged with structure information) or IDOMForm, the node will have it's marked content information set accordingly and the returned result will be that node. However, if the node is some other graphical node, it will be moved inside an IDOMGroup with the optional content information set. If the node is in a tree, the node will be replaced with the group. The group will be returned.

The given element must be present in the structure.

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

- structure.h
A structure element in the structure tree.

```cpp
#include <structure.h>
```

Inheritance diagram for JawsMako::IStructureElement:

```
+-------------------+        +-------------------+
| IRCObject         |        | 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 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 into 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.
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, IMarkedContentStructureDetailsPtr &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.

Static Public Member Functions

- static JAWSMAKO_API IStructureElementPtr create (const IJawsMakoPtr &jawsMako, const U8String &structureType)
  
  Create a structure element.

Additional Inherited Members

8.342.1 Detailed Description

A structure element in the structure tree.

8.342.2 Member Function Documentation
8.342.2.1 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.

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

- structure.h

8.343 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.
Additional Inherited Members

8.343.1 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

8.344 JawsMako::IStructureElementReference Class Reference

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

#include <structure.h>

Inheritance diagram for JawsMako::IStructureElementReference:

```
+----------------------------------------+
<table>
<thead>
<tr>
<th>IRCObject</th>
</tr>
</thead>
<tbody>
<tr>
<td>+----------------------------------------</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>+----------------------------------------</td>
</tr>
</tbody>
</table>
```

Public Member Functions

- virtual bool equals (const IStructureElementReferencePtr &other) const =0
  
  Determine if another structure element reference refers to the same element.

Additional Inherited Members

8.344.1 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
A child of a structure element that points to another structure element.

#include <structure.h>

Inheritance diagram for JawsMako::IStructureElementReferenceChild:

```
IROCObject

|<------------------------|
|-- JawsMako::IStructureElementChild |
|<------------------------|

|<------------------------|
|-- JawsMako::IStructureElementReferenceChild |
|<------------------------|
```

Public Member Functions

- virtual IStructureElementPtr getElement () const =0
  
  Obtain a reference to this structure element.

Additional Inherited Members

8.345.1 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
A child of a structure element that points to a piece of marked content. Note; to create these, please see `StructureElement::createMarkedContentReferencePair()`.

#include <structure.h>

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.

Additional Inherited Members

8.346.1 Detailed Description

A child of a structure element that points to a piece of marked content. Note; to create these, please see `StructureElement::createMarkedContentReferencePair()`.

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

- structure.h
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:

![Inheritance Diagram]

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.

Additional Inherited Members

8.347.1 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
A SVG generator for JawsMako, allowing simple generation of SGG fragments for individual DOM nodes or entire pages.

```cpp
#include <svggenerator.h>
```

Inheritance diagram for JawsMako::ISVGGenerator:

```
IRObject
```

### Public Member Functions

- `virtual IRAInputStreamPtr generateSVG (const IDOMNodePtr &node)=0`  
  Generate SVG for the given DOM Node, returning the result in a stream.
- `virtual void generateSVG (const IDOMNodePtr &node, const IOutputStreamPtr &outputStream)=0`  
  Alternate form of `generateSVG()` when an existing stream should be used.
- `virtual IInputStreamPtr getResource (const U8String &name)=0`  
  Get the stream for a named resource. An exception will be thrown if the resource cannot be found.
- `virtual void getResources (CSVGResourceVect &resources)=0`  
  Get all the resources in a vector.
- `virtual void 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 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.

### Static Public Member Functions

- `static JAWSMAKO_API ISVGGeneratorPtr create (const IJawsMakoPtr &jawsMako, const U8String &resourcePrefix=U8String())`  
  Create a SVG generator instance.
Additional Inherited Members

8.348.1 Detailed Description

A SVG generator for JawsMako, allowing simple generation of SGG 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.

8.348.2 Member Function Documentation

8.348.2.1 create()

```cpp
static JAWSMAKO_API ISVGGeneratorPtr JawsMako::ISVGGenerator::create (
    const IJawsMakoPtr & jawsMako,
    const U8String & resourcePrefix = U8String() ) [static]
```

Create a SVG generator instance.

**Parameters**

<table>
<thead>
<tr>
<th>JawsMako</th>
<th>The JawsMako instance.</th>
</tr>
</thead>
<tbody>
<tr>
<td>resourcePrefix</td>
<td>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.</td>
</tr>
</tbody>
</table>

**Returns**

`ISVGGeneratorPtr` The new instance.

8.348.2.2 generateSVG() [1/2]

```cpp
virtual IRAInputStreamPtr JawsMako::ISVGGenerator::generateSVG (
    const IDOMNodePtr & node ) [pure virtual]
```

Generate SVG for the given DOM Node, returning the result in a stream.

**Parameters**

| node | The DOM node to be represented as SVG. |

Returns

IRAInputStreamPtr The SVG stream.

### 8.348.2.3 generateSVG() [2/2]

```cpp
def generateSVG(self, node, outputStream):
    # Alternate form of generateSVG() when an existing stream should be used.
```

Parameters

| **node** | The DOM node to be represented as SVG. |
| **IOutputStreamPtr** | The destination stream for the SVG. |

### 8.348.2.4 getResource()

```cpp
def getResource(self, name):
    # Get the stream for a named resource. An exception will be thrown if the resource cannot be found.
```

Parameters

| **name** | The name of the resource. |

Returns

IInputStreamPtr The resource stream.

### 8.348.2.5 getResources()

```cpp
def getResources(self, resources):
    # Get all the resources in a vector.
```

Parameters

| **resources** | A reference to a vector to receive the entries. |
8.348.2.6 setEnableImageDownsampling()

virtual void JawsMako::ISVGGenerator::setEnableImageDownsampling (  
    bool downsample ) [pure virtual]

Enable or disable image downsampling.

If true, images whose resolution is 50% higher than the target resolution (see setResolution()) will be downsampled
  to the target resolution.

If false, images will not be downsampled.

The default is true.

8.348.2.7 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.

8.348.2.8 setTargetResolutionCallback()

virtual void JawsMako::ISVGGenerator::setTargetResolutionCallback (  
    void * priv,  
    UrlCallbackForTarget callback ) [pure virtual]

Set a callback to provide the Url (absolute or relative) of the object where a target may be found. Return either the
  URL of the containing object, or an empty string if no such target exists.

The priv argument will be provided to the callback at each invocation.

See svggenerator.cpp for an example of its use.

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

- svggenerator.h
A generic interface class for a text (sticky note) annotation.

#include <interactive.h>

Inheritance diagram for JawsMako::ITextAnnotation:

```
IRObject
```

```
JawsMako::IAnnotation
```

```
JawsMako::IMarkupAnnotation
```

```
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.*

### 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.*
**8.349.1 Detailed Description**

A generic interface class for a text (sticky note) annotation.

**8.349.2 Member Function Documentation**

**8.349.2.1 create()**

```cpp
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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>jawsMako</code></td>
<td>The JawsMako instance</td>
</tr>
<tr>
<td><code>rect</code></td>
<td>The annotations bounds. Must not be empty.</td>
</tr>
<tr>
<td><code>contents</code></td>
<td>The textual contents of the text annotation.</td>
</tr>
<tr>
<td><code>open</code></td>
<td>Optional; whether or not the note is open. Default is false.</td>
</tr>
<tr>
<td><code>iconName</code></td>
<td>Optional; the name of the icon to use for displaying the annotation. The default is &quot;Note&quot;. Must not be empty.</td>
</tr>
<tr>
<td><code>popup</code></td>
<td>The associated popup. The associated popup must be added to the annotations of the same page.</td>
</tr>
</tbody>
</table>

**Returns**

`ITextAnnotationPtr` A smart pointer to the new text annotation

**8.349.2.2 getIconName()**

```cpp
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
8.349.2.3 getOpen()

virtual bool JawsMako::ITextAnnotation::getOpen() const [pure virtual]

Get the annotation’s open status.

Returns

bool True if open, false if closed

8.349.2.4 setIconName()

virtual void JawsMako::ITextAnnotation::setIconName(const U8String & iconName) [pure virtual]

Set the annotation’s icon name.

Parameters

| iconName | the annotation’s icon name. Cannot be empty |

8.349.2.5 setOpen()

virtual void JawsMako::ITextAnnotation::setOpen(bool open) [pure virtual]

Set the annotation’s open status.

Parameters

| open | Set to true for open, false for closed |

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

- interactive.h

8.350 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:

```
!#$$%-..#))/#.
%01234'5)

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.

Additional Inherited Members

8.350.1  Detailed Description

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

8.350.2  Member Function Documentation

8.350.2.1  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
8.350.2.2 setQuadPoints()

virtual void JawsMako::ITextMarkupAnnotation::setQuadPoints (const CQuadPointVect & quadPoints) [pure virtual]

Set the markup annotation's quad points.

Parameters

| quadPoints | The markup annotation's quad points |

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

- interactive.h

8.351 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:

![Inheritance diagram](image)

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

Generated by Doxygen
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 get_cornersOnPage (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.

Additional Inherited Members

8.351.1 Detailed Description

A run of text, containing unicode information, the position, transformation and bounds of the text.

8.351.2 Member Function Documentation

8.351.2.1 getBoundsOnPage()

FRect JawsMako::ITextRun::getBoundsOnPage ( bool tight = true ) const [inline]

Get the bounds of the run, in page coordinates. Convenience.

Parameters

| tight | Whether the bounding box should be 'tight' around the actual glyph markup, or expanded to include the full em height. |

8.351.2.2 getLocalBounds()

virtual FRect JawsMako::ITextRun::getLocalBounds (
bool tight = true) const  [pure virtual]

Get the bounds of the run, in glyph-local coordinates.

Parameters

| tight | Whether the bounding box should be 'tight' around the actual glyph markup, or expanded to include the full em height. |

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

- text.h

8.352 JawsMako::ITextSearch Class Reference

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

#include <text.h>

Inheritance diagram for JawsMako::ITextSearch:

![Inheritance Diagram](image)

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'.

Static Public Member Functions

- static JAWSMAKO_API ITextSearchPtr create (IEDLClassFactory *factory, const IPageLayoutPtr &page←Layout)

  Creation function for ITextSearch that performs text searching using page information obtained from an IPageLayout. Throws an IEDLError on failure.
Additional Inherited Members

8.352.1 Detailed Description

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

8.352.2 Member Function Documentation

8.352.2.1 create()

```cpp
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

<table>
<thead>
<tr>
<th>pFactory</th>
<th>The factory to use</th>
</tr>
</thead>
<tbody>
<tr>
<td>pageLayout</td>
<td>The page layout to use</td>
</tr>
</tbody>
</table>

Returns

ITextSearchPtr The new text searcher

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

- text.h

8.353 JawsMako::ITextSelect Class Reference

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

```cpp
#include <text.h>
```
Inheritance diagram for JawsMako::ITextSelect:

```
IRCOBJECT

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.

**Static Public Member Functions**

- static `JAWSMAKO_API ITextSelectPtr 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.

**Additional Inherited Members**

8.353.1 Detailed Description

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

8.353.2 Member Function Documentation

8.353.2.1 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

<table>
<thead>
<tr>
<th>pFactory</th>
<th>The factory to use</th>
</tr>
</thead>
<tbody>
<tr>
<td>pageLayout</td>
<td>The page layout to use</td>
</tr>
</tbody>
</table>

Returns

ITextSelectPtr The new text selector

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

- text.h

8.354 JawsMako::IThreads Class Reference

An interface class for document threads. Currently a stub interface.

#include <interactive.h>

Inheritance diagram for JawsMako::IThreads:

```
IRCOBJECT
<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
```

JawsMako::IThreads

Additional Inherited Members

8.354.1 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
ITransforms provide a method of applying common operations on DOM objects such as brushes, nodes, colours, colouverspaces 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 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.

### Additional Inherited Members

#### 8.355.1 Detailed Description

ITransforms provide a method of applying common operations on DOM objects such as brushes, nodes, colours, colourspaces 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.

#### 8.355.2 Member Function Documentation

##### 8.355.2.1 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>brush</code></td>
<td>The brush to be processed.</td>
</tr>
<tr>
<td><code>brushUsage</code></td>
<td>What the brush is to be used for.</td>
</tr>
<tr>
<td><code>state</code></td>
<td>The active <code>CTransformState</code> when this brush is encountered. Not all transforms require this information.</td>
</tr>
</tbody>
</table>

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.

8.355.2.2 transform() [2/5]

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>image</code></td>
<td>The image to be processed.</td>
</tr>
<tr>
<td><code>state</code></td>
<td>The active <code>CTransformState</code> when this image is encountered. Not all transforms require this information.</td>
</tr>
</tbody>
</table>

Returns

IDOMImagePtr 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.

8.355.2.3 transform() [3/5]

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>color</code></td>
<td>The color to be processed.</td>
</tr>
<tr>
<td><code>state</code></td>
<td>The active <code>CTransformState</code> when this color is encountered. Not all transforms require this information.</td>
</tr>
</tbody>
</table>
Returns

IDOMColorPtr The result. If modifications have been applied the result will be a new color.

8.355.2.4 transform() [4/5]

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

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>colorSpace</td>
<td>The colorSpace to be processed.</td>
</tr>
<tr>
<td>state</td>
<td>The active CTransformState when this color space is encountered. Not all transforms require this information.</td>
</tr>
</tbody>
</table>

Returns

IDOMColorSpacePtr The result. If modifications have been applied the result will be a new color space.

8.355.2.5 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

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>node</td>
<td>The node to be processed.</td>
</tr>
<tr>
<td>changed</td>
<td>Will be set to true on return if any changes have been made.</td>
</tr>
<tr>
<td>transformChildren</td>
<td>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.</td>
</tr>
<tr>
<td>state</td>
<td>The active CTransformState when this node is encountered. Not all transforms require this information.</td>
</tr>
</tbody>
</table>

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()

```cpp
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>page</code></td>
<td>The page to be processed.</td>
</tr>
<tr>
<td><code>transformContent</code></td>
<td>If true, apply the transform to the page content</td>
</tr>
<tr>
<td><code>transformAnnotations</code></td>
<td>If true, apply the transform to the annotations</td>
</tr>
</tbody>
</table>

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

- `transforms.h`

---

### 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.

```cpp
#include <transforms.h>
```

Inheritance diagram for JawsMako::ITransformChain:

```
IRCOBJ
```

```
JawsMako::ITransformChain
```

---

Generated by Doxygen
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 transformAnnotations=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.

Static Public Member Functions

- static JAWSMAKO_API ITransformChainPtr create (const IJawsMakoPtr &jawsMako, const CTransformVect &transforms=CTransformVect())  
  Create a new transform chain.

Additional Inherited Members

8.356.1 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 only once.

8.356.2 Member Function Documentation

8.356.2.1 create()

static JAWSMAKO_API ITransformChainPtr JawsMako::ITransformChain::create (  
  const IJawsMakoPtr & jawsMako,  
  const CTransformVect & transforms = CTransformVect() ) [static]

Create a new transform chain.
Parameters

<table>
<thead>
<tr>
<th>JawsMako</th>
<th>The JawsMako instance.</th>
</tr>
</thead>
<tbody>
<tr>
<td>transforms</td>
<td>The initial vector of ITransforms for the chain.</td>
</tr>
</tbody>
</table>

Returns

ITransformChainPtr The new instance.

8.356.2.2 getTransforms()

virtual CTransformVect JawsMako::ITransformChain::getTransforms ( ) const [pure virtual]

Get the transforms in a vector.

Returns

CTransformVect The transforms.

8.356.2.3 pushTransform()

virtual void JawsMako::ITransformChain::pushTransform ( |
const ITransformPtr & transform ) [pure virtual]

Push an ITransform onto the end of the ITransformChain. Should not be called if another thread is currently using the transform.

Parameters

| transform | The transform to push. |

8.356.2.4 pushTransformFront()

virtual void JawsMako::ITransformChain::pushTransformFront ( |
const ITransformPtr & transform ) [pure virtual]

Push an ITransform onto the front of the ITransformChain. Should not be called if another thread is currently using the transform.

Parameters

| transform | The transform to push. |
8.356.2.5 removeTransform()

virtual void JawsMako::ITransformChain::removeTransform ( uint32 index ) [pure virtual]

Remove the ITransform at the specified index from the ITransformChain.

Parameters

The index of the ITransform to remove

8.356.2.6 transform() [1/2]

virtual IDOMNodePtr JawsMako::ITransformChain::transform ( const IDOMNodePtr & node ) [pure virtual]

Apply the transform chain to the given node and its children, returning the resulting node. This is thread safe.

Parameters

node The node to transform.

Returns

IDOMNodePtr the modified node.

8.356.2.7 transform() [2/2]

virtual IDOMNodePtr JawsMako::ITransformChain::transform ( const IDOMNodePtr & node, bool & changed ) [pure virtual]

Apply the transform chain to the given node and its children, returning the resulting node, and providing an indication if any changes were actually made.

Parameters

node The node to transform.

changed On exit changes will be set to true if any modifications were made, false otherwise.
8.357 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:

```
IRCOBJECT

JawsMako::ITransform

JawsMako::IType3UnpackerTransform
```
Static Public Member Functions

- static JAWSMAKO_API IType3UnpackerTransformPtr create (const IJawsMakoPtr &jawsMako)
  Create the transform.

Additional Inherited Members

8.357.1 Detailed Description

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

Glyphs are recoloured if a plain solid colour 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.

8.357.2 Member Function Documentation

8.357.2.1 create()

static JAWSMAKO_API IType3UnpackerTransformPtr JawsMako::IType3UnpackerTransform::create {
  const IJawsMakoPtr & jawsMako } [static]

Create the transform.

Parameters

- **JawsMako**: The JawsMako instance.

Returns

- The new instance.

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

- transforms.h

8.358 JawsMako::IUnicodeHelper Class Reference

An interface into language specific unicode helpers.

#include <text.h>
Inheritance diagram for JawsMako::IUnicodeHelper:

![Inheritance Diagram](image)

### Public Member Functions

- **virtual bool isCombiningCharacter (const wchar_t wchar) const =0**
  
  Returns true if 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 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.

### Static Public Member Functions

- **static bool isSpaceCharacter (wchar_t ch)**

  Returns true if ch is a unicode space character.

### Additional Inherited Members

#### 8.358.1 Detailed Description

An interface into language specific unicode helpers.

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

- **text.h**
8.359   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:

```
  IRCObject
     |
     v
JawsMako::IAnnotation
     |
     v
JawsMako::IWidgetAnnotation
```

Public Member Functions

- 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 bool getFieldFlags (uint32 &flags) const =0
  Get the field flags. Please see the PDF specification for the definition of the flags. Note that the significance of the flags changes depending on type, and that the value may be inheritable.

Additional Inherited Members

8.359.1 Detailed Description

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

8.359.2 Member Function Documentation
8.359.2.1 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. Note that the significance of the flags changes depending on type, and that the value may be inheritable.
Parameters

flags | Reference to receive the field flags

Returns

bool False if the flags have not been set for this widget, otherwise true

8.359.2.2 getFieldType()

virtual eFieldType JawsMako::IWidgetAnnotation::getFieldType ( ) const [pure virtual]

Get the field type.

Returns

eFieldType The field type

8.359.2.3 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

8.359.2.4 setPartialName()

virtual void JawsMako::IWidgetAnnotation::setPartialName ( const U8String & name ) [pure virtual]

Set the partial name of the widget.

Parameters

name | The partial name to be set

The documentation for this class was generated from the following file:
8.360 JawsMako::IXAMLGenerator Class Reference

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

#include <xamlgenerator.h>

Inheritance diagram for JawsMako::IXAMLGenerator:

![Inheritance Diagram](image)

Classes

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

Public Member Functions

- virtual IRAInputStreamPtr generateXAML (const IDOMNodePtr &node, const IDOMCatalogPtr &catalog=I
  
  Generate XAML for the given DOM Node, returning the result in a stream.

- virtual void generateXAML (const IDOMNodePtr &node, const IOutputStreamPtr &outputStream, const ID
  
  Alternate form of generateXAML() when an existing stream should be used.

- virtual IRAInputStreamPtr generateXAMLForPageAndAnnotationAppearances (const IPagePtr &page, C
  
  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 generateXAMLForPageAndAnnotationAppearances().

- 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 generateXAMLForPageAndAnnotationAppearances() which is in turn slower than using generateXAML().

Generated by Doxygen
• 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, CAnnotationXAMLVector&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::eDownsamplingMethod method=IDOMImageDownsamplerFilter::eBicubic)=0
  Set the desired maximum resolution, threshold and downsampling method for colour images for XAML output.

• virtual void setGrayImageMaxResolution (float resolution, float threshold=0.0f, IDOMImageDownsamplerFilter::eDownsamplingMethod 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::eDownsamplingMethod 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 XPSX in the desired form.
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.

### Static Public Member Functions

- static JAWSMAKO_API IXAMLGeneratorPtr create (const IJawsMakoPtr &jawsMako, const U8String &resourcePrefix=U8String())
  Create a XAML generator instance.

### Additional Inherited Members

8.360.1 Detailed Description

A XAML generator for JawsMako, 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.

8.360.2 Member Function Documentation

8.360.2.1 create()

```cpp
static JAWSMAKO_API IXAMLGeneratorPtr JawsMako::IXAMLGenerator::create (
    const IJawsMakoPtr & jawsMako,
    const U8String & resourcePrefix = U8String() ) [static]
```

Create a XAML generator instance.

### Parameters

<table>
<thead>
<tr>
<th>JawsMako</th>
<th>The JawsMako instance.</th>
</tr>
</thead>
<tbody>
<tr>
<td>resourcePrefix</td>
<td>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.</td>
</tr>
</tbody>
</table>


Returns

IXAMLGeneratorPtr The new instance.

8.360.2.2 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

<table>
<thead>
<tr>
<th>node</th>
<th>The DOM node to be represented as XAML.</th>
</tr>
</thead>
<tbody>
<tr>
<td>catalog</td>
<td>The catalog associated with the node. Optional, but required if the names of named elements are to be present in the generated XAML.</td>
</tr>
</tbody>
</table>

Returns

IRAInputStreamPtr The XAML stream.

8.360.2.3 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

<table>
<thead>
<tr>
<th>node</th>
<th>The DOM node to be represented as XAML.</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOutputStreamPtr</td>
<td>The destination stream for the XAML.</td>
</tr>
<tr>
<td>catalog</td>
<td>The catalog associated with the node. Optional, but required if the names of named elements are to be present in the generated XAML.</td>
</tr>
</tbody>
</table>

8.360.2.4 generateXAMLForAppearance() [1/2]

virtual IRAInputStreamPtr JawsMako::IXAMLGenerator::generateXAMLForAppearance (  
const IAnnotationAppearancePtr & appearance,  

Generated by Doxygen
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**

| appearance | The appearance to be presented as XAML. |
| annotation | The annotation that contains the appearance. Required in order to position the annotation against the page background correctly. |
| page | The page the appearance appears on. |

### 8.360.2.5 generateXAMLForAppearance()

```cpp
def generateXAMLForAppearance(self, appearance: IAnnotationAppearancePtr, annotation: IAnnotationPtr, page: IPagePtr) -> None:
    # Implementation...
```

Alternate form of `generateXAMLForAppearance()` where an existing stream should be used. Provide the annotation and page from which the appearance comes. The page is required as some annotations may need to be composited against the page backdrop. This is slower than invoking `generateXAMLForPageAnnotationAppearances()` which is in turn slower than using `generateXAMLForPageAndAnnotations()`.

**Parameters**

| appearance | The appearance to be presented as XAML. |
| annotation | The annotation that contains the appearance. Required in order to position the annotation against the page background correctly. |
| page | The page the appearance appears on. |

### 8.360.2.6 generateXAMLForPageAndAnnotationAppearances()

```cpp
def generateXAMLForPageAndAnnotationAppearances(self, page: IPagePtr, appearanceXAMLs: CAnnotationXAMLVect) -> IRAInputStreamPtr:
    # Implementation...
```

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

<table>
<thead>
<tr>
<th>page</th>
<th>The DOM node to be produced as XAML</th>
</tr>
</thead>
<tbody>
<tr>
<td>appearanceXAMLs</td>
<td>A vector to receive the XAML for the annotation appearances.</td>
</tr>
</tbody>
</table>

Returns

IRAInputStreamPtr The XAML stream.

8.360.2.7 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

<table>
<thead>
<tr>
<th>page</th>
<th>The page whose annotations should be represented as XAML.</th>
</tr>
</thead>
<tbody>
<tr>
<td>appearanceXAMLs</td>
<td>A vector to receive the XAML for the annotation appearances.</td>
</tr>
</tbody>
</table>

8.360.2.8 getResource()

virtual IInputStreamPtr JawsMako::IXAMLGenerator::getResource (const U8String & name ) [pure virtual]

Get the stream for a named resource. An exception will be thrown if the resource cannot be found.

Parameters

| name              | The name of the resource. |

Returns

IInputStreamPtr The resource stream.

8.360.2.9 getResources()

virtual void JawsMako::IXAMLGenerator::getResources (CEDLVector<CResourceEntry> & resources ) [pure virtual]
Get all the resources in a vector.

**Parameters**

| resources | A reference to a vector to receive the entries. |

### 8.360.2.10 setColorImageMaxResolution()

```cpp
template<typename T> virtual void JawsMako::IXAMLGenerator::setColorImageMaxResolution (const T& colors, float resolution, float threshold = 0.0f, IDOMImageDownsamplerFilter::eDownsamplingMethod method = IDOMImageDownsamplerFilter::eBicubic) = default
```

Set the desired maximum resolution, threshold and downsampling method for colour images for XAML output.

The default behaviour is leave the image resolution unchanged.

**Parameters**

| resolution | The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default. |
| threshold  | The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution. |
| method     | The method to use when downsampling. The default is bicubic for colour images. |

### 8.360.2.11 setGrayImageMaxResolution()

```cpp
virtual void JawsMako::IXAMLGenerator::setGrayImageMaxResolution (float resolution, float threshold = 0.0f, IDOMImageDownsamplerFilter::eDownsamplingMethod method = IDOMImageDownsamplerFilter::eBicubic) = default
```

Set the desired maximum resolution, threshold and downsampling method for gray images.

The default behaviour is leave the image resolution unchanged.

**Parameters**

| resolution | The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default. |
| threshold  | The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution. |
| method     | The method to use when downsampling. The default is bicubic for gray images. |
8.360.2.12 setJPEGQuality()

virtual void JawsMako::IXAMLGenerator::setJPEGQuality ( uint8 quality ) [pure virtual]

Set the JPEG quality to use when encoding images in JPEG format. Equivalent to calling setParameter() with the parameter name "JPEGQuality" and the value being the required quality.

Parameters

| quality | The desired quality level, with 1 being lowest quality and 5 being highest quality. |

8.360.2.13 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.

8.360.2.14 setMonoImageMaxResolution()

virtual void JawsMako::IXAMLGenerator::setMonoImageMaxResolution ( 
    float resolution, 
    float threshold = 0.0f, 
    IDOMImageDownsamplerFilter::eDownsamplingMethod method = IDOMImageDownsamplerFilter::eSubsample ) [pure virtual]

Set the desired maximum resolution, threshold and downsampling method for monochrome images.

The default behaviour is leave the image resolution unchanged.

Parameters

| resolution | The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default. |
| threshold  | The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution. |
| method     | The method to use when downsampling. The default is subsample for monochrome images; using any other method will result in grayscale output. |
8.360.2.15 setPreferredColorImageFormat()

virtual void JawsMako::IXAMLGenerator::setPreferredColorImageFormat (  
    IImageEncoderTransform::eEncodeFormat format ) [pure virtual]

Set the desired image format for color images that need to be reencoded for XAML output. The default is eEFAuto.  
Note: this is advisory only and may not be honoured in all cases if the image cannot be represented in XAML in the  
desired form.

Parameters

| format | The desired format. |

8.360.2.16 setPreferredGrayImageFormat()

virtual void JawsMako::IXAMLGenerator::setPreferredGrayImageFormat (  
    IImageEncoderTransform::eEncodeFormat format ) [pure virtual]

Set the desired image format for gray images that need to be reencoded for XAML output. The default is eEFAuto.  
Note: this is advisory only and may not be honoured in all cases if the image cannot be represented in XPS in the  
desired form.

Parameters

| format | The desired format. |

8.360.2.17 setPreferredMonoImageFormat()

virtual void JawsMako::IXAMLGenerator::setPreferredMonoImageFormat (  
    IImageEncoderTransform::eEncodeFormat format ) [pure virtual]

Set the desired image format for monochrome images that need to be reencoded for XAML output. The default is  
eEFAuto.  Note: this is advisory only and may not be honoured in all cases if the image cannot be represented in  
XPS in the desired form.

Parameters

| format | The desired format. |

8.360.2.18 setRenderResolution()

virtual void JawsMako::IXAMLGenerator::setRenderResolution (  
    uint32 resolution ) [pure virtual]

Generated by Doxygen
Set the resolution to use if page content requires rendering in order to be output as XAML. The default is 150dpi. This is affected also by the maximum image resolution parameters.

**Parameters**

| resolution | The desired resolution in dpi. |

8.360.2.19 setSubsetFonts()

```cpp
template<
    bool subset
>
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.

8.360.2.20 setTargetColorSpace()

```cpp
template<
    const IDOMColorSpacePtr & targetSpace
>
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**

| colorSpace | The desired color space. Must be a simple or ICC space. |

8.360.2.21 setTargetProfile()

```cpp
template<
    const IDOMICCProfilePtr & profile
>
setTargetProfile (const IDOMICCProfilePtr & profile) [pure virtual]
```

Set the target color space for the output using an ICC profile. The default behaviour is to, where possible, leave the color space of objects unchanged.

**Parameters**

| profile | The desired profile. |
8.361 JawsMako::IXPSInput Class Reference

An instance of the JawsMako XPS input class.

#include <xpsinput.h>

Inheritance diagram for JawsMako::IXPSInput:

```
IROObject
  |
  |
  |
|  |
| ^
| JawsMako::IInput
  |
  |
  |
|  |
| ^
| JawsMako::IXPSInput
```

Public Member Functions

- virtual IDocumentAssemblyPtr openStreaming (const IInputStreamPtr &inputStream)=0
  
  Open a stream, returning the IDocumentAssembly representing the contents, in streaming mode.

Static Public Member Functions

- static JAWSMAKO_API IXPSInputPtr create (const IJawsMakoPtr &jawsMako)
  
  Create an input for reading source documents in XPS format.

Additional Inherited Members

8.361.1 Detailed Description

An instance of the JawsMako XPS input class.

8.361.2 Member Function Documentation

Generated by Doxygen
8.361.2.1 create()

\texttt{static JAWSMAKO\_API IXPSInputPtr JawsMako::IXPSInput::create ( const IJawsMakoPtr & jawsMako ) \ [static]}

Create an input for reading source documents in XPS format.

Returns

\texttt{IXPSInputPtr the xps input}

8.361.2.2 openStreaming()

\texttt{virtual IDocumentAssemblyPtr JawsMako::IXPSInput::openStreaming ( const IInputStreamPtr & inputStream ) \ [pure virtual]}

Open a stream, returning the \texttt{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 \texttt{IError} exception with error code \texttt{JM\_ERR\_PAGE\_NOT\_FOUND} or \texttt{JM\_ERR\_DOCUMENT\_NOT\_FOUND} will be thrown. These can be caught and appropriately handled.

Returns

\texttt{IDocumentAssemblyPtr the document assembly}

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

- xpsinput.h
8.362 JawsMako::IXPSOutput Class Reference

Interface for the XPS IOutput class.

#include <xpsoutput.h>

Inheritance diagram for JawsMako::IXPSOutput:

```
IRObject

JawsMako::IOutput

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 setColorImageMaxResolution (float resolution, float threshold=0.0f, IDOMImageDownsampler Filter::eDownsamplingMethod method=IDOMImageDownsamplerFilter::eBicubic)=0
  
  Set the desired maximum resolution, threshold and downsampling method for colour images.

- virtual void setGrayImageMaxResolution (float resolution, float threshold=0.0f, IDOMImageDownsampler Filter::eDownsamplingMethod 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, IDOMImageDownsampler Filter::eDownsamplingMethod method=IDOMImageDownsamplerFilter::eSubsample)=0
  
  Set the desired maximum resolution, threshold and downsampling method for monochrome images.

- virtual void setTargetColorSpace (const IDOMColorSpacePtr &targetSpace)=0
  
  Set the target color space for the output. The default behaviour is to, where possible, leave the color space of objects unchanged. Equivalent to calling setParameter with the param name "TargetColorSpace" with appropriate values (please refer to documentation).
• virtual void setTargetProfile (const IDOMICCProfilePtr &profile)=0
  
  Set the target color space for the output using an ICC profile. The default behaviour is to, where possible, leave the color space of objects unchanged. Equivalent to calling setParameter() with the param name “TargetProfile” with the value as the path to the profile.

• virtual void applyColorConverterTransform (const IColorConverterTransformPtr &transform)=0
  
  Apply the given color converter transform to the contents before writing to XPS. This supersedes the target color space parameters described above. This allows for more advanced configuration of the color spaces of the output.

• virtual void setRenderResolution (uint32 resolution)=0
  
  Set the resolution to use if page content requires rendering in order to be output as XPS. The default is 150dpi. This is affected also by the maximum image resolution parameters. Equivalent to calling setParameter() with param name “RenderResolution” with the value as the desired resolution as value.

• virtual void applyRendererTransform (const IRendererTransformPtr &transform)=0
  
  Apply the given renderer transform to the contents before writing to XPS. This supersedes the target color space parameters described above. This allows for more advanced configuration of rendering.

• virtual void setPreferredColorImageFormat (IImageEncoderTransform::eEncodeFormat format)=0
  
  Set the desired image format for color images that need to be reencoded for XPS output. The default is eEFAuto. Note: this is advisory only and may not be honoured in all cases if the image cannot be represented in XPS in the desired form. Equivalent to calling setParameter() with the parameter name “ColorImageFormat” with appropriate values (please refer to documentation).

• virtual void setPreferredGrayImageFormat (IImageEncoderTransform::eEncodeFormat format)=0
  
  Set the desired image format for gray images that need to be reencoded for XPS output. The default is eEFAuto. Note: this is advisory only and may not be honoured in all cases if the image cannot be represented in XPS in the desired form. Equivalent to calling setParameter() with the parameter name “GrayImageFormat” with appropriate values (please refer to documentation).

• virtual void setPreferredMonoImageFormat (IImageEncoderTransform::eEncodeFormat format)=0
  
  Set the desired image format for monochrome images that need to be reencoded for XPS output. The default is eEFAuto. Note: this is advisory only and may not be honoured in all cases if the image cannot be represented in XPS in the desired form. Equivalent to calling setParameter() with the parameter name “MonoImageFormat” with appropriate values (please refer to documentation).

• virtual void setJPEGQuality (uint8 quality)=0
  
  Set the JPEG quality to use when encoding images in JPEG format. Equivalent to calling setParameter() with the parameter name “JPEGQuality” and the value being the required quality.

• virtual void applyEncoderTransform (const IImageEncoderTransformPtr &transform)=0
  
  Apply the given image encoder transform to the contents before writing to XPS. This supersedes the image encoding parameters described above. This allows for more advanced configuration of image encoding.

Static Public Member Functions

• static JAWSMAKO_API IXPSOutputPtr create (const IJawsMakoPtr &jawsMako)
  
  Create an XPS Output instance.

Additional Inherited Members

8.362.1 Detailed Description

Interface for the XPS IOutput class.

8.362.2 Member Function Documentation
8.362 JawsMako::IXPSOutput Class Reference

8.362.2.1 setColorImageMaxResolution()

virtual void JawsMako::IXPSOutput::setColorImageMaxResolution(
  float resolution,
  float threshold = 0.0f,
  IDOMImageDownsamplerFilter::eDownsamplingMethod method = IDOMImageDownsamplerFilter::eBicubic ) [pure virtual]

Set the desired maximum resolution, threshold and downsampling method for colour images.

The default behaviour is leave the image resolution unchanged. Equivalent to calling setParameter() with the param names "ColorImageDownsamplingResolution", "ColorImageDownsamplingThreshold" and "ColorImageDownsamplingMethod" with the respective values.

Parameters

<table>
<thead>
<tr>
<th>parameter</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resolution</td>
<td>The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default.</td>
</tr>
<tr>
<td>threshold</td>
<td>The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution.</td>
</tr>
<tr>
<td>method</td>
<td>The method to use when downsampling. The default is bicubic for colour images.</td>
</tr>
</tbody>
</table>

8.362.2.2 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 "GrayImageDownsamplingMethod" with the respective values.

Parameters

<table>
<thead>
<tr>
<th>parameter</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resolution</td>
<td>The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default.</td>
</tr>
<tr>
<td>threshold</td>
<td>The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution.</td>
</tr>
<tr>
<td>method</td>
<td>The method to use when downsampling. The default is bicubic for gray images.</td>
</tr>
</tbody>
</table>

8.362.2.3 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.
8.362.2.4 setMergeFonts()

```cpp
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.

8.362.2.5 setMonoImageMaxResolution()

```cpp
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resolution</td>
<td>The desired output resolution, in dpi. Pass 0 to leave images unchanged, which is the default.</td>
</tr>
<tr>
<td>threshold</td>
<td>The threshold above which images will be reduced to the desired resolution. Pass 0 to use the resolution.</td>
</tr>
<tr>
<td>method</td>
<td>The method to use when downsampling. The default is subsample for monochrome images; using any other method will result in grayscale output.</td>
</tr>
</tbody>
</table>

8.362.2.6 setPreferredColorImageFormat()

```cpp
virtual void JawsMako::IXPSOutput::setPreferredColorImageFormat (IImageEncoderTransform::eEncodeFormat format) [pure virtual]
```
Set the desired image format for color images that need to be reencoded for XPS output. The default is eEFAuto. Note: this is advisory only and may not be honoured in all cases if the image cannot be represented in XPS in the desired form. Equivalent to calling setParameter() with the parameter name "ColorImageFormat" with appropriate values (please refer to documentation).

### Parameters

| format | The desired format. |

#### 8.362.2.7 setPreferredGrayImageFormat()

virtual void JawsMako::IXPSOutput::setPreferredGrayImageFormat (IImageEncoderTransform::eEncodeFormat format) [pure virtual]

Set the desired image format for gray images that need to be reencoded for XPS output. The default is eEFAuto. Note: this is advisory only and may not be honoured in all cases if the image cannot be represented in XPS in the desired form. Equivalent to calling setParameter() with the parameter name "GrayImageFormat" with appropriate values (please refer to documentation).

### Parameters

| format | The desired format. |

#### 8.362.2.8 setPreferredMonoImageFormat()

virtual void JawsMako::IXPSOutput::setPreferredMonoImageFormat (IImageEncoderTransform::eEncodeFormat format) [pure virtual]

Set the desired image format for monochrome images that need to be reencoded for XPS output. The default is eEFAuto. Note: this is advisory only and may not be honoured in all cases if the image cannot be represented in XPS in the desired form. Equivalent to calling setParameter() with the parameter name "MonoImageFormat" with appropriate values (please refer to documentation).

### Parameters

| format | The desired format. |

#### 8.362.2.9 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.

Generated by Doxygen
Parameters

| resolution | The desired resolution in dpi. |

8.362.2.10 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.

8.362.2.11 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

| colorSpace | The desired color space. Must be a simple or ICC space. |

8.362.2.12 setTargetProfile()

virtual void JawsMako::IXPSOutput::setTargetProfile (const IDOMICCProfilePtr & profile) [pure virtual]

Set the target color space for the output using an ICC profile. The default behaviour is to, where possible, leave the color space of objects unchanged. Equivalent to calling setParameter() with the param name "TargetProfile" with the value as the path to the profile.

Parameters

| profile | The desired profile. |

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

- xpsoutput.h

Generated by Doxygen
8.363  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:

![Inheritance Diagram]

Additional Inherited Members

8.363.1  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

8.364  PValue Class Reference

Stores a "property" value that is tagged with an enumeration value that indicates the underlying type.

#include <edlproperty.h>

8.364.1  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
Opentype table signatures.

#include <idomfont.h>

8.365.1 Detailed Description

Opentype table signatures.

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

- idomfont.h
Chapter 9

File Documentation

9.1 edlblackpointcompensation.h File Reference

Simple enum for black point compensation.

#include <edl/edlnamespaces.h>

Enumerations

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

  Black point compensation enumeration.

9.1.1 Detailed Description

Simple enum for black point compensation.

9.1.2 Enumeration Type Documentation

9.1.2.1 eBlackPointCompensation

enum eBlackPointCompensation

Black point compensation enumeration.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eBPCDefault</td>
<td>Default behavior.</td>
</tr>
<tr>
<td>eBPCOn</td>
<td>Use black point compensation if applicable during color conversion.</td>
</tr>
<tr>
<td>eBPCOff</td>
<td>Do not use black point compensation during color conversion.</td>
</tr>
</tbody>
</table>
9.2 edblend.h File Reference

An enum for transparency blend modes.

#include <edl/edlnamespaces.h>

9.2.1 Detailed Description

An enum for transparency blend modes.

9.3 edlerrors.cpp File Reference

Publishes a look-up table that maps integer error codes, as returned by many of the EDL APIs, into a corresponding (English) string. It is expected that alternate language translations are implemented by creating alternate versions of this file.

#include <edl/edlerrors.h>

Functions

- EDLString EDL::getEDLErrorString (uint32 errorCode)

  Implements a simple numeric-error-code-to-corresponding-language-specific-error-message-string look-up using the "errorList" array/table.

9.3.1 Detailed Description

Publishes a look-up table that maps integer error codes, as returned by many of the EDL APIs, into a corresponding (English) string. It is expected that alternate language translations are implemented by creating alternate versions of this file.

9.3.2 Function Documentation

9.3.2.1 getEDLErrorString()

EDLString EDL::getEDLErrorString ( uint32 errorCode )

Implements a simple numeric-error-code-to-corresponding-language-specific-error-message-string look-up using the "errorList" array/table.
### Parameters

| errorCode | The error code number |

### Returns

EDLString Error message text

### 9.4 edlerrors.h File Reference

EDL errors.

```c
#include <typeinfo>
#include <edl/edltypes.h>
#include <edl/edlstring.h>
```

### Classes

- class `IEDLError`
  
  *An abstract class for EDL exceptions.*

### Functions

- EDLString `getEDLErrorString (uint32 errorCode)`
  
  *Implements a simple numeric-error-code-to-corresponding-language-specific-error-message-string look-up using the “errorList” array/table.*

- EDL_API void `throwEDLError (uint32 errorcode)`
  
  *Utility - Throw an `IEDLError` exception with the given error code.*

### Variables

- const uint32 `EDL_ERR_PANIC = 1`
  
  *Unstable state - abort the application.*

- const uint32 `EDL_ERR_UNDEFINED = 2`
  
  *Undefined error.*

- const uint32 `EDL_ERR_LICENSE = 3`
  
  *Invalid serial number or licence provided.*

- const uint32 `EDL_ERR_FILENOTFOUND = 4`
  
  *File not found.*

- const uint32 `EDL_ERR_CORRUPT_PNG_IMAGE = 5`
  
  *Corrupt PNG image encountered.*

- const uint32 `EDL_ERR_OUTOFMEMORY = 6`
  
  *Out of memory.*

- const uint32 `EDL_ERR_READ = 7`
  
  *Read error.*

- const uint32 `EDL_ERR_LARGE_FILE = 8`
File too large.

- `const uint32 EDL_ERR_OPENFORWRITE = 9`
  Open file for write failed.
- `const uint32 EDL_ERR_WRITE = 10`
  Write failed.
- `const uint32 EDL_ERR_IOERROR = 11`
  General I/O error.
- `const uint32 EDL_ERR_COMPRESS = 12`
  Compression error.
- `const uint32 EDL_ERR_CONFIG = 13`
  Configuration error.
- `const uint32 EDL_ERR_RESTRICTED_FONT = 14`
  A restricted font has been encountered.
- `const uint32 EDL_ERR_ABORTED = 15`
  Job aborted by user.
- `const uint32 EDL_ERR_BAD_ARGUMENTS = 16`
  General error for bad arguments passed to an API function.
- `const uint32 EDL_ERR_VECTOR_ERROR = 17`
  An out-of-range value was accessed in a CEDLVector.
- `const uint32 EDL_ERR_BAD_FONT_MAP = 18`
  An error was found in a FontMap file.
- `const uint32 EDL_ERR_UNICODE = 19`
  An invalid Unicode string was encountered.
- `const uint32 EDL_ERR_PAGE_TOO_LARGE = 20`
  A page was too large to be used.
- `const uint32 JM_ERR_UNSUPPORTED = 100`
  An attempt was made to use an unsupported, unimplemented feature.
- `const uint32 JM_ERR_REVERT_FAILED = 101`
  An attempt was made to revert a page that could not be performed.
- `const uint32 JM_ERR_RANGE_ERROR = 103`
  An attempt was made to request or add an item from the API with an out-of-bounds index.
- `const uint32 JM_ERR_PAGE_NOT_FOUND = 104`
  The given page was not found in a document.
- `const uint32 JM_ERR_DOCUMENT_NOT_FOUND = 105`
  The given document was not found in an assembly.
- `const uint32 JM_DUPLICATE_FIELD_PARTIAL_NAME = 106`
  A field with a duplicate partial name was found.
- `const uint32 JM_ERR_FORM_FIELD_NOT_FOUND = 107`
  The given form field was not found.
- `const uint32 JM_ERR_DUPLICATE_WIDGET = 108`
  A widget with a duplicate reference was not found.
- `const uint32 JM_ERR_WIDGET_NOT_FOUND = 109`
  The given widget annotation or annotation reference could not be found.
- `const uint32 JM_ERR_BAD_CONFIGURATION = 110`
  General configuration error.
- `const uint32 JM_ERR_APPEARANCE_NOT_FOUND = 111`
  The required annotation appearance was not found.
- `const uint32 JM_ERR_ANNOTATION_NOT_FOUND = 112`
  The required annotation was not found.
- const uint32 JM_ERR_ASSEMBLY_WRITE_FORBIDDEN = 113
  The given assembly cannot be written due to security permissions.
- const uint32 JM_ERR_FONT_NOT_FOUND = 114
  The requested font was not found.
- const uint32 JM_ERR_TOO_MANY_PDFOUT_WRITERS = 115
  There are too many PDF Out writers currently in progress.
- const uint32 JM_ERR_INFORMATION_NOT_AVAILABLE = 116
  The requested information is not yet available.
- const uint32 JM_ERR_DIRECTORY_DOESNT_EXIST = 117
  A required directory does not exist on disk.
- const uint32 JM_ERR_RESOURCE_NOT_FOUND = 118
  The requested resource could not be found.
- const uint32 JM_ERR_BAD_UNICODE_CMAP = 119
  A bad ToUnicode CMap was encountered.
- const uint32 JM_ERR_OPTIONAL_CONTENT_GROUP_NOT_FOUND = 120
  The requested optional content group was not found.
- const uint32 JM_ERR_INVALID_OPTIONAL_CONTENT = 121
  Invalid optional content information encountered.
- const uint32 JM_ERR_PCLXL = 122
  PCL/XL Processing error.
- const uint32 JM_ERR_PCL = 123
  PCL5 Processing error.
- const uint32 JM_ERR_PJL = 124
  PJL Processing error.
- const uint32 JM_ERR_SYMBOLSET_NOT_FOUND = 125
  The requested PCL or PCL/XL symbol set was not found.
- const uint32 JM_ERR_PDF_OBJECT_NOT_FOUND = 126
  An expected PDF object was not found.
- const uint32 EDL_ERR_BAD_DIMENSIONS = 501
  Bad dimensions specified for an object.
- const uint32 EDL_ERR_NULL_NODE = 502
  An unexpected null DOM node pointer was encountered.
- const uint32 EDL_ERR_BAD_BRUSH = 503
  A bad brush was encountered.
- const uint32 EDL_ERR_DOM_CREATION_FAILED = 504
  Failure while attempting to create a DOM object.
- const uint32 EDL_ERR_INVALID_DOM_ID = 505
  An invalid DOMid was encountered.
- const uint32 EDL_ERR_INVALID_TYPE3_GLYPH = 506
  An invalid Type 3 Glyphs was encountered.
- const uint32 EDL_ERR_INVALID_GLYPHS = 507
  An invalid Glyphs node was encountered.
- const uint32 EDL_ERR_INVALID_ZIP = 1001
  XPS input: Invalid ZIP file encountered.
- const uint32 EDL_ERR_NOCONTENTTYPE = 1002
  XPS input: Unable to retrieve the content type for the part.
- const uint32 EDL_ERR_PROCESS_PART = 1003
  XPS input: Error processing the part.
- const uint32 EDL_ERR_XPS_PROVIDER = 1004
  XPS input: XPS provider error.
XPS input: Missing resource.

- \texttt{const uint32 EDL\_ERR\_DTD\_CONTENT} = 1006
  XPS input: DTD content detected.

- \texttt{const uint32 EDL\_ERR\_MISSING\_REQ\_ATTRIBUTE} = 1007
  XPS input: An element is missing a required attribute.

- \texttt{const uint32 EDL\_ERR\_INVALID\_ATTRIBUTE} = 1008
  XPS input: An invalid attribute has been encountered in the element.

- \texttt{const uint32 EDL\_ERR\_PROPERTY\_ALREADY\_SET} = 1009
  XPS input: Attempt to set the same property twice.

- \texttt{const uint32 EDL\_ERR\_REMOTERESOURCE\_REF} = 1010
  XPS input: Remote resource dictionary references another remote resource dictionary.

- \texttt{const uint32 EDL\_ERR\_KEY\_SET\_NON\_RESOURCE} = 1011
  XPS input: Key attribute is set for an element that is not in the resource dictionary.

- \texttt{const uint32 EDL\_ERR\_UNsupported\_GLYPHs} = 1012
  XPS input: Unsupported glyphs encountered.

- \texttt{const uint32 EDL\_ERR\_INVALID\_URI} = 1013
  XPS input: Invalid URI encountered.

- \texttt{const uint32 EDL\_ERR\_DEFLATE} = 1014
  XPS input: Deflate error.

- \texttt{const uint32 EDL\_ERR\_NO\_KEY\_RESDICT\_OBJECT} = 1015
  XPS input: Resource dictionary is missing the key attribute.

- \texttt{const uint32 EDL\_ERR\_ADD\_OBJECT\_RESDICT} = 1016
  XPS input: Error adding the element to the resource dictionary.

- \texttt{const uint32 EDL\_ERR\_INVALID\_ABBR\_PATH} = 1017
  XPS input: Error processing abbreviated path.

- \texttt{const uint32 EDL\_ERR\_NAMESPACE\_NOT\_UNDERSTOOD} = 1018
  XPS input: Namespace is not understood.

- \texttt{const uint32 EDL\_ERR\_XML\_PARSER} = 1019
  XPS input: Parser error encountered.

- \texttt{const uint32 EDL\_ERR\_NO\_REQ\_ELEMENT} = 1020
  XPS input: Element is missing a required subelement.

- \texttt{const uint32 EDL\_ERR\_MORE\_THAN\_ONE\_ELEMENT} = 1021
  XPS input: Element has more than one instance of a subelement.

- \texttt{const uint32 EDL\_ERR\_INVALID\_ELEMENT} = 1022
  XPS input: Element contains an invalid subelement.

- \texttt{const uint32 EDL\_ERR\_FALLBACK\_BEFORE\_CHOICE} = 1023
  XPS input: Fallback element has been found before choice.

- \texttt{const uint32 EDL\_ERR\_PREFIX\_NOT\_DEFINED} = 1024
  XPS input: Undefined prefix encountered.

- \texttt{const uint32 EDL\_ERR\_UNPREFIXED\_ATTRIBUTE} = 1025
  XPS input: Unprefixed attribute encountered.

- \texttt{const uint32 EDL\_ERR\_BROKEN\_ELEMENT\_SEQUENCE} = 1026
  XPS input: Broken element sequence encountered.

- \texttt{const uint32 EDL\_ERR\_PREFIXED\_ATTRIBUTE} = 1027
  XPS input: Prefixed attribute encountered.

- \texttt{const uint32 EDL\_ERR\_INVALID\_IMAGE} = 1028
  XPS input: Invalid image encountered.

- \texttt{const uint32 EDL\_ERR\_NUMBER\_REQ\_ELEMENT} = 1029
  XPS input: Insufficient elements of a particular type encountered.

- \texttt{const uint32 EDL\_ERR\_MISSING\_XPS\_PART} = 1030
  XPS input: Missing XPS part.
• const uint32 EDL_ERR_PRINT_TICKET_EXISTS = 1031
  XPS input: PrintTicket already exists.
• const uint32 EDL_ERR_MULTIPLE_STARTPARTS_RELS = 1032
  XPS input: Multiple StartPart relationships encountered.
• const uint32 EDL_ERR_DICT_ITEM_PRESENT = 1033
  XPS input: Multiple items with the same key present.
• const uint32 EDL_ERR_MULTIPLE_COREPROPERTIES_RELS = 1034
  XPS input: Multiple CoreProperty relationships encountered.
• const uint32 EDL_ERR_MULTIPLE_DOCSTRUCTURE_RELS = 1035
  XPS input: Multiple DocumentStructure relationships encountered.
• const uint32 EDL_ERR_MULTIPLE_THUMBNAIL_RELS = 1036
  XPS input: Multiple thumbnail relationships encountered.
• const uint32 EDL_ERR_INVALID_THUMBNAIL_TYPE = 1037
  XPS input: Invalid thumbnail type encountered.
• const uint32 EDL_ERR_PRESERVED_ITEMS_NOT_DECLARED_IGNORABLE = 1038
  XPS input: Unable to preserve items! The namespace is not declared as ignorable.
• const uint32 EDL_ERR_MORE_THAN_ONE_GROUP_ELEMENT = 1039
  XPS input: Element contains more than one subelement.
• const uint32 EDL_ERR_NO_REQ_GROUP_ELEMENT = 1040
  XPS input: Element requires one subelement.
• const uint32 EDL_ERR_INVALID_COLOR_SPECIFICATION = 1041
  XPS input: Invalid color specification encountered.
• const uint32 EDL_ERR_PROCESSCONTENT_NOT_DECLARED_IGNORABLE = 1042
  XPS input: Namespace in ProcessContent attribute is not declared as ignorable.
• const uint32 EDL_ERR_DUPLICATE_URI = 1043
  XPS input: Duplicate URI encountered.
• const uint32 EDL_ERR_UNEXPECTED_RESOURCE = 1300
  PDF output: Unexpected resource encountered.
• const uint32 EDL_ERR_DEVICE_OUTOFMEMORY = 1301
  PDF output: Insufficient device memory to process page.
• const uint32 EDL_ERR_XPS_FROM_ENCRYPTED_SOURCE = 1302
  PDF output: Attempt to create XPS from an encrypted document.
• const uint32 EDL_ERR_INTERNAL_RIP = 2000
  Internal RIP error.
• const uint32 EDL_ERR_OPEN_PDF = 2300
  PDF input: Error opening PDF.
• const uint32 EDL_ERR_INVALID_PDF_PASSWORD = 2301
  PDF input: Error opening encrypted PDF; Invalid password.
• const uint32 EDL_ERR_PSOUT_GENERAL_FAILURE = 2400
  PS output: General error.
• const uint32 EDL_ERR_PDFOUT_GENERAL_FAILURE = 2401
  PDF output: General error.
• const uint32 EDL_ERR_INVALID_FONT = 2402
  PDF output: Error processing a font.
PDF/A output: Incompatible content found.

- const uint32 EDL_ERR_INCOMPATIBLE_PDFX = 2404

PDF/X output: Incompatible content found.

- const uint32 EDL_ERR_BAD_COLOR_SPACE = 2500
  
  Color: A bad color space was encountered.

- const uint32 EDL_ERR_COLOR_CONVERSION_FAILURE = 2501
  
  Color: Color conversion failed.

- const uint32 EDL_ERR_INCOMPATIBLE_IMAGE_PARAMETER = 2600
  
  Image: An image encoder has been given an image or parameter that it cannot handle.

- const uint32 EDL_ERR_IMAGE_PARAMETER_OUT_OF_RANGE = 2601
  
  Image: An image parameter for encoding or decoding is outside the expected range.

- const uint32 EDL_ERR_IMAGE DECODE_FAILURE = 2602
  
  Image: An image could not be decoded.

- const uint32 EDL_ERR_IMAGE_ENCODE FAILURE = 2603
  
  Image: An image could not be encoded.

- const uint32 EDL_ERR_ ZIP_WRITE ONLY = 2700
  
  Zip: An attempt was made to retrieve a stream from a write-only Zip file.

- const uint32 EDL_ERR_ZIP_PATH NOT FOUND = 2701
  
  Zip: An expected entry in a Zip package was not found.

### 9.4.1 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.

There is a separate file "edlerrors.cpp" that provides a look-up function to convert these integer result codes into (localizable) strings

### 9.4.2 Function Documentation

#### 9.4.2.1 getEDLErrorString()

```cpp
EDLString getEDLErrorString (  
    uint32 errorCode  )
```

Implements a simple numeric-error-code-to-corresponding-language-specific-error-message-string look-up using the "errorList" array/table.

**Parameters**

<table>
<thead>
<tr>
<th>errorCode</th>
<th>The error code number</th>
</tr>
</thead>
</table>

Generated by Doxygen
9.5 edlfwd.h File Reference

A header containing forward declarations for certain EDL interfaces.

```c
#include <edl/edlnamespaces.h>
#include <edl/smartptr.h>
```

9.5.1 Detailed Description

A header containing forward declarations for certain EDL interfaces.

9.6 edlgeom.h File Reference

Geometry primitives including: point, rectangle and matrix types supporting both integer and floating point values within.

```c
#include <edl/edltypes.h>
#include <edl/edlmath.h>
#include <edl/edlvector.h>
```

Classes

- class `CTransformMatrix<TItem>`

  Matrix class - special 3x2 matrix.
9.6.1 Detailed Description

Geometry primitives including: point, rectangle and matrix types supporting both integer and floating point values within.

9.7 edlmath.h File Reference

(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

```
#include <edl/edlnamespaces.h>
#include <math.h>
```

Macros

- `#define PI 3.14159265358979323846`
  
  *Local definition of PI to 20 decimal places.*

9.7.1 Detailed Description

(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

9.7.2 Macro Definition Documentation

9.7.2.1 PI

```
#define PI 3.14159265358979323846
```

*Local definition of PI to 20 decimal places.*

This should give sufficient precision for calculations using double-precision floating-point math

9.8 edlnamespaces.h File Reference

EDL C++ namespace(s)
9.8.1 Detailed Description

EDL C++ namespace(s)

This file defines a collection of macros that collectively define the namespace(s) that EDL provides.

For each namespace (the primary one being "EDL") there is a collection of macros:

BEGIN<\<x\>>_NAMESPACE intended to mark the start of a series of declarations/definitions that are in namespace \<\<x\>>. This is typically used near the start of the EDL header files after all the included dependent header files before any local declarations

END<\<x\>>_NAMESPACE intended to mark the end of a series of declarations Typically used at the end of an EDL header file

USING<\<x\>>_NAMESPACE is used whenever EDL code wishes to use types, data, functions from within namespace \<\<x\>> without having to explicitly specify namespace \"\<x\>\::\" everywhere This is typically found at the start of most C++ source files.

FROM<\<x\>>_NAMESPACE is used for isolated individual locations where a type, data item or function from namespace \<\<x\>> is needed and where a explicit namespace qualifier is needed for disambiguation This macro takes a single argument which is the name of the type, data item or function that is in the particular \<\<x\>> namespace

Note that for all EDL namespaces, the above-named macros are always defined.

However they are only defined to expand to some namespace-specific text when both __cplusplus and the _\<\<x\>\<-\_<\<x\>>_NAMESPACE macro is defined

9.9 edlproperty.h File Reference

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.)

#include <edl/edltypes.h>
#include <edl/edltime.h>
#include <edl/iedlobject.h>
#include <edl/edlstring.h>
#include <edl/edlqname.h>
#include <edl/edlerrors.h>

Classes

- class PValue
  Stores a "property" value that is tagged with an enumeration value that indicates the underlying type.

9.9.1 Detailed Description

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.)

Generated by Doxygen
9.10 edlqname.h File Reference

IEDLNamespace and EDLQName classes.

```cpp
#include <edl/edltypes.h>
#include <edl/iedlobject.h>
#include <edl/iedlenum.h>
#include <edl/edlstring.h>
```

Classes

- class IEDLNamespace
  * Interface to EDL Namespace class.
- class IEDLNamespace::Data
  * Initialization data.
- class EDLQName
  * Implementation of qualified name class.

9.10.1 Detailed Description

IEDLNamespace and EDLQName classes.

9.11 edlquartz.h File Reference

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.

```cpp
#include "TargetConditionals.h"
#import <Cocoa/Cocoa.h>
#include <edl/isession.h>
#include <edl/edltypes.h>
#include <edl/idomnode.h>
#include <edl/edlgeom.h>
```

9.11.1 Detailed Description

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.

9.12 edlrenderingintent.h File Reference

Simple enum for rendering intent.

```cpp
#include <edl/edlnamespaces.h>
```
9.13 edlstream.h File Reference

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

```cpp
#include <fstream>
#include <edl/edltypes.h>
#include <edl/edlstream.h>
#include <edl/iedlobject.h>
#include <edl/idomhashable.h>
#include <edl/edlvector.h>
```

### 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.

### Typedefs

- typedef int(* UserStreamWriteFunc)(void *pPriv, void *pBuff, unsigned int len)
  Type definition of a callback function to receive streamed output.

- typedef int(* UserStreamReadFunc)(void *pPriv, void *pBuff, unsigned int len, unsigned int *pLenRead, int *pEof)
  Callback typedef for streaming input.

- typedef int(* UserRAReadFunc)(void *pPriv, void *pBuff, int64 offset, int32 length)
  Callback typedef for user defined random access input.

- typedef void(* FileStreamReadFunc)(void *pPriv, int64 pos, int32 len)
  Callback typedef for CFileStreamWithCallback that is used to notify a client of reads from stream.

### 9.13.1 Detailed Description

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

### 9.13.2 Typedef Documentation

#### 9.13.2.1 FileStreamReadFunc

typedef void(* FileStreamReadFunc)(void *pPriv, int64 pos, int32 len)

Callback typedef for CFileStreamWithCallback that is used to notify a client of reads from stream.
Parameters

<table>
<thead>
<tr>
<th>pPriv</th>
<th>A private pointer that is passed with each invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>pos</td>
<td>Starting position for this read.</td>
</tr>
<tr>
<td>len</td>
<td>The number of bytes that were read.</td>
</tr>
</tbody>
</table>

### 9.13.2.2 UserRAReadFunc

typedef int(* UserRAReadFunc) (void *pPriv, void *pBuff, int64 offset, int32 length)

Callback typedef for user defined random access input.

For correct operation, this callback function must be thread safe as it is possible that multiple threads may call this function simultaneously.

Parameters

<table>
<thead>
<tr>
<th>pPriv</th>
<th>A private pointer that is passed with each invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>pBuff</td>
<td>The buffer to read data into</td>
</tr>
<tr>
<td>offset</td>
<td>The position within the input from which the data will be read.</td>
</tr>
<tr>
<td>len</td>
<td>The number of bytes that are requested for the read.</td>
</tr>
</tbody>
</table>

Returns

int Zero on success, non-zero on failure.

### 9.13.2.3 UserStreamReadFunc

typedef int(* UserStreamReadFunc) (void *pPriv, void *pBuff, unsigned int len, unsigned int *pLenRead, int *pEof)

Callback typedef for streaming input.

Parameters

<table>
<thead>
<tr>
<th>pPriv</th>
<th>A private pointer that is passed with each invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>pBuff</td>
<td>The buffer to read data into</td>
</tr>
<tr>
<td>len</td>
<td>The number of bytes that are requested for the read.</td>
</tr>
<tr>
<td>pLenRead</td>
<td>A pointer to the number of bytes actually read.</td>
</tr>
<tr>
<td>pEof</td>
<td>A pointer to a int value that indicates whether the end of file (EOF) was read. *pEof = 1 if end of file, *pEof = 0 otherwise</td>
</tr>
</tbody>
</table>
Returns

**int** Zero on success, non-zero on failure.

9.13.2.4 UserStreamWriteFunc

typedef int(* UserStreamWriteFunc) (void *pPriv, void *pBuff, unsigned int len)

Type definition of a callback function to receive streamed output.

Pointers to functions of this type can be passed to EDL via the `outputcallback` parameter. Any implementations of this type are expected to return zero on success, and a non-zero error code if the call fails. They should also be able to handle the special case where `pBuff` is set to NULL and `len` is set to 0. The stream writer passes these values to signal an aborted output stream. That is, no more data will be written to and incomplete stream, due to an error upstream in the process.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pPriv</code></td>
<td>A private pointer that is passed with each invocation.</td>
</tr>
<tr>
<td><code>pBuff</code></td>
<td>The buffer containing the data to be written.</td>
</tr>
<tr>
<td><code>len</code></td>
<td>The length of the data in the buffer. 0 indicates that the stream is ending.</td>
</tr>
</tbody>
</table>

Returns

**int** Zero on success, non-zero on failure.

9.14 edlstring.h File Reference

EDLString and EDLSysString classes and associated EDL string manipulation functions.

```cpp
#include <string>
#include <sstream>
#include <limits>
#include <edl/edltypes.h>
#include <edl/edlvector.h>
```

Functions

- `uint8 sxtob (char c)`
  sxtob converts a hexadecimal character into corresponding (unsigned) integer value. (Note that there is no guard against being supplied with non-hexadecimal characters)

- `bool edlstringEqualsIgnoreCaseCnt (const EDLString &str1, const EDLString &str2, size_t len)`
  edlstringEqualsIgnoreCaseCnt performs a case-insensitive maximum-length-constrained string equality test

- `size_t edlstringFind (const EDLString &str, EDLString &resultStr)`
  edlstringFind searches for a substring within a larger string.

- `void edlstringSubstr (const EDLString &str, EDLString &result, size_t start, size_t end)`
  edlstringSubstr returns a substring from the source string starting at the specified index.
**edlstringSubstr** returns a substring from specified start and end points from a larger string

- EDL_API EDLString **EDLSysStringToEDLString** (const EDLSysString &edlSysString)
  
  *EDLSysStringToEDLString* converts an EDLSysString (UTF8) to an EDLString (UTF16 or UTF32 depending on platform).

- EDL_API EDLSysString **EDLStringToEDLSysString** (const EDLString &edlString)
  
  *EDLStringToEDLSysString* converts an (UTF16 or UTF32 depending on platform) EDLString to an EDLSysString (UTF8).

### 9.14.1 Detailed Description

EDLString and EDLSysString classes and associated EDL string manipulation functions.

### 9.14.2 Function Documentation

#### 9.14.2.1 edlstringEqualsIgnoreCaseCnt()

```cpp
bool edlstringEqualsIgnoreCaseCnt ( 
    const EDLString & str1,        
    const EDLString & str2,        
    size_t len) [inline]
```

*edlstringEqualsIgnoreCaseCnt* performs a case-insensitive maximum-length-constrained string equality test

**Parameters**

<table>
<thead>
<tr>
<th></th>
<th>First string</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>str1</strong></td>
<td>First string</td>
</tr>
<tr>
<td><strong>str2</strong></td>
<td>Second string</td>
</tr>
<tr>
<td><strong>len</strong></td>
<td>Number of characters to compare</td>
</tr>
</tbody>
</table>

**Returns**

- **bool** True if the two strings are equal, false if not

#### 9.14.2.2 edlstringFind()

```cpp
size_t edlstringFind ( 
    const EDLString & str,      
    EDLString & searchStr) [inline]
```

*edlstringFind* searches for a substring within a larger string.
Parameters

<table>
<thead>
<tr>
<th>str</th>
<th>String to be searched</th>
</tr>
</thead>
<tbody>
<tr>
<td>searchStr</td>
<td>Substring to search for</td>
</tr>
</tbody>
</table>

Returns

size_t The starting position of the (first instance of the) substring.

9.14.2.3 edlstringSubstr()

```c
void edlstringSubstr (
    const EDLString & str,
    EDLString & result,
    size_t start,
    size_t end ) [inline]
```

edlstringSubstr returns a substring from specified start and end points from a larger string

Parameters

<table>
<thead>
<tr>
<th>str</th>
<th>Source string</th>
</tr>
</thead>
<tbody>
<tr>
<td>result</td>
<td>String to accept result</td>
</tr>
<tr>
<td>start</td>
<td>Start point</td>
</tr>
<tr>
<td>end</td>
<td>End point</td>
</tr>
</tbody>
</table>

9.14.2.4 EDLStringToEDLSysString()

```c
EDL_API EDLSysString EDLStringToEDLSysString ( const EDLString & ediString )
```

EDLStringToEDLSysString converts an (UTF16 or UTF32 depending on platform) EDLString to an EDLSysString (UTF8).

Parameters

| ediString | Source string |

Returns

EDLSysString Result of conversion
9.14.2.5 EDLSysStringToEDLString()

EDL_API EDLString EDLSysStringToEDLString ( const EDLSysString & edlSysString )

EDLSysStringToEDLString converts an EDLSysString (UTF8) to an EDLString (UTF16 or UTF32 depending on platform).

Parameters

| edlSysString | Source string |

Returns

| EDLString | Result of conversion |

9.14.2.6 sxtob()

uint8 sxtob ( char c ) [inline]

sxtob converts a hexadecimal character into corresponding (unsigned) integer value. (Note that there is no guard against being supplied with non-hexadecimal characters)

Parameters

| c | The hexadecimal digit |

Returns

| uint8 | The unsigned integer value |

9.15 edltime.h File Reference

IEDLTime class that represents the EDL date-time.

#include <edl/iedlobject.h>
#include <edl/iedlfactory.h>
#include "edltypes.h"
#include "edlstring.h"
#include <time.h>

Classes

- class IEDLTime
  
  Interface to EDL date-time class.
- class IEDLTime::Data
  
  Initialization data.
9.15.1 Detailed Description

IEDLTime class that represents the EDL date-time.

9.16 edltypes.h File Reference

EDL "standard" types including known bit-length signed and unsigned integer type[def]s and definitions of EDL_TRUE and EDL_FALSE.

```c
#include <stddef.h>
#include "platform.h"
#include <edl/edlnamespaces.h>
```

9.16.1 Detailed Description

EDL "standard" types including known bit-length signed and unsigned integer type[def]s and definitions of EDL_TRUE and EDL_FALSE.

9.17 edlvector.h File Reference

Simple buffer class which preserves the allocation context where the buffers were created.

```c
#include <edl/edltypes.h>
#include <edl/edlnamespaces.h>
#include <edl/edlallocator.h>
```

9.17.1 Detailed Description

Simple buffer class which preserves the allocation context where the buffers were created.

Simple template vector class for general use.

9.18 edlversion.h File Reference

defines EDL major and minor version numbers

9.18.1 Detailed Description

defines EDL major and minor version numbers
9.19 icolormanager.h File Reference

Public interface to the EDL color manager.

```cpp
#include <edl/iedlfactory.h>
#include <edl/iedlobject.h>
#include <edl/idomresources.h>
#include <edl/idomcolor.h>
#include <edl/idomcolorspace.h>
#include <edl/idomcolorspace.h>
#include <edl/edlvector.h>
#include <edl/edlsimplebuffer.h>
```

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.

9.19.1 Detailed Description

Public interface to the EDL color manager.

9.20 idombrush.h File Reference

Interfaces to DOM brush objects.

```cpp
#include <edl/edlnamespaces.h>
#include <edl/idomnode.h>
#include <edl/idomhashable.h>
#include <edl/edltypes.h>
#include <edl/iedlenum.h>
#include <edl/idomcolor.h>
#include <edl/idomfunction.h>
#include <edl/idomimageresource.h>
#include <edl/idomgroup.h>
```

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 IDOMGradientStop::Data**
  
  Initialization data.
- **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 IDOMSolidColorBrush::Data
  Initialization data.
- 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 IDOMLinearGradientBrush::Data
  Initialization data.
- 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 IDOMRadialGradientBrush::Data
  Initialization data.
- class IDOMImageBrush
  Provides an interface to a DOM image brush object.
- class IDOMImageBrush::Data
  Initialization data.
- 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 IDOMMaskedBrush::Data
  Initialization data.
- class IDOMVisualBrush
  A visual brush is used to fill a region with a vector drawing.
- class IDOMVisualBrush::Data
  Initialization data.
- 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 IDOMSoftMaskBrush::Data
  Initialization data.
- class IDOMTilingPatternBrush
  IDOMTilingPatternBrush provides a way of representing a PS style tiling pattern.
- class IDOMTilingPatternBrush::Data
  Initialization data.
- class IDOMShadingPatternBrush
  IDOMShadingBrush provides a way of representing a PS style shading pattern.
- class IDOMShadingPatternType1Brush
  IDOMShadingPatternType1Brush provides a way of representing a PS style type 1 shading pattern.
- class IDOMShadingPatternType1Brush::Data
  Initialization data.
IDOMShadingBrush provides a way of representing a PS style type 2 shading pattern.

- class IDOMShadingPatternType2Brush::Data
  Initialization data.

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

- class IDOMShadingPatternType3Brush::Data
  Initialization data.

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

- class IDOMShadingPatternType4567Brush::Data
  Initialization data.

- 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.

- class IDOMNullBrush::Data
  Initialization data.

Enumerations

- 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.

9.20.1 Detailed Description

Interfaces to DOM brush objects.

9.20.2 Enumeration Type Documentation

9.20.2.1 eColorInterpolationMode

enum eColorInterpolationMode

Color interpolation mode type enumeration.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eSRgbLinearInterpolation</td>
<td>Linear interpolation for sRGB color space.</td>
</tr>
<tr>
<td>eSCRgbLinearInterpolation</td>
<td>Linear interpolation for scRGB color space.</td>
</tr>
</tbody>
</table>
9.20.2.2 eSpreadMethod

enum eSpreadMethod

Spread Method type enumeration.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ePad</td>
<td>Fill the remaining area with the color specified by the final gradient stop.</td>
</tr>
<tr>
<td>eReflect</td>
<td>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.</td>
</tr>
<tr>
<td>eRepeat</td>
<td>Fill the remaining area by repeating the gradient from its first gradient stop, starting at the point defined by the last gradient stop.</td>
</tr>
<tr>
<td>eNoSpread</td>
<td>Do not fill the remaining area, but allow it to remain transparent.</td>
</tr>
</tbody>
</table>

9.20.2.3 eTilingMode

enum eTilingMode

Tiling mode type enumeration.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eNoTile</td>
<td>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.</td>
</tr>
<tr>
<td>eTile</td>
<td>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.</td>
</tr>
<tr>
<td>eFlipX</td>
<td>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.</td>
</tr>
<tr>
<td>eFlipY</td>
<td>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.</td>
</tr>
<tr>
<td>eFlipXY</td>
<td>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.</td>
</tr>
</tbody>
</table>

9.20.2.4 eViewUnits

enum eViewUnits

View units type enumeration.
9.21  idomcanvas.h File Reference

**IDOMCanvas** interface.

```c
#include <edl/idomgroup.h>
#include <edl/idomid.h>
#include <edl/idomresources.h>
#include <edl/edlstring.h>
#include <edl/idomtarget.h>
#include <edl/idomedgemode.h>
```

**Classes**

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

- **class IDOMCanvas::Data**
  
  Initialization data.

9.21.1  Detailed Description

**IDOMCanvas** interface.

Interface to DOM node representing `<Canvas>` element

9.22  idomcatalog.h File Reference

**IDOMCatalog** Interface.

```c
#include <edl/edltypes.h>
#include <edl/edlstring.h>
#include <edl/iedlobject.h>
#include <edl/idomid.h>
#include <edl/idomnode.h>
```

**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.
9.22.1 Detailed Description

**IDOMCatalog** Interface.

CDOM Catalog provides association of addressable CDOM Nodes with CDOM Node IDs. Provides address space for the CDOMs in the context of a single EDL job.

9.23 idomcharpathgroup.h File Reference

**IDOMCharPathGroup** interface.

```c
#include <edl/idomnode.h>
#include <edl/edlnamespaces.h>
#include <edl/idomgroup.h>
#include <edl/edlblend.h>
```

**Classes**

- class **IDOMCharPathGroup**
  
  **IDOMCharPathGroup** interface.

- class **IDOMCharPathGroup::Data**
  
  Initialization data.

9.23.1 Detailed Description

**IDOMCharPathGroup** interface.

Interface to DOM node representing Group Elements that consist of stroked text.

9.24 idomcolor.h File Reference

**IDOMColor** Interface.

```c
#include <edl/edlrenderingintent.h>
#include <edl/edlblackpointcompensation.h>
#include <edl/edltypes.h>
#include <edl/iedlobject.h>
#include <edl/idomcolorspace.h>
#include <edl/idomresources.h>
#include <edl/idomhashable.h>
#include <edl/edlvector.h>
```

**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.
9.25 idomcolorspace.h File Reference

idomcolorspace Interface

#include <stdarg.h>
#include <edl/edlrenderingintent.h>
#include <edl/edlblackpointcompensation.h>
#include <edl/edlmath.h>
#include <edl/edltypes.h>
#include <edl/iedlobject.h>
#include <edl/idomresources.h>
#include <edl/idomfunction.h>
#include <edl/edlvector.h>
#include <edl/idomhashable.h>

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 IDOMColorSpaceICCBased::Data
  Initialization data.
• class IDOMColorSpaceIndexed
  This color space is analogous to the PostScript/PDF Indexed color space.
• class IDOMColorSpaceIndexed::Data
  Initialization data.
• class IDOMDeviceNColorant
  This class enables the specification of colorant information for PDF style NChannel variants of DeviceN color spaces.
• class IDOMDeviceNColorant::Data
Initialization data.

• class IDOMColorSpaceDeviceN
  This color space is analogous to the PostScript/PDF DeviceN/Separation color spaces.

• class IDOMColorSpaceDeviceN::Data
  Initialization data.

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

• class IDOMColorSpaceLAB::Data
  Initialization data.

9.25.1 Detailed Description

idomcolorsapce Interface

9.26 idomedgemode.h File Reference

Enumeration for edge mode, used to flag when a node should not be rendered using anti-aliasing on anti-aliasing renderers.

#include <edl/edlnamespaces.h>

Enumerations

• enum eEdgeMode { eEMDefault, eEMAliased }
  Available options for the RenderOptionsEdgeMode property, which controls anti-aliasing behavior.

9.26.1 Detailed Description

Enumeration for edge mode, used to flag when a node should not be rendered using anti-aliasing on anti-aliasing renderers.

9.26.2 Enumeration Type Documentation

9.26.2.1 eEdgeMode

enum eEdgeMode

Available options for the RenderOptionsEdgeMode property, which controls anti-aliasing behavior.
Enumerator

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eEMDefault</td>
<td>Use default anti-aliasing behavior when rendering the contents of this node. Anti-aliasing consumers will perform anti-aliasing.</td>
</tr>
<tr>
<td>eEMAliased</td>
<td>Do not perform anti-aliasing when rendering the contents of this node or its children.</td>
</tr>
</tbody>
</table>

9.27 idomfont.h File Reference

**IDOMFont** interface.

```c
#include <edl/edltypes.h>
#include <edl/edlstream.h>
#include <edl/idomnode.h>
#include <edl/idomglyph.h>
#include <edl/isession.h>
#include <edl/edlvector.h>
#include <edl/idomhashable.h>
```

**Classes**

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

- class **IDOMFontSource::Data**
  
  Initialization data.

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

- class **IDOMFontSourceStreamFilter::Data**
  
  Initialization data.

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

- class **IDOMFontSourceFromStream::Data**
  
  Initialization data.

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

- class **IDOMFontSourceObfuscationConverter::Data**
  
  Initialization data.

- class **IDOMFont**
  
  **IDOMFont** Base Class.

- class **IDOMFont::Data**
  
  Initialization data.

- class **IDOMFontOpenType**
  
  **IDOMFontOpenType** interface.

- class **IDOMFontOpenType::Data**
  
  Initialization data.

- class **IDOMFontOpenTypeTT::Data**
  
  Initialization data.

- class **IFontHeaderWriteSegmentBlockEnumerator**
IFontHeaderWriteSegmentBlockEnumerator Enumerates over the PCLXL Font Header block items for the XL ReadFontHeader operator.

• class IFontPCL5WriteSegmentBlockEnumerator
  IFontPCL5WriteSegmentBlockEnumerator Enumerates over the PCL5 font blocks.

• class IDOMType3Font
  Representation of a PostScript/PDF Type 3 Font. At present, the stream cannot be set, only retrieved.

• class IDOMType3Font::Data
  Initialization data.

• class IEDLFontSystemFont
  Representation of fonts installed on the target system (OS dependant).

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

• class IDOMFontPCLXL::Data
  Initialization data.

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

• class IDOMFontPCL5::Data
  Initialization data.

Typedefs

• typedef uint32 TableSignatureID
type used to uniquely identify an opentype font signature

9.27.1 Detailed Description

IDOMFont interface.

9.28 idomform.h File Reference

IDOMGroup interface.

#include <edl/edlblend.h>
#include <edl/idomnode.h>
#include <edl/idombrush.h>
#include <edl/edlnamespaces.h>
#include <jawsmako/types.h>

Classes

• class IDOMForm
  IDOMForm interface. 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.

• class IDOMForm::Data
  Initialization data.

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

• class IDOMFormInstance::Data
  Initialization data.
9.28.1 Detailed Description

`IDOMGroup` interface.

Interface to DOM node representing PDF-style Form Elements

---

### 9.29 idomglyph.h File Reference

`IDOMGlyph` Interface.

```c
#include <edl/edltypes.h>
#include <edl/edlgeom.h>
#include <edl/idomnode.h>
```

#### Classes

- class `IDOMGlyphName`
- class `IDOMGlyph`
  
  *The `IDOMGlyph` class is an abstract class modelling a single character from a font.*
- class `IDOMGlyph::Data`
  
  *Initialization data.*
- class `IDOMGlyph::IDEnumerator`
  
  *DOM Glyph ID Enumerator.*

---

### 9.29.1 Detailed Description

`IDOMGlyph` Interface.

---

### 9.30 idomglyphs.h File Reference

`IDOMGlyphs` Interface.

```c
#include <edl/edltypes.h>
#include <edl/edlgeom.h>
#include <edl/idomnode.h>
#include <edl/idombrush.h>
#include <edl/idompathgeometry.h>
#include <edl/idomgroup.h>
#include <edl/idomfont.h>
#include <edl/idomglyph.h>
#include <edl/idomnode.h>
#include <edl/idomresources.h>
#include <edl/idomtarget.h>
#include <edl/idompath.h>
```
Classes

- 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 IDOMGlyphs::Data
  
  Initialization data.

9.30.1 Detailed Description

IDOMGlyphs Interface.

9.31 idomgroup.h File Reference

IDOMGroup interface.

```cpp
#include <edl/idomnode.h>
#include <edl/edlnamespaces.h>
#include <edl/edlfwd.h>
#include <edl/edlblend.h>
#include <edl/idombrush.h>
#include <edl/idomcolorspace.h>
#include <edl/idompathgeometry.h>
#include <jawsmako/types.h>
```

Classes

- class IDOMGroup
  
  IDOMGroup interface.

- class IDOMGroup::Data
  
  Initialization data.

- class IDOMTransparencyGroup
  
  IDOMTransparencyGroup interface. Analogous to PDF Transparency groups.

- class IDOMTransparencyGroup::Data
  
  Initialization data.

9.31.1 Detailed Description

IDOMGroup interface.

Interface to DOM node representing Group Elements
9.32 idomhashable.h File Reference

Abstract interface for objects that can be hashed.

#include <edl/edlnamespaces.h>
#include <edl/iedlobject.h>

Classes

- class IDOMHashable
  
  Abstract interface for EDL objects that may be hashed.

9.32.1 Detailed Description

Abstract interface for objects that can be hashed.

9.33 idomid.h File Reference

DOMid Interface.

#include <edl/edltypes.h>

Typedefs

- typedef EDL::uint64 DOMid
  Type used to uniquely identify a DOM node.

Functions

- EDL_API DOMid allocateNewDOMid ()
  Allocate a unique DOMid.

9.33.1 Detailed Description

DOMid Interface.

9.33.2 Function Documentation
9.33.2.1 allocateNewDOMid()

EDL_API DOMid allocateNewDOMid ( )

Allocate a unique DOMid.

Returns

DOMid The new DOMid

9.34 idomimageresource.h File Reference

IDOMResources interface.

#include <edl/edltypes.h>
#include <edl/edlstream.h>
#include <edl/idomnode.h>
#include <edl/imagecodec.h>
#include <edl/idomcolor.h>
#include <edl/idomhashable.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.

- class IDOMImage::Data
  Initialization data.

- 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 IDOMWMPImage
  IDOMWMPImage interface.

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

- class IDOMRawImage::Data
  Initialization data.

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

- class IDOMPDFImage::IDecodeParams
  Abstract interface for per-image decoding filter parameters.

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

- **class IDOMPDFImage::JBIG2Params**
  Class to hold filter parameters for JBIG2-compressed image data. Please see the PDF specification for the meaning of these parameters.

- **class IDOMPDFImage::DCTParams**
  Class to hold filter parameters for DCT-compressed image data. Please see the PDF specification for the meaning of these parameters.

- **class IDOMPDFImage::Data**
  Initialization data.

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

- **class IDOMPCLImage::Data**
  Initialization data.

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

- **class IDOMRecombineImage::Data**
  Initialization data.

- **class IDOMRecombineAlphaImage::Data**
  Initialization data.

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

- **class IDOMCompositeImage::Data**
  Initialization data.

- **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).

- **class IDOMImageChannelSelectorFilter::Data**
  Initialization data.

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

- **class IDOMImageColorSpaceSubstitutionFilter::Data**
  Initialization data.

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

- **class IDOMImageColorConverterFilter::Data**
  Initialization data.

- **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 IDOMImageBleederFilter::Data**
  Initialization data.

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

- **class IDOMImageDownsamplerFilter::Data**
  Initialization data.

- **class IDOMImageMaskExpanderFilter**
  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.
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 a IDOMMaskedBrush where the brush masked by the image is a solid color.

- **class IDOMImageMaskExpanderFilter::Data**
  - Initialization data.

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

- **class IDOMImageDeindexerFilter::Data**
  - Initialization data.

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

- **class IDOMImageDeviceNToBaseFilter::Data**
  - Initialization data.

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

- **class IDOMImageInverterFilter::Data**
  - Initialization data.

- **class IDOMDePremultiplierFilter::Data**
  - Initialization data.

- **class IDOMImageMatteRemoverFilter::Data**
  - Initialization data.

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

- **class IDOMImageBitScalerFilter::Data**
  - Initialization data.

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

- **class IDOMImageColorKeyFilter::Data**
  - Initialization data.

- **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 IDOMImageDecodeFilter::Data**
  - Initialization data.

- **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 IDOMFilteredImage::Data**
  - Initialization data.

### 9.34.1 Detailed Description

IDOMResources interface.
DOM Job Ticket Interface.

```cpp
#include <edl/edltypes.h>
#include <edl/edlqname.h>
#include <edl/edgeom.h>
#include <edl/idomnode.h>
#include <edl/idomresources.h>
#include <edl/iedlfactory.h>
#include <edl/edlvector.h>
```

### Classes

- **class IDOMJobTkOwner**
  Interface to the `IDOMJobTkOwner` node.

- **class IDOMJobTkGenericNode**
  Interface to the `IDOMJobTkGenericNode` node.

- **class IDOMJobTkGenericNode::Data**
  Initialization data.

- **class IDOMJobTkGenericCharacterData**
  Interface to the `IDOMJobTkGenericCharacterData` node.

- **class IDOMJobTkGenericCharacterData::Data**
  Initialization data.

- **class IDOMJobTkNode**
  Represents a Job Ticket Node.

- **class IDOMJobTkNode::Data**
  Initialization data.

- **class IDOMJobTkValue**
  Represents a Job Ticket value element.

- **class IDOMJobTkValue::Data**
  Initialization data.

- **class IDOMJobTkContent**
  Represents the content element of the JobTicket.

- **class IDOMJobTkContent::Data**
  Initialization data.

- **class IDOMJobTk**
  Represents an EDL JobTicket.

### Enumerations

- **enum eDOMJobTkLevel**
  
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>eDOMJobTkLevelNotValid</td>
</tr>
<tr>
<td>1</td>
<td>eDOMJobTkLevelDefault</td>
</tr>
<tr>
<td>2</td>
<td>eDOMJobTkLevelJob</td>
</tr>
<tr>
<td>3</td>
<td>eDOMJobTkLevelDoc</td>
</tr>
<tr>
<td>4</td>
<td>eDOMJobTkLevelPage</td>
</tr>
<tr>
<td>5</td>
<td>eDOMJobTkLevelCnt</td>
</tr>
</tbody>
</table>

  DOMJobTk Job Ticket Level (docseq, doc, page)
Functions

- IDOMJobTkPtr cloneJobTkTree (IDOMJobTkPtr &jobTk, IEDLClassFactory *factory)
  
  Clone the JobTicket and the whole JobTicket content tree.

- IDOMJobTkContentPtr cloneJobTkContentTree (IDOMJobTkContentPtr &jobTkContent, IEDLClassFactory *factory)
  
  Clone the whole JobTicket content tree.

9.35.1 Detailed Description

DOM Job Ticket Interface.

9.35.2 Enumeration Type Documentation

9.35.2.1 eDOMJobTkLevel

enum eDOMJobTkLevel

DOMJobTk Job Ticket Level (docseq, doc, page)

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eDOMJobTkLevelNotValid</td>
<td>Level value not initialized or invalid.</td>
</tr>
<tr>
<td>eDOMJobTkLevelDefault</td>
<td>The default level of JobTicket.</td>
</tr>
<tr>
<td>eDOMJobTkLevelJob</td>
<td>The document sequence level of JobTicket.</td>
</tr>
<tr>
<td>eDOMJobTkLevelDoc</td>
<td>The document level of JobTicket.</td>
</tr>
<tr>
<td>eDOMJobTkLevelPage</td>
<td>The page level of JobTicket.</td>
</tr>
<tr>
<td>eDOMJobTkLevelCnt</td>
<td>The number of levels.</td>
</tr>
</tbody>
</table>

9.35.3 Function Documentation

9.35.3.1 cloneJobTkContentTree()

IDOMJobTkContentPtr cloneJobTkContentTree ( 
    IDOMJobTkContentPtr & jobTkContent, 
    IEDLClassFactory * factory )

Clone the whole JobTicket content tree.

Parameters

<table>
<thead>
<tr>
<th>jobTkContent</th>
<th>Smart pointer to the source JobTicket content.</th>
</tr>
</thead>
<tbody>
<tr>
<td>factory</td>
<td>Pointer to the class factory interface.</td>
</tr>
</tbody>
</table>
9.36 idommetadata.h File Reference

Returns

IDOMJobTkContentPtr. Returns the cloned JobTicket content.

9.35.3.2 cloneJobTkTree()

IDOMJobTkPtr cloneJobTkTree ( 
    IDOMJobTkPtr & jobTk, 
    IEDLClassFactory * factory )

Clone the JobTicket and the whole JobTicket content tree.

Parameters

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>jobTk</td>
<td>Smart pointer to the source JobTicket.</td>
</tr>
<tr>
<td>factory</td>
<td>Pointer to the class factory interface.</td>
</tr>
</tbody>
</table>

Returns

IDOMJobTkPtr. Returns the cloned JobTicket, with cloned JobTicket content set.

9.36 idommetadata.h File Reference

#include <edl/edltypes.h>
#include <edl/iedlenum.h>
#include <edl/edlproperty.h>

Classes

• 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.

9.37 idomnode.h File Reference

IDOMNode Interface.

#include <edl/iedlobject.h>
#include <edl/iedlenum.h>
#include <edl/edltypes.h>
#include <edl/idomid.h>
#include <edl/edlproperty.h>
#include <edl/edlvector.h>

Generated by Doxygen
Classes

- **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.

Enumerations

- **enum eDOMNodeType**
  DOM node type enumeration.

  ```
  enum eDOMNodeType {
  }
  ```

Functions

- **EDL_API IDOMNodePtr cloneNodeWithChildren (IDOMNodePtr &ptrNode, IDOMNodePtr &ptrParent, IEDLClassFactory *pFactory)**
  Clones a full node tree and (optionally) appends it to the children of a parent node. The clone is made starting with `ptrNode` and recurses down from there, using `pFactory`. If a value is supplied for `ptrParent`, the resulting cloned tree will be appended to that node’s child list.

- **EDL_API IDOMNodePtr deepCloneNodeWithChildren (IDOMNodePtr &ptrNode, IDOMNodePtr &ptrParent, IEDLClassFactory *pFactory, const char **discardProperties)**
  Clones a full node tree and (optionally) appends it to the children of a parent node. The clone is made starting with `ptrNode` and recurses down from there, using `pFactory`. If a value is supplied for `ptrParent`, the resulting cloned tree will be appended to that node's child list. This variant of cloning function also includes cloning of sharing objects such as forms. Also this function allows to discard properties on the cloned nodes.

9.37.1 Detailed Description

`IDOMNode` Interface.

9.37.2 Enumeration Type Documentation

9.37.2.1 eDOMNodeType

**enum eDOMNodeType**

DOM node type enumeration.
9.37 idomnode.h File Reference

### Enumerator

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eDOMContentRootNode</td>
<td>Content Root Node.</td>
</tr>
<tr>
<td>eDOMDocumentSequenceNode</td>
<td>Document Sequence Node.</td>
</tr>
<tr>
<td>eDOMPageNode</td>
<td>Page Node.</td>
</tr>
<tr>
<td>eDOMFixedPageNode</td>
<td>FixedPage Node.</td>
</tr>
<tr>
<td>eDOMGroupNode</td>
<td>Group Node.</td>
</tr>
<tr>
<td>eDOMCharPathGroupNode</td>
<td>Char Path Group Node.</td>
</tr>
<tr>
<td>eDOMTransparencyGroupNode</td>
<td>Transparency GroupNode.</td>
</tr>
<tr>
<td>eDOMCanvasNode</td>
<td>Canvas Node.</td>
</tr>
<tr>
<td>eDOMGlyphsNode</td>
<td>Glyphs Node.</td>
</tr>
<tr>
<td>eDOMGlyphNode</td>
<td>Glyph Node.</td>
</tr>
<tr>
<td>eDOMPathNode</td>
<td>Path Node.</td>
</tr>
<tr>
<td>eDOMRefNode</td>
<td>Ref Node.</td>
</tr>
<tr>
<td>eDOMJobTkContentTypeNode</td>
<td>JobTicket Content Node.</td>
</tr>
<tr>
<td>eDOMJobTkNodeNode</td>
<td>JobTicket Node Node.</td>
</tr>
<tr>
<td>eDOMJobTkValueNode</td>
<td>JobTicket Value Node.</td>
</tr>
<tr>
<td>eDOMJobTkGenericNodeNode</td>
<td>JobTicket Generic Node Node.</td>
</tr>
<tr>
<td>eDOMJobTkGenericCharacterDataNode</td>
<td>JobTicket Generic Character Data Node.</td>
</tr>
<tr>
<td>eDOMVisualRootNode</td>
<td>Visual Root Node.</td>
</tr>
<tr>
<td>eDOMOutlineNode</td>
<td>Outline Node.</td>
</tr>
<tr>
<td>eDOMOutlineEntryNode</td>
<td>Outline Entry Node.</td>
</tr>
<tr>
<td>eDOMAnnotationAppearanceNode</td>
<td>Annotation Appearance Node.</td>
</tr>
<tr>
<td>eDOMFormNode</td>
<td>Form Node.</td>
</tr>
<tr>
<td>eDOMFormInstanceNode</td>
<td>Form Instance Node.</td>
</tr>
<tr>
<td>eDOMFragmentNode</td>
<td>Fragment Node.</td>
</tr>
<tr>
<td>eDOMTileNode</td>
<td>Tile Node.</td>
</tr>
<tr>
<td>eDOMNodeTypeCnt</td>
<td>Node Type Container.</td>
</tr>
</tbody>
</table>

### 9.37.3 Function Documentation

#### 9.37.3.1 cloneNodeWithChildren()

```cpp
EDL_API IDOMNodePtr cloneNodeWithChildren ( 
    IDOMNodePtr & ptrNode,
    IDOMNodePtr & ptrParent,
    IEDLClassFactory * pFactory )
```

Clones a full node tree and (optionally) appends it to the children of a parent node. The clone is made starting with `ptrNode` and recurses down from there, using `pFactory`. If a value is supplied for `ptrParent`, the resulting cloned tree will be appended to that node's child list.

**Parameters**

<table>
<thead>
<tr>
<th><strong>ptrNode</strong></th>
<th>Smart pointer to the source node.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ptrParent</strong></td>
<td>Smart pointer to the parent node where the cloned tree is to be appended(can be NULL).</td>
</tr>
<tr>
<td><strong>pFactory</strong></td>
<td>Pointer to the class factory interface to use when cloning the tree.</td>
</tr>
</tbody>
</table>
Returns

\texttt{IDOMNodePtr} A smart pointer to the cloned node tree.

9.37.3.2 deepCloneNodeWithChildren()

\begin{verbatim}
EDL_API IDOMNodePtr deepCloneNodeWithChildren(
    IDOMNodePtr & ptrNode,
    IDOMNodePtr & ptrParent,
    IEDLClassFactory * pFactory,
    const char ** discardProperties)
\end{verbatim}

Clones a full node tree and (optionally) appends it to the children of a parent node. The clone is made starting with \texttt{ptrNode} and recurses down from there, using \texttt{pFactory}. If a value is supplied for \texttt{ptrParent}, the resulting cloned tree will be appended to that node’s child list. This variant of cloning function also includes cloning of sharing objects such as forms. Also this function allows to discard properties on the cloned nodes.

Parameters

\begin{tabular}{|l|p{0.8\textwidth}|}
\hline
\texttt{ptrNode} & Smart pointer to the source node. \\
\hline
\texttt{ptrParent} & Smart pointer to the parent node where the cloned tree is to be appended(can be NULL). \\
\hline
\texttt{pFactory} & Pointer to the class factory interface to use when cloning the tree. \\
\hline
\texttt{discardProperties} & Array of pointers to property names to discard. Last pointer must be NULL. \\
\hline
\end{tabular}

Returns

\texttt{IDOMNodePtr} A smart pointer to the cloned node tree.

9.38 idomopi.h File Reference

Interface to EDL DOM OPI.

\begin{verbatim}
#include <edl/edltypes.h>
#include <edl/edlstring.h>
#include <edl/iedlobject.h>
#include <edl/ifilespec.h>
\end{verbatim}

Classes

- class \texttt{IDOMOPI}
  
  \textit{Base class representing OPI proxy. Has two descendant interfaces IDOMOPI13 and IDOMOPI20.}
- class \texttt{IDOMOPI13}
  
  \textit{The interface representing OPI proxy with accordance to Open Prepress Interface Specification 1.3.}
- class \texttt{IDOMOPI20}
  
  \textit{The interface representing OPI proxy with accordance to Open Prepress Interface Specification 2.0.}
9.38.1 Detailed Description

Interface to EDL DOM OPI.

9.39 idomoutline.h File Reference

```c
#include <edl/edltypes.h>
#include <edl/iedltree.h>
#include <edl/idomnode.h>
#include <edl/idomtarget.h>
#include <edl/idomcolor.h>
```

Classes

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

- class IDOMOutlineEntry::Data

- class IDOMOutline

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

- class IDOMOutline::Data

9.40 idompage.h File Reference

Page and Fixed page interfaces.

```c
#include <edl/edltypes.h>
#include <edl/edlgeom.h>
#include <edl/idomid.h>
#include <edl/idomnode.h>
#include <edl/idomjobtk.h>
#include <edl/edlfwd.h>
#include <edl/idomresources.h>
#include <edl/idomimageresource.h>
#include <edl/idomgroup.h>
```

Classes

- class IDOMPage

  The base class for DOM page classes such as IDOMFixedPage.

- class IDOMFixedPage

  Represents `<FixedPage>` element.

- class IDOMFixedPage::Data

  Initialization data.
9.40.1 Detailed Description

Page and Fixed page interfaces.

9.41 idompath.h File Reference

Path related interfaces of the DOM.

```c
#include <edl/edltypes.h>
#include <edl/edlgeom.h>
#include <edl/iedlobject.h>
#include <edl/idomnode.h>
#include <edl/idomshape.h>
#include <edl/idombrush.h>
#include <edl/idomtarget.h>
#include <edl/edlblend.h>
#include <edl/idompathgeometry.h>
#include <edl/idomedgemode.h>
```

Classes

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

- class IDOMPathNode::Data
  
  Initialization data.

9.41.1 Detailed Description

Path related interfaces of the DOM.

9.42 idompathgeometry.h File Reference

DOM interfaces for path geometry, figures and segments.

```c
#include <edl/edltypes.h>
#include <edl/edlfwd.h>
#include <edl/edlgeom.h>
#include <edl/iedlobject.h>
#include <edl/edlnamespaces.h>
#include <edl/idomshape.h>
```
Classes

- 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 IDOMArcSegment::Data
  Initialization data.

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

- class IDOMPolyLineSegment::Data
  Initialization data.

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

- class IDOMPolyBezierSegment::Data
  Initialization data.

- class IDOMPolyQuadraticBezierSegment
  Interface to a poly quadratic Bézier segment. A poly quadratic 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).

- class IDOMPolyQuadraticBezierSegment::Data
  Initialization data.

- 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 PathFigures.

- class IDOMPathFigure::Data
  Initialization data.

- class IDOMPathGeometry
  Interface to a path geometry node.

- class IDOMPathGeometry::Data
  Initialization data.

9.42.1 Detailed Description

DOM interfaces for path geometry, figures and segments.

9.43 idomresources.h File Reference

IDOMResources interface.

```c
#include <edl/edltypes.h>
#include <edl/edlstream.h>
#include <edl/idomnode.h>
#include <edl/idomhashable.h>
```
Classes

- 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.

- class IDOMICCProfile
  IDOMICCProfile interface.

- class IDOMICCProfile::Data
  Initialization data.

- class IDOMPrintTicket
  IDOMPrintTicket interface.

- class IDOMPrintTicket::Data
  Initialization data.

- class IDOMAudioFile
  IDOMAudioFile interface.

- class IDOMAudioFile::Data
  Initialization data.

- class IDOMRawDataFile
  IDOMRawDataFile interface.

- class IDOMRawDataFile::Data
  Initialization data.

- 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.

- class IDOMResourceDictionary::Data
  Initialization data.

- class IDOMMatrix
  Defines the render transform matrix.

- class IDOMMatrix::Data
  Initialization data.

9.43.1 Detailed Description

IDOMResources interface.

9.44 idomsecurity.h File Reference

#include <edl/edltypes.h>
#include <edl/edlstring.h>
#include <edl/iedlobject.h>

Classes

- class IDOMSecurityInfo
  Base DOM security class.

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


### Typedefs

- typedef bool(* GetPasswordFunc) (void *pPriv, unsigned char *pBuff)

  
  *Callback typedef for the password getter.*

#### 9.44.1 Typedef Documentation

#### 9.44.1.1 GetPasswordFunc

```c
typedef bool(* GetPasswordFunc) (void *pPriv, unsigned char *pBuff)
```

*Callback typedef for the password getter.*

The callback will be called repeatedly until a valid password is supplied or the callback returns false.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pPriv</code></td>
<td>A private pointer that is passed with each invocation</td>
</tr>
<tr>
<td><code>pBuff</code></td>
<td>The buffer allocated by the library where the password is to be written as a NULL terminated string. The password is expected in the Hex form. The buffer size is 65 bytes - sufficient to store a 32 byte long password in hex form as a null terminated string.</td>
</tr>
</tbody>
</table>

**Returns**

true on success, false if no valid password could be provided.

#### 9.45 idomshape.h File Reference

Classes for handling Shapes.

```c
#include <edl/edlnamespaces.h>
#include <edl/smartptr.h>
#include <edl/edltypes.h>
#include <edl/edlgeom.h>
#include <edl/iedlobject.h>
#include <edl/isession.h>
#include <edl/idombrush.h>
#include <edl/idompathgeometry.h>
```

**Classes**

- class IDOMShape

  *Interface to an IDOMShape.*

- class IDOMShape::Data

  *Initialization data.*

---

*Generated by Doxygen*
9.45.1 Detailed Description

Classes for handling Shapes.

9.46 idomtarget.h File Reference

IDOMTarget Interface.

```c
#include <assert.h>
#include <edl/iedlobject.h>
#include <edl/edltypes.h>
#include <edl/iedlenum.h>
#include <edl/edlstring.h>
#include <edl/idomid.h>
```

Classes

- 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.*

9.46.1 Detailed Description

IDOMTarget Interface.

9.47 iedlcollection.h File Reference

EDL "collection" interface.

```c
#include <edl/ircobject.h>
#include <edl/iedlenum.h>
#include <edl/idomid.h>
#include <edl/smartptr.h>
```
9.48  iedlenum.h File Reference

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.

```c
#include <edl/ircobject.h>
#include <edl/idomid.h>
#include <edl/edlstring.h>
#include <edl/edlgeom.h>
#include <edl/edltypes.h>
```

9.48.1  Detailed Description

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.

9.49  iedlfactory.h File 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.

```c
#include <edl/edltypes.h>
#include <edl/edlstring.h>
#include <edl/iedlobject.h>
#include <edl/ircobject.h>
#include <edl/objclassid.h>
```

Classes

- class IEDLClassFactory
  
  **EDL Class Factory.**

9.49.1  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.
9.50  iedlobject.h File Reference

EDL Object Interface.

#include <edl/edltypes.h>
#include <edl/smartptr.h>
#include <edl/ircobject.h>
#include <edl/objclassid.h>

Classes

• class CClassParams
  
  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.

• 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.

Functions

• template<class Type>
  
  CSmartPtr<Type> clone (const CSmartPtr<Type> & src, IEDLClassFactory * pFactory)
  
  Convenience template to clone a node, returning a reference to the clone using the original type. Throws an IEDLError on failure.

9.50.1  Detailed Description

EDL Object Interface.

9.50.2  Function Documentation

9.50.2.1  clone()

template<class Type>
CSmartPtr<Type> clone ( 
  const CSmartPtr<Type> & src, 
  IEDLClassFactory * pFactory )

Convenience template to clone a node, returning a reference to the clone using the original type. Throws an IEDLError on failure.

Parameters

<table>
<thead>
<tr>
<th>src</th>
<th>Smart pointer to the source node</th>
</tr>
</thead>
<tbody>
<tr>
<td>pFactory</td>
<td>Pointer to the EDL class factory</td>
</tr>
</tbody>
</table>
9.51 iedltempstore.h File Reference

A mechanism for storing and accessing temporary data for use with EDL.

```c
#include <edl/edlstream.h>
```

Classes

- class `IEDLTempStoreObject`
  
  A temporary, file-like object, stored with an `IEDLTempStore`.

- class `IEDLTempStore`
  
  A self-cleaning area for storage of temporary data in the form of streams. One per session, obtainable from an `ISession`.

- class `IEDLTempStore::Data`
  
  Initialization data.

9.51.1 Detailed Description

A mechanism for storing and accessing temporary data for use with EDL.

9.52 iedltree.h File Reference

EDL “tree” interface.

```c
#include <edl/smartptr.h>
#include <edl/iedlobject.h>
#include <edl/edltypes.h>

#include <edl/edlstream.h>
#include <edl/edlvector.h>
```

9.52.1 Detailed Description

EDL “tree” interface.

9.53 ifilespec.h File Reference

Interface to File specification.

```c
#include <edl/edltypes.h>
#include <edl/edlstring.h>
#include <edl/iedlobject.h>
#include <edl/edlstream.h>
#include <edl/edlvector.h>
```
9.53.1 Detailed Description

Interface to File specification.

9.54 iimagecodec.h File Reference

```c
#include <edl/isession.h>
#include <edl/edlstream.h>
#include <edl/edlgeom.h>
#include <edl/iedlobject.h>
#include <edl/idomresources.h>
#include <edl/idomcolorspace.h>
```

Classes

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

- class IImageFrameReader
  
  *IImageFrameReader* reads an image into an imageframe.

- 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 ImageExtraChannel_t
  
  *type used to uniquely identify the type of extra image channel.*

9.55 interactive.h File Reference

JawsMako interactive features.

```c
#include <jawsmako/types.h>
#include <edl/idomform.h>
```
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::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 IFormField and widgets.

- 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. It is intended that future releases of JawsMako will extend this interface.

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

- class JawsMako::ITextMarkupAnnotation
  A generic interface class for 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::IAnnotationUtils::CXMLResource
  Simple class for tracking streams associated with XML generated by `generateXMLForDocument()`
- class JawsMako::INamedDestination
  A named destination in a PDF Document.

Enumerations

- enum JawsMako::eFieldType
  The type of a form field. These map to field types present in the pdf specification.
- enum JawsMako::eAppearanceUsage { JawsMako::eAUNormal, JawsMako::eAURollover, JawsMako::eAUDown }
  The usage scenario for an annotation appearance.

9.55.1 Detailed Description

JawsMako interactive features.

This header contains definitions of interactive objects of JawsMako, such as annotations and forms.

9.56 ircobject.h File Reference

Interface for Reference Counted Object.

```cpp
#include "smartptr.h"
#include "edltypes.h"
```

Classes

- class IRCObject
  Base class Interface for all Reference Counted objects.
9.56.1 Detailed Description

Interface for Reference Counted Object.

In order to allow applications and filters to not have to worry about remembering to release/free/delete EDL allocated objects when they are no longer needed EDL implements so-called "smart pointers" around the EDL objects. As the smart pointers are created/destroyed the underlying object's reference count is automatically incremented/decremented Only when the underlying object's reference count returns to zero that it is actually released/deleted.

9.57 isession.h File Reference

```cpp
#include <edl/iedlobject.h>
#include <edl/iedlfactory.h>
#include <edl/edlfwd.h>
#include <edl/iedltempstore.h>
```

Classes

- class ISession
  
  EDL session class.

- class IRunnable
  
  Interface to filter's runnable classes.

Functions

- EDL_API bool setSessionTempRootDirectory (const EDLSysString &sTempRootDirectory)

  Sets a directory that is used as a root for temporary directories of the EDL sessions. For every new session a unique subfolder under this root directory will be created and used as this session's temp folder.

9.57.1 Function Documentation

9.57.1.1 setSessionTempRootDirectory()

```cpp
EDL_API bool setSessionTempRootDirectory (  
    const EDLSysString & sTempRootDirectory )
```

Sets a directory that is used as a root for temporary directories of the EDL sessions. For every new session a unique subfolder under this root directory will be created and used as this session's temp folder.

Parameters

- `sTempRootDirectory` temp root directory
Returns

bool Returns true on success

9.58  jawsmako.h File Reference

The JawsMako library API.

```c
#include <jawsmako/types.h>
#include <jawsmako/interactive.h>
#include <jawsmako/optionalcontent.h>
```

Classes

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

- **class JawsMako::IOutputAbort**
  A simple class, usable with IOutput and IOutputWriter to signal an abort of output.

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

- **class JawsMako::IOutputWriter**
  A writer for writing individual pages and documents to an output in piecemeal fashion.

- **class JawsMako::IDocumentAssembly**
  A self contained collection of IDocuments.

- **class JawsMako::IDocument**
  A document from an IDocumentAssembly, allowing for high level document and page mangement, and providing on-demand lazy loading of page markup.

- **class JawsMako::IPage**
  A page from an IDocument, allowing high level page management, and providing on-demand access to page contents.

- **class JawsMako::IJawsRenderer**
  A renderer that uses the Jaws RIP to create images from arbitrary DOM.

- **class JawsMako::IJawsRenderer::IHalftone**
  An abstract base class for communicating halftone information to the Jaws renderer, for use with renderMonochrome() and renderMonochromeToFrameBuffer()

- **class JawsMako::IJawsRenderer::CSpotHalftone**
  Description of a simple spot halftone, at 45 degrees, using Jaws's default spot function.

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

- **class JawsMako::IJawsRenderer::CThresholdHalftone**
  A halftone representing a simple threshold.

Enumerations

- **enum JawsMako::eFileFormat**
  JawsMako::eFFPDF, JawsMako::eFFXPS, JawsMako::eFFPS, JawsMako::eFFSVG, JawsMako::eFFPCL5, JawsMako::eFFPCLXL, JawsMako::eFFUnknown
9.58.1 Detailed Description

The JawsMako library API.

9.58.2 Enumeration Type Documentation

9.58.2.1 eFileFormat

```
enum JawsMako::eFileFormat

Document format enumeration

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eFFPDF</td>
<td>PDF</td>
</tr>
<tr>
<td>eFFXPS</td>
<td>XPS</td>
</tr>
<tr>
<td>eFFPS</td>
<td>PostScript</td>
</tr>
<tr>
<td>eFFSVG</td>
<td>SVG</td>
</tr>
<tr>
<td>eFFPCL5</td>
<td>PCL5</td>
</tr>
<tr>
<td>eFPCLXL</td>
<td>PCL/XL</td>
</tr>
<tr>
<td>eFFUnknown</td>
<td>None of the above</td>
</tr>
</tbody>
</table>
```

9.59 memutils.h File Reference

EDL portability wrappers around memset(), memcpy() and memcmp() to allow EDL to use alternate implementations where necessary.

```
#include <string.h>
```

9.59.1 Detailed Description

EDL portability wrappers around memset(), memcpy() and memcmp() to allow EDL to use alternate implementations where necessary.

9.60 objclassid.h File Reference

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.

```
#include "edltypes.h"
#include "edlstring.h"
#include "memutils.h"
```
Classes

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

Functions

- bool equal (const CClassID &id1, const CClassID &id2)
  Compare two CClassID (for the hashtable)
- uint32 hashValue (const CClassID &id)
  Obtain hash value for given CClassID.

9.60.1 Detailed Description

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.

9.60.2 Function Documentation

9.60.2.1 equal()

bool equal {
    const CClassID & id1,
    const CClassID & id2 ) [inline]

Compare two CClassID (for the hashtable)

Parameters

<table>
<thead>
<tr>
<th>id1</th>
<th>First CClassID</th>
</tr>
</thead>
<tbody>
<tr>
<td>id2</td>
<td>Second CClassID</td>
</tr>
</tbody>
</table>

Returns

bool True if equal, false if not

9.60.2.2 hashValue()

uint32 hashValue {
    const CClassID & id ) [inline]

Obtain hash value for given CClassID.
9.61 optionalcontent.h File Reference

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.

```c
#include <jawsmako/types.h>
```

## 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::IOptionalContentConfiguration::COrderEntry`**
  
  Class for presenting the order that groups should be displayed in a user interface. May be arranged in a tree.

- **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.

9.61.1 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.
9.62  outputintent.h File Reference

Declarations of interfaces for querying output intents.

```
#include <jawsmako/types.h>
#include <edl/ifilespec.h>
```

Classes

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

9.62.1 Detailed Description

Declarations of interfaces for querying output intents.

Please see the PDF specification for details.

9.63  pcl5input.h File Reference

PCL5 Input-Specific Features.

```
#include <jawsmako/jawsmako.h>
#include <jawsmako/pjl.h>
```

Classes

- class JawsMako::IPCL5Input
  
  An instance of the JawsMako PCL5 input class.

9.63.1 Detailed Description

PCL5 Input-Specific Features.

This header describes the IPCL5Input interface. These are a subclass of IInput. Instances of this type are returned by IInput::create() when eFFPCL5 is specified. This allows access to PCL5 specific input features and modes of operation.

9.64  pcl5output.h File Reference

JawsMako PCL5 Output.

```
#include <jawsmako/jawsmako.h>
```
Classes

- class JawsMako::IPCL5Output
  Interface for the PCL5 IOutput class.

9.64.1 Detailed Description

JawsMako PCL5 Output.

9.65 pclxlinput.h File Reference

PCL/XL Input-Specific Features.

#include <jawsmako/jawsmako.h>
#include <jawsmako/pjl.h>

Classes

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

9.65.1 Detailed Description

PCL/XL Input-Specific Features.

This header describes the IPCLXLInput interface. These are a subclass of IInput. Instances of this type are returned by IInput::create() when eFFPCLXL is specified. This allows access to PCL/XL specific input features and modes of operation.

9.66 pclxloutput.h File Reference

JawsMako PCLXL Output.

#include <jawsmako/jawsmako.h>

Classes

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

9.66.1 Detailed Description

JawsMako PCLXL Output.
9.67 pdfinput.h File Reference

PDF Input-Specific Features.

#include <jawsmako/jawsmako.h>

Classes

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

- class JawsMako::IPDFInput::CPdfFontInfo
  
  Information about a font in a PDF file, obtained by scanning the PDF font structures.

- class JawsMako::IPDFInput::CPdfScannedInk
  
  Basic information about an ink used in a PDF file, obtained by scanning the PDF page tree.

9.67.1 Detailed Description

PDF Input-Specific Features.

This header describes the IPDFInput interface. These are a subclass of IInput. Instances of this type are returned by IInput::create() when eFFPDF is specified. This allows access to PDF specific input features and modes of operation.

9.68 pdfoutput.h File Reference

JawsMako PDF Output.

#include <jawsmako/jawsmako.h>
#include <jawsmako/transforms.h>

Classes

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

9.68.1 Detailed Description

JawsMako PDF Output.

9.69 pjl.h File Reference

A PJL (Printer Job Language) parser for Mako.

#include <jawsmako/types.h>
#include <jawsmako/jawsmako.h>
Classes

- class JawsMako::IPJLParser
  An instance of the Mako PJL Parser.
- class JawsMako::IPJLParser::CPjLAttributeValue
  A captured PJL attribute.

9.69.1 Detailed Description

A PJL (Printer Job Language) parser for Mako.

This header describes the IPJPParser interface.

9.70 platform.h File Reference

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

9.70.1 Detailed Description

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

9.71 platform_utils.h File Reference

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

#include <stdio.h>
#include <stdarg.h>
#include <edl/edlnamespaces.h>

Functions

- _BEGIN_EDL_NAMESPACE int edlMkdir (const char *dir)
  Creates a subdirectory in an existing directory tree.
- int edlRmdir (const char *dir)
  Removes a subdirectory, if it is empty.
- 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. For example, if TEMP is set to c:\tmp and you invoke edlMakeTempDir("edl", 2) then it will create c:\tmp\edl\2.
- char * edlExclusiveMakeTempDir (const char *subdir, int pidNo)
  As edlMakeTempDir, but fails if the directory exists.
- 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.
• int ediGetProcessId ()
  
  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.

• FILE * ediFopen (const char *filename, const char *mode)
  
  Open a file as per fopen, handling UTF8 file names on all platforms.

• int ediVsnprintf (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 snprintf().

• int ediSnprintf (char *buffer, size_t n, const char *format,...)
  
  Implementation of snprintf that always uses the C locale. Parameters as per snprintf().

• void ediSnprintfE (char *buffer, size_t n, const char *format,...)
  
  As ediSnprintf(), but throws an exception if the operation fails.

9.71.1 Detailed Description

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

9.71.2 Function Documentation

9.71.2.1 ediExclusiveMakeTempDir()

char* ediExclusiveMakeTempDir (  
  const char * subdir,  
  int pidNo )

As ediMakeTempDir, but fails if the directory exists.

Parameters

<table>
<thead>
<tr>
<th>subdir</th>
<th>Specifies the first part of the path to the temporary directory to be created. The path is treated as a UTF8 string.</th>
</tr>
</thead>
<tbody>
<tr>
<td>pidNo</td>
<td>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.</td>
</tr>
</tbody>
</table>

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().

9.71.2.2 ediFopen()

FILE* ediFopen (  
  const char * filename,  
  const char * mode )
Open a file as per fopen, handling UTF8 file names on all platforms.
Parameters

<table>
<thead>
<tr>
<th>filename</th>
<th>The path to the file.</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode</td>
<td>An fopen mode string.</td>
</tr>
</tbody>
</table>

Returns

**FILE** A file pointer, or NULL on failure.

9.71.2.3 edlGetProcessId()

```c
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.

9.71.2.4 edlMakeTempDir()

```c
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. For example, if TMP is set to c:\tmp and you invoke `edlMakeTempDir("edl", 2)` then it will create c:\tmp\edl\2.

Parameters

<table>
<thead>
<tr>
<th>subdir</th>
<th>Specifies the first part of the path to the temporary directory to be created. The path is treated as a UTF8 string.</th>
</tr>
</thead>
<tbody>
<tr>
<td>pidNo</td>
<td>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.</td>
</tr>
</tbody>
</table>

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()`.
9.71.2.5 edlMakeTempDirProvidingSubDirPath()

```c
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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>subdir</code></td>
<td>Specifies the first part of the path to the temporary directory to be created. The path is treated as a UTF8 string.</td>
</tr>
<tr>
<td><code>pidNo</code></td>
<td>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.</td>
</tr>
<tr>
<td><code>fullSubDir</code></td>
<td>The pointer to the address of a UTF8 path to the subdir</td>
</tr>
</tbody>
</table>

**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()`.

9.71.2.6 edlMkdir()

```c
int edlMkdir(
    const char * dir )
```

Creates a subdirectory in an existing directory tree.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>dir</code></td>
<td>Specifies the path to, and name of, the directory 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 bar. The path is treated as a UTF8 string.</td>
</tr>
</tbody>
</table>

**Returns**

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

9.71.2.7 edlRmdir()

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

Removes a subdirectory, if it is empty.
Parameters

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

Returns

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

9.71.2.8 edlSnprintf()

```c
int edlSnprintf (  
        char ∗ buffer,  
        size_t n,  
        const char ∗ format,  
        ... )
```

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

Parameters

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

Returns

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

9.71.2.9 edlSnprintfE()

```c
void edlSnprintfE (  
        char ∗ buffer,  
        size_t n,  
        const char ∗ format,  
        ... )
```

As edlSnprintf(), but throws an exception if the operation fails.

Parameters

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

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

9.71.2.10 edlVsnprintf()

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>buffer</td>
<td>C string to accept the result</td>
</tr>
<tr>
<td>n</td>
<td>Buffer size</td>
</tr>
<tr>
<td>format</td>
<td>Format string</td>
</tr>
<tr>
<td>ap</td>
<td>Variable argument list</td>
</tr>
</tbody>
</table>

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.

9.72 psoutput.h File Reference

JawsMako PS Output.

#include <jawsmako/jawsmako.h>
#include <jawsmako/transforms.h>

Classes

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

9.72.1 Detailed Description

JawsMako PS Output.
9.73  skiarenderer.h File Reference

A Renderer for XPS compatible DOM using Skia.

#include <jawsmako/jawsmako.h>
#include "SkCanvas.h"
#include "SkBitmap.h"

Classes

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

9.73.1  Detailed Description

A Renderer for XPS compatible DOM using Skia.

This header describes the ISkiaRenderer interface.

9.74  smartptr.h File Reference

EDL smart pointers which, in conjunction with the IRCObject class provide reference-counted and automatically garbage-collected IEDLObject which are typically returned by one of the EDL class factories.

#include <stddef.h>
#include <edl/edlnamespaces.h>

9.74.1  Detailed Description

EDL smart pointers which, in conjunction with the IRCObject class provide reference-counted and automatically garbage-collected IEDLObject which are typically returned by one of the EDL class factories.

9.75  structure.h File Reference

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.

#include <jawsmako/types.h>
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()

9.75.1 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.

9.76 svggenerator.h File Reference

A svg generator for JawsMako.

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

Classes

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

9.76.1 Detailed Description

A svg generator for JawsMako.
9.77 text.h File Reference

JawsMako Text Conveniences.

#include <jawsmako/types.h>
#include <jawsmako/interactive.h>
#include <edl/idomglyphs.h>

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.

Typedefs

- typedef CEDLVector<IPageLayoutNodePtr> JawsMako::IPageLayoutNodeCollection

Enumerations

- enum JawsMako::ePageAnalysis {
  JawsMako::ePAAll = -1, JawsMako::ePABuildPage, JawsMako::ePAJoinHRuns, JawsMako::ePAJoinVRuns, JawsMako::ePADefineReadingOrder
}

- enum JawsMako::ePageLayoutType { JawsMako::ePLTRoot, JawsMako::ePLTColumn, JawsMako::ePLTTextRun }

9.77.1 Detailed Description

JawsMako Text Conveniences.

This header contains definitions of interfaces for extracting text information from a page.

9.77.2 Typedef Documentation

Generated by Doxygen
9.77.2.1 IPageLayoutNodeCollection

typedef CEDLVector<IPageLayoutNodePtr> JawsMako::IPageLayoutNodeCollection

A collection of IPageLayoutNodes

9.77.3 Enumeration Type Documentation

9.77.3.1 ePageAnalysis

enum JawsMako::ePageAnalysis

IPageLayout analysis options

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ePAAll</td>
<td>Complete all page analysis steps (build, join and reorder text runs) in one operation</td>
</tr>
<tr>
<td>ePABuildPage</td>
<td>Locate all glyphs nodes on the page</td>
</tr>
<tr>
<td>ePAJoinHRuns</td>
<td>Find all compatible horizontal runs of text</td>
</tr>
<tr>
<td>ePAJoinVRuns</td>
<td>Find all compatible vertical runs of text</td>
</tr>
<tr>
<td>ePADefineReadingOrder</td>
<td>Sort the runs of text into a suitable reading order</td>
</tr>
</tbody>
</table>

9.77.3.2 ePageLayoutType

enum JawsMako::ePageLayoutType

IPageLayoutNode types

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ePLTRoot</td>
<td>Whole page</td>
</tr>
<tr>
<td>ePLTColumn</td>
<td>Column</td>
</tr>
<tr>
<td>ePLTTextRun</td>
<td>Run of text</td>
</tr>
</tbody>
</table>

9.78 transforms.h File Reference

Transforms for JawsMako, allowing direct modification of individual brushes, nodes, or entire pages.

#include <jawsmako/jawsmako.h>
#include <edl/idompath.h>
#include <edl/idomglyphs.h>
#include <edl/idombrush.h>
Classes

- 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, colours, colourspace 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 colour 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.

- 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.

- 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::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 IDOM<--FormInstance 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::IPageCropperTransform
  Very simple transform for cropping pages to one of the standard boxes.

Typedefs

- typedef CEDLVector< ITransformPtr > JawsMako::CTransformVect
  A vector of transform instances.
Enumerations

- 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.*

9.78.1 Detailed Description

Transforms for JawsMako, allowing direct modification of individual brushes, nodes, or entire pages.

9.79 types.h File Reference

Common types and required headers for the JawsMako interface.

```c
#include <edl/edlfwd.h>
#include <edl/edlstring.h>
#include <edl/smartptr.h>
#include <edl/ircobject.h>
#include <edl/iedlfactory.h>
#include <edl/isession.h>
#include <edl/edlerrors.h>
#include <edl/iedlvector.h>
#include <edl/idomid.h>
#include <edl/ifilespec.h>
#include <edl/idomfont.h>
#include <edl/idomform.h>
#include <edl/idomjobtk.h>
#include <edl/idommetadata.h>
#include <edl/idomoutline.h>
#include <edl/idompage.h>
#include <edl/idomcatalog.h>
#include <edl/idomsecurity.h>
#include <edl/idomedgemode.h>
#include <edl/idomresources.h>
```

Classes

- class JawsMako::CTemporaryStoreParameters
  
  *Allows the temporary storage parameters to be optionally overridden.*

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::U16String
  
  *An explicit UTF-16 string, regardless of platform.*

- typedef EDLU16String JawsMako::U32String
  
  *An explicit UTF-32 string, regardless of platform.*
Enumerations

- enum JawsMako::eOptionalContentEvent { JawsMako::eOCEUnknown, JawsMako::eOCEView, JawsMako::eOCEPrint, JawsMako::eOCEExport }
  
  Optional content Event types.
- enum JawsMako::eNormalizationForm { JawsMako::eNFNFKD, JawsMako::eNFNFKC, JawsMako::eNFNFD, JawsMako::eNFNFC }
  
  Normalization forms (for use with NormalizeString )

Functions

- String JawsMako::U8StringToString (const U8String &string)
  Convert UTF8 string to EDLString (wide char)
- U16String JawsMako::U8StringToU16String (const U8String &string)
  Convert UTF8 string to UTF16 string.
- U32String JawsMako::U8StringToU32String (const U8String &string)
  Convert UTF8 string to UTF32 string.
- U8String JawsMako::StringToU8String (const String &string)
  Convert EDLString (wide char) to UTF8 string.
- U16String JawsMako::StringToU16String (const String &string)
  Convert EDLString (wide char) to UTF16 string.
- U32String JawsMako::StringToU32String (const String &string)
  Convert EDLString (wide char) to UTF32 string.
- String JawsMako::NormalizeStringForCompare (const String &string)
  Normalize to Unicode for comparison purposes.
- String JawsMako::NormalizeString (const String &string, eNormalizationForm normalizationForm)
  Normalize to Unicode using the supplied normalization form.
- String JawsMako::ReplaceNonListedLigatures (String string)
  There are chars that Unicode was not sure whether to classify them as ligatures. They were not ligatures in 1.0,
  became ligatures in 1.1 but dropped out in 2.0.
- void JawsMako::RemoveCombinationCharacters (String &string)
  Remove combination characters.
- bool JawsMako::IsCombinationChar (wchar_t c)
  Combination characters are in the range 0300-036F in Unicode table.  https://en.wikipedia.org/wiki/Combining_character#Unicode_ranges.
9.79.1 Detailed Description

Common types and required headers for the JawsMako interface.

Note
Currently, these are simply imported from the EDL namespace. This may change in future releases.

9.79.2 Enumeration Type Documentation

9.79.2.1 eNormalizationForm

enum JawsMako::eNormalizationForm

Normalization forms (for use with NormalizeString )

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eNFNFKD</td>
<td>Compatibility Decomposition.</td>
</tr>
<tr>
<td>eNFNFKC</td>
<td>Compatibility Decomposition, followed by Canonical Composition.</td>
</tr>
<tr>
<td>eNFNFD</td>
<td>Canonical Decomposition.</td>
</tr>
<tr>
<td>eNFNFC</td>
<td>Canonical Decomposition, followed by Canonical Composition.</td>
</tr>
</tbody>
</table>

9.79.2.2 eOptionalContentEvent

enum JawsMako::eOptionalContentEvent

Optional content Event types.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eOCEUnknown</td>
<td>Unknown.</td>
</tr>
<tr>
<td>eOCEView</td>
<td>View.</td>
</tr>
<tr>
<td>eOCEPrint</td>
<td>Print.</td>
</tr>
<tr>
<td>eOCEExport</td>
<td>Export.</td>
</tr>
</tbody>
</table>

9.79.3 Function Documentation

Generated by Doxygen
9.79.3.1  IsCombinationChar()

```cpp
bool JawsMako::IsCombinationChar ( wchar_t c )
```

Combination characters are in the range 0300-036F in Unicode table. [https://en.wikipedia.org/wiki/Combining_character#Unicode_ranges](https://en.wikipedia.org/wiki/Combining_character#Unicode_ranges).

**Parameters**

| c | Character to test |

**Returns**

```cpp
bool True if combination character, otherwise false
```

9.79.3.2  NormalizeString()

```cpp
String JawsMako::NormalizeString ( const String & string, eNormalizationForm normalizationForm )
```

Normalize to Unicode using the supplied normalization form.

**Parameters**

<table>
<thead>
<tr>
<th>string</th>
<th>String to be normalized</th>
</tr>
</thead>
<tbody>
<tr>
<td>normalizationForm</td>
<td>Required normalization form</td>
</tr>
</tbody>
</table>

**Returns**

```cpp
String Normalized result
```

9.79.3.3  NormalizeStringForCompare()

```cpp
String JawsMako::NormalizeStringForCompare ( const String & string )
```

Normalize to Unicode for comparison purposes.

**Parameters**

| string | String to be normalized |

Generated by Doxygen
9.79.3.4 RemoveCombinationCharacters()

```c
void JawsMako::RemoveCombinationCharacters (  
    String & string  
)
```

Remove combination characters.

**Parameters**

<table>
<thead>
<tr>
<th>string</th>
<th>Description for string</th>
</tr>
</thead>
</table>

9.79.3.5 ReplaceNonListedLigatures()

```c
String JawsMako::ReplaceNonListedLigatures (  
    String string  
)
```

There are chars that Unicode was not sure whether to classify them as ligatures. They were not ligatures in 1.0, became ligatures in 1.1 but dropped out in 2.0.

**Parameters**

<table>
<thead>
<tr>
<th>string</th>
<th>String to be processed</th>
</tr>
</thead>
</table>

Returns

**String** Result after processing

9.79.3.6 StringToU16String()

```c
U16String JawsMako::StringToU16String (  
    const String & string  
)
```

Convert EDLString (wide char) to UTF16 string.

**Parameters**

<table>
<thead>
<tr>
<th>string</th>
<th>String to be converted</th>
</tr>
</thead>
</table>
Returns

**U16String** Result of conversion

9.79.3.7 StringToU32String()

```
U32String JawsMako::StringToU32String ( const String & string )
```

Convert EDLString (wide char) to UTF32 string.

**Parameters**

| string | String to be converted |

Returns

**U32String** Result of conversion

9.79.3.8 StringToU8String()

```
U8String JawsMako::StringToU8String ( const String & string )
```

Convert EDLString (wide char) to UTF8 string.

**Parameters**

| string | String to be converted |

Returns

**U8String** Result of conversion

9.79.3.9 U16StringToString()

```
String JawsMako::U16StringToString ( const U16String & string )
```

Convert UTF16 string to EDLString (wide char)
## 9.79.3.10 U16StringToU32String()

```cpp
class JawsMako {
public:
    U32String U16StringToU32String (const U16String & string);
};
```

Convert UTF16 string to UTF32 string.

### Parameters

| string | String to be converted |

### Returns

**U32String** Result of conversion

## 9.79.3.11 U16StringToU8String()

```cpp
class JawsMako {
public:
    U8String U16StringToU8String (const U16String & string);
};
```

Convert UTF16 string to UTF8 string.

### Parameters

| string | String to be converted |

### Returns

**U8String** Result of conversion

## 9.79.3.12 U32StringToString()

```cpp
class JawsMako {
public:
    String U32StringToString (const U32String & string);
};
```

### Parameters

| string | String to be converted |

### Returns

**String** Result of conversion
Convert UTF16 string to EDLString (wide char)
9.79 types.h File Reference

**Parameters**

| string | String to be converted |

**Returns**

*String* Result of conversion

---

### 9.79.3.13 U32StringToU16String()

```cpp
U16String JawsMako::U32StringToU16String (const U32String & string)
```

Convert UTF32 string to UTF16 string.

**Parameters**

| string | String to be converted |

**Returns**

*U16String* Result of conversion

---

### 9.79.3.14 U32StringToU8String()

```cpp
U8String JawsMako::U32StringToU8String (const U32String & string)
```

Convert UTF32 string to UTF8 string.

**Parameters**

| string | String to be converted |

**Returns**

*U8String* Result of conversion

---

### 9.79.3.15 U8StringToString()

```cpp
String JawsMako::U8StringToString (const U8String & string)
```

Generated by Doxygen
Convert UTF8 string to EDLString (wide char)
9.79.3.16 U8StringToU16String()

U16String JawsMako::U8StringToU16String (const U8String & string)

Convert UTF8 string to UTF16 string.

Parameters
- **string**: String to be converted

Returns
- **U16String**: Result of conversion

9.79.3.17 U8StringToU32String()

U32String JawsMako::U8StringToU32String (const U8String & string)

Convert UTF8 string to UTF32 string.

Parameters
- **string**: String to be converted

Returns
- **U32String**: Result of conversion

9.80 xamlgenerator.h File Reference

A xaml generator for JawsMako.
#include <jawsmako/jawsmako.h>
#include <jawsmako/interactive.h>
#include <jawsmako/transforms.h>

Classes

- class JawsMako::IXAMLGenerator
  A XAML generator for JawsMako, allowing simple generation of XAML fragments for individual DOM nodes or entire pages.
- class JawsMako::IXAMLGenerator::CAnnotationXAML
  Class for receiving XAML generated for annotation appearances in a bulk fashion.

9.80.1 Detailed Description

A xaml generator for JawsMako.

9.81 xpsinput.h File Reference

XPS Input-Specific Features.
#include <jawsmako/jawsmako.h>

Classes

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

9.81.1 Detailed Description

XPS Input-Specific Features.

This header describes the IXPSInput interface. These are a subclass of IInput. Instances of this type are returned by IInput::create() when eFFXPS is specified. This allows access to XPS specific input features and modes of operation.

9.82 xpsoutput.h File Reference

JawsMako XPS Output.
#include <jawsmako/jawsmako.h>
#include <jawsmako/transforms.h>

Classes

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

9.82.1 Detailed Description

JawsMako XPS Output.
Index

API, 37
  eAppearanceUsage, 43
add
  IDOMGlyphIDEnumerator, 352
addAction
  IDOMActionArray, 191
addAppearance
  JawsMako::IAnnotation, 145
addAttribute
  IDOMJobTkGenericNode, 453
addChildField
  JawsMako::IFormField, 758
addChildWidget
  JawsMako::IFormField, 758
addFeature
  IDOMJobTkContent, 440
addField
  JawsMako::IForm, 751
addFigure
  IDOMPathGeometry, 546
addGradientStop
  IDOMType3Font, 701
addGroup
  JawsMako::IOptionalContent, 832
addNamedDestination
  JawsMako::IDocument, 181
addNamespace
  IDOMJobTkContent, 440, 441
addParameterInit
  IDOMJobTkContent, 441
addPoint
  IDOMPolyBezierSegment, 592
  IDOMPolyLineSegment, 595
  IDOMPolyQuadraticBezierSegment, 597
addSegment
  IDOMPathFigure, 540
addStrokeDash
  IDOMPathNode, 558
addTag
  IDOMOPI13, 507
addToInitString
  IDOMJobTkContent, 442
addWidget
  JawsMako::IForm, 751
afterBeginPageSetup
  JawsMako::IPSInjector, 919
afterBeginSetup
  JawsMako::IPSInjector, 920
allocateNewDOMid
  idomid.h, 1051
appendCharacterData
  IDOMJobTkGenericCharacterData, 450
appendChild
  IDOMNode, 486
appendDocument
  JawsMako::IDocumentAssembly, 186
appendPage
  JawsMako::IDocument, 181
beforeEndPageSetup
  JawsMako::IPSInjector, 920
beforeEndSetup
  JawsMako::IPSInjector, 920
beforeFirstByte
  JawsMako::IPSInjector, 922
beforePsHeader
  JawsMako::IPSInjector, 922
beforeShowpage
  JawsMako::IPSInjector, 922
beginDocument
  JawsMako::IOutputWriter, 865
beginEnumeration
  IDOMGlyphIDEnumerator, 352
CAnnotationBorder
  JawsMako::CAnnotationBorder, 46
CClassID, 48
  CClassID, 49, 50
equal, 50
CClassParams, 51
CTemporaryStoreParameters
  JawsMako::CTemporaryStoreParameters, 56
CTransformMatrix
  CTransformMatrix, 60, 61
classify, 62
dx, 62
dy, 62
eOperationTypes, 60
equal, 62
iTransform, 63
identity, 63
invert, 63
postMul, 64
preMul, 64
rotate, 64
scale, 65
set, 65
setDX, 66
setDY, 66
setXX, 66
setXY, 66
setYX, 67
setYY, 67
transform, 67
transformRect, 68
translate, 68
xx, 68
xy, 69
yx, 69
yy, 69
CTransformMatrix<TItem>, 58
classID
IDOMArcSegment, 199
IDOMAudioFile, 204
IDOMCanvas, 209
IDOMCatalog, 215
IDOMCharPathGroup, 222
IDOMColor, 228
IDOMColorSpaceDeviceCMYK, 240
IDOMColorSpaceDeviceCMY, 239
IDOMColorSpaceDeviceGray, 242
IDOMColorSpaceDeviceRGB, 249
IDOMColorSpaceDeviceN, 245
IDOMColorSpaceICCBased, 252
IDOMColorSpaceIndexed, 255
IDOMColorSpaceLAB, 258
IDOMColorSpacesGray, 262
IDOMColorSpacesRGB, 264
IDOMColorSpacescRGB, 261
IDOMCompositedImage, 266
IDOMDeviceNColorant, 269
IDOMExponentialFunction, 272
IDOMFilteredImage, 279
IDOMFixedPage, 285
IDOMFont, 295
IDOMFontOpenType, 300
IDOMFontPCL5, 308
IDOMFontPCLXL, 312
IDOMFontSource, 315
IDOMFontSourceFromStream, 318
IDOMFontSourceObfuscationConverter, 321
IDOMForm, 329
IDOMFormInstance, 333
IDOMGlyph, 345
IDOMGlyphDEnumerator, 352
IDOMGlyphs, 358
IDOMGradientStop, 382
IDOMGroup, 385
IDOMGroupingFunction, 390
IDOMICCProfile, 396
IDOMImageBitScalerFilter, 401
IDOMImageBleederFilter, 402
IDOMImageBrush, 405
IDOMImageChannelSelectorFilter, 414
IDOMImageColorConverterFilter, 415
IDOMImageColorKeyFilter, 417
IDOMImageColorSpaceSubstitutionFilter, 418
IDOMImageDecodeFilter, 419
IDOMImageDeindexerFilter, 421
IDOMImageDeviceNTobaseFilter, 422
IDOMImageDownsamplerFilter, 423
IDOMImageInverterFilter, 425
IDOMImageMaskExpanderFilter, 426
IDOMImageProperties, 428
IDOMJPEGImage, 466
IDOMJobTk, 435
IDOMJobTkContent, 442
IDOMJobTkGenericCharacterData, 450
IDOMJobTkGenericNode, 453
IDOMJobTkNode, 456
IDOMJobTkOwner, 462
IDOMJobTkValue, 464
IDOMLinearGradientBrush, 469
IDOMMaskedBrush, 473
IDOMMatrix, 476
IDOMNullBrush, 502
IDOMPCImage, 583
IDOMPDFImage, 585
IDOMPNGImage, 589
IDOMPage, 529
IDOMPathFigure, 540
IDOMPathGeometry, 546
IDOMPathNode, 558
IDOMPolyBezierSegment, 593
IDOMPolyLineSegment, 595
IDOMPolyQuadraticBezierSegment, 598
IDOMPostScriptCalculatorFunction, 600
IDOMPrintTicket, 603
IDOMRadialGradientBrush, 605
IDOMRawDataFile, 610
IDOMRawImage, 612
IDOMRecombineImage, 615
IDOMResourceDictionary, 621
IDOMSampledFunction, 623
IDOMShadingPatternType1Brush, 635
IDOMShadingPatternType2Brush, 639
IDOMShadingPatternType3Brush, 644
IDOMShadingPatternType4567Brush, 650
IDOMShape, 659
IDOMSoftMaskBrush, 665
IDOMSolidColorBrush, 668
IDOMStitchingFunction, 673
IDOMTIFFImage, 679
IDOMTilingPatternBrush, 683
IDOMTransparencyGroup, 694
IDOMType3Font, 701
IDOMVisualBrush, 705
IDOMVisualRoot, 713
IDOMWMPImage, 714
IEDLFontSystemFont, 720
IEDLNamespace, 725
<table>
<thead>
<tr>
<th>Index Item</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEDLTempStore</td>
<td>733</td>
</tr>
<tr>
<td>IFontHeaderWriteSegmentBlockEnumerator</td>
<td>744</td>
</tr>
<tr>
<td>IFontPCL5WriteSegmentBlockEnumerator</td>
<td>747</td>
</tr>
<tr>
<td>classify</td>
<td></td>
</tr>
<tr>
<td>CTransformMatrix</td>
<td>62</td>
</tr>
<tr>
<td>clone</td>
<td></td>
</tr>
<tr>
<td>IEDLObject</td>
<td>728</td>
</tr>
<tr>
<td>iedlobject.h</td>
<td>1068</td>
</tr>
<tr>
<td>JawsMako::IAntenna</td>
<td>146</td>
</tr>
<tr>
<td>JawsMako::IAntennaAppearance</td>
<td>153</td>
</tr>
<tr>
<td>JawsMako::IDocument</td>
<td>181</td>
</tr>
<tr>
<td>JawsMako::IDocumentAssembly</td>
<td>187</td>
</tr>
<tr>
<td>JawsMako::IForm</td>
<td>751</td>
</tr>
<tr>
<td>JawsMako::IFormField</td>
<td>758</td>
</tr>
<tr>
<td>JawsMako::INamedDestination</td>
<td>829</td>
</tr>
<tr>
<td>JawsMako::IPage</td>
<td>869</td>
</tr>
<tr>
<td>cloneJobTkContentTree</td>
<td></td>
</tr>
<tr>
<td>idomjobtk.h</td>
<td>1056</td>
</tr>
<tr>
<td>cloneJobTkTree</td>
<td></td>
</tr>
<tr>
<td>idomjobtk.h</td>
<td>1057</td>
</tr>
<tr>
<td>cloneNode</td>
<td></td>
</tr>
<tr>
<td>IDOMNode</td>
<td>486</td>
</tr>
<tr>
<td>cloneNodeWithChildren</td>
<td></td>
</tr>
<tr>
<td>idomnode.h</td>
<td>1059</td>
</tr>
<tr>
<td>cloneTree</td>
<td></td>
</tr>
<tr>
<td>IDOMNode</td>
<td>488</td>
</tr>
<tr>
<td>cloneTreeAndAppend</td>
<td></td>
</tr>
<tr>
<td>IDOMNode</td>
<td>488</td>
</tr>
<tr>
<td>compare</td>
<td></td>
</tr>
<tr>
<td>IEDLTime</td>
<td>738</td>
</tr>
<tr>
<td>completeRead</td>
<td></td>
</tr>
<tr>
<td>IInputStream</td>
<td>797</td>
</tr>
<tr>
<td>completeWrite</td>
<td></td>
</tr>
<tr>
<td>IOutputStream</td>
<td>860</td>
</tr>
<tr>
<td>completelyContainsShape</td>
<td></td>
</tr>
<tr>
<td>IDOMShape</td>
<td>659</td>
</tr>
<tr>
<td>convertColors</td>
<td></td>
</tr>
<tr>
<td>IColorManager</td>
<td>166, 167</td>
</tr>
<tr>
<td>convertToCubicBezierSegment</td>
<td></td>
</tr>
<tr>
<td>IDOMPolyQuadraticBezierSegment</td>
<td>598</td>
</tr>
<tr>
<td>convertToQName</td>
<td></td>
</tr>
<tr>
<td>IDOMJobTkContent</td>
<td>442</td>
</tr>
<tr>
<td>convertToSimpleSegment</td>
<td></td>
</tr>
<tr>
<td>IDOMArcSegment</td>
<td>199</td>
</tr>
<tr>
<td>copy</td>
<td></td>
</tr>
<tr>
<td>IOutputStream</td>
<td>860</td>
</tr>
<tr>
<td>copyNodeData</td>
<td></td>
</tr>
<tr>
<td>IDOMNode</td>
<td>488</td>
</tr>
<tr>
<td>create</td>
<td></td>
</tr>
<tr>
<td>IDOMColor</td>
<td>228</td>
</tr>
<tr>
<td>IDOMColorSpaceDeviceCMYK</td>
<td>241</td>
</tr>
<tr>
<td>IDOMColorSpaceDeviceCMY</td>
<td>239</td>
</tr>
<tr>
<td>IDOMColorSpaceDeviceGray</td>
<td>242</td>
</tr>
<tr>
<td>IDOMColorSpaceDeviceRGB</td>
<td>250</td>
</tr>
<tr>
<td>IDOMColorSpaceDeviceN</td>
<td>245, 246</td>
</tr>
<tr>
<td>IDOMColorSpaceICCBased</td>
<td>252</td>
</tr>
<tr>
<td>IDOMColorSpaceIndexed</td>
<td>255, 256</td>
</tr>
<tr>
<td>IDOMColorSpaceLAB</td>
<td>258</td>
</tr>
</tbody>
</table>

**Note:**
- The text is extracted from a documentation index, indicating the page numbers where each item is referenced.
- The items listed are likely identifiers or classes associated with a specific software or documentation context.

*Generated by Doxygen*
INDEX

JawsMako::IPage, 869
JawsMako::IPageCropperTransform, 872
JawsMako::IPageLayout, 874
JawsMako::IPageLayoutData, 875
JawsMako::IPageLayoutNode, 877
JawsMako::IPopupAnnotation, 917
JawsMako::IPopupAnnotation, 936
JawsMako::IRedactionAnnotation, 942
JawsMako::IRedactorTransform, 939
JawsMako::IRendererTransform, 942
JawsMako::ISVGGenerator, 970
JawsMako::ISkiaRenderer, 952
JawsMako::IXAMLGenerator, 999
JawsMako::IXPSInput, 1007
createAnnotationReferenceFromTag
JawsMako::IAnnotationUtils, 156
createCalibratedGrayProfile
IColorManager, 168
createCalibratedRGBProfile
IColorManager, 168
createCompositeStream
IInputStream, 797
createFilled
IDOMPathNode, 558
createFromArray
IDOMColor, 229
createFromFile
IInputStream, 797, 798
createFromFileShared
IInputStream, 798, 799
createFromFlateCompressed
IInputStream, 799
createFromLz4Compressed
IInputStream, 800
createFromMemory
IInputStream, 800
createFromFileWithContents
IInputStream, 801
createFromRAUserFunc
IInputStream, 801
createFromUserFunc
IInputStream, 802
IOutputStream, 861
createFromVect
IDOMColor, 229
createGrayProfile
IColorManager, 169
createImage
IDOMImage, 398
createImageDecoder
IDOMImage, 398
createImageEncoder
IDOMImage, 398
createlInstance
IEDLClassFactory, 716
createMarkedContentReferencePair
JawsMako::IStructureElement, 963
createNewDOMid
IDOMCatalog, 215, 216
createNode
IDOMOutlineEntry, 522
createPushbackStream
IInputStream, 802
createReader
IEDLTempStoreObject, 736
createRenamedFont
IDOMFontOpenType, 300
createShading
IDOMLinearGradientBrush, 469
IDOMRadialGradientBrush, 606
createStroked
IDOMPathNode, 559
createSubFile
IInputStream, 803
createSubsetFont
IDOMFontOpenType, 301
createTemporaryObject
IEDLTempStore, 733
createTemporaryObjectWithContents
IEDLTempStore, 733
createTemporaryReaderWriter
IEDLTempStore, 734
createTemporaryReaderWriterPair
IEDLTempStore, 734
createTemporaryStreamWithContents
IEDLTempStore, 734
createToFile
IOutputStream, 861, 862
createToFlateCompressed
IOutputStream, 862
createToLz4Compressed
IOutputStream, 863
createTrueTypeOnlyFontVersion
IDOMFontOpenType, 301
createWriter
IEDLTempStoreObject, 736
createWriterAndImage
IDOMRawImage, 612
DOMNodeFlags
IDOMNodeFlags, 500
dash
JawsMako::CAnnotationBorder, 46
decRef
IRCOObject, 934
deepCloneNodeWithChildren
idomnode.h, 1060
deleteGlyphs
IDOMType3Font, 701
deleteTag
IDOMOPI13, 507

Generated by Doxygen
determineUri
IDOMFontSource, 315, 316
IDOMFontSourceFromStream, 318
IDOMFontSourceObfuscationConverter, 322
IDOMFontSourceStreamFilter, 325
difference
IDOMShape, 660
drawNode
JawsMako::ISkiaRenderer, 953
dx
CTransformMatrix, 62
dy
CTransformMatrix, 62
eAnnotationType
JawsMako::IAnnotation, 144
eAppearanceUsage
API, 43
eBlackPointCompensation
edlblackpointcompensation.h, 1019
eBorderType
JawsMako::CAnnotationBorder, 46
eColorInterpolationMode
idombrush.h, 1041
eColorSpaceType
IDOMColorSpace, 235
EDLIFStream, 136
EDLQFStream, 137
EDLQName, 137
EDLQName, 138
getName, 138
getNameWithPrefix, 139
getNameNamespace, 139
getSysName, 139
operator==, 140
setName, 140
setNamespace, 141
EDLStringToEDLSysString
edlstring.h, 1036
EDLSysStringToEDLString
edlstring.h, 1036
eDOMJobTkLevel
idomjobtk.h, 1056
eDOMNodeType
idomnode.h, 1058
eEdgeMode
idomedgegmode.h, 1046
eEncodeFormat
JawsMako::IImageEncoderTransform, 773
eFileFormat
jawsmako.h, 1075
eFillRule
IDOMPathGeometry, 546
eFontSourceType
IDOMFontSource, 315
eFontStreamFilterType
IDOMFontSourceStreamFilter, 325
eFunctionType
IDOMFunction, 340
eGlyphIDSpecial
IDOMGlyph, 345
eNormalizationForm
types.h, 1095
eOperation
IDOMFontOpenType, 299
eOperationTypes
CTransformMatrix, 60
eOptionalContentEvent
types.h, 1095
eOriginalFontType
IDOMFontOpenType, 299
ePCLXLFontType
IDOMFontPCLXL, 312
ePDFVersion
JawsMako::IPDFOutput, 898
ePageAnalysis
text.h, 1091
ePageLayoutType
text.h, 1091
ePdfXDeviceNHandling
JawsMako::IPDFOutput, 899
ePdfXExtendedGraphicsStateHandling
JawsMako::IPDFOutput, 899
ePdfXOptionalContentHandling
JawsMako::IPDFOutput, 899
ePermissionsFlags
IDOMStandardPDFSecurityInfo, 671
eRenderingIntent
edlrenderingintent.h, 1031
eSpreadMethod
idombrush.h, 1042
eStrokeLineCap
IDOMPathNode, 556
eStrokeLineJoin
IDOMPathNode, 556
eStrokeMiterLimitTreatment
IDOMPathNode, 556
eTIFFCompression
IDOMTIFFImage, 679
eTargetType
IDOMTarget, 676
eTextStyle
IDOMOutlineEntry, 521
eTilingMode
idomoutlineEntry, 521
idombrush.h, 1042
eType
IDOMMetadata, 481
eViewUnits
idombrush.h, 1042
eVisibilityExpressionOperation
JawsMako::IOptionalContentVisibilityExpression, 849
eVisibilityPolicy
JawsMako::IOptionalContentDetails, 838
getBrushType
getIDOMBrush, 206
getCMYKSWOPProfile
getIDOMColorManager, 170
getCalculator
getIDOMPostScriptCalculatorFunction, 601
getCalculatorAsPostScriptStream
getIDOMPostScriptCalculatorFunction, 602
getCalloutLine
JawsMako::IFreeTextAnnotation, 765
getCaptionOffset
JawsMako::ILineAnnotation, 818
getCaretStops
getIDOMGlyphs, 360
getAddress
getIDOMRadialGradientBrush, 606
getCharPathType
getIDOMCharPathGroup, 223
getCharacterData
getIDOMJobTkGenericCharacterData, 450
getCheckSum
JawsMako::IOutputIntent, 858
gerChildFields
JawsMako::IField, 759
gerChildValue
getIDOMJobTkNode, 458
gerChildWidgets
JawsMako::IField, 759
getClassID
IEDLObject, 728
gerClip
getIDOMGlyphs, 360
getIDOMGroup, 386
getIDOMPathNode, 561
gerClippedGroup
getIDOMCharPathGroup, 223
gerColor
getIDOMGradientStop, 382
getIDOMSolidColorBrush, 669
JawsMako::IAnnotation, 147
gerColorInterpolationMode
getIDOMGradientBrush, 379
gerColorKey
getIDOMPDFImage, 586
gerColorName
getIDOMOPI13, 507
gerColorSpace
getIDOMColor, 230
getIDOMPDFImage, 586
getIDOMShadingPatternBrush, 630
getIDOMTransparencyGroup, 695
getImageFrame, 778
gerColorSpaceType
getIDOMColorSpace, 236
gerColorType
getIDOMOPI13, 508
gerColorValue
getIDOMOPI13, 508
gerColorant
getIDOMColorSpaceDeviceN, 247
gerColored
getIDOMGlyph, 347
gerCombinedContent
getIDOMJobTk, 436
gerComments
getIDOMOPI13, 508
gerComponentRange
getIDOMColorSpace, 236
gerComponentValue
getIDOMColor, 231
gerComponentsHaveSameRange
getIDOMColorSpace, 237
gerContent
getIDOMJobTk, 436
JawsMako::IPage, 870
gerContentBox
getIDOMFixedPage, 286
gerContents
JawsMako::IAnnotation, 147
gerCoords
getIDOMShadingPatternType2Brush, 639
getIDOMShadingPatternType3Brush, 645
gerCount
getIDOMCatalog, 216
gerCreationTime
JawsMako::IMarkupAnnotation, 826
gerCropBox
getIDOMFixedPage, 287
gerCropFixed
getIDOMOPI13, 509
gerCropRect
getIDOMOPI13, 509
gerDOMid
getIDOMNode, 492
gerDataSource
getIDOMShadingPatternType4567Brush, 652
gerDay
IEDLTime, 739
gerDecode
getIDOMPDFImage, 586
getIDOMShadingPatternType4567Brush, 652
gerDecodeParameters
getIDOMPDFImage, 587
gerDecodeSize
getIDOMShadingPatternType4567Brush, 652
gerDefaultIntentForICCBasedSpace
getIDOMColorManager, 170
gerDefaultRenderingIntent
getIDOMColorSpace, 237
gerDescription
getIDOMOutlineEntry, 522
gerDeviceCMYKIntercept
getIDOMColorManager, 170
gerDeviceFontName
getIDOMGlyphs, 361
getDeviceGrayIntercept
IColorManager, 170
getDeviceRGBIntercept
IColorManager, 171
getDocument
JawsMako::IDocumentAssembly, 187
getDomain
IDOMShadingPatternType1Brush, 635
IDOMShadingPatternType2Brush, 640
IDOMShadingPatternType3Brush, 645
getDotGainFunction
IDOMDeviceNColorant, 269
getEDLErrorString
edlerrors.cpp, 1020
edlerrors.h, 1026
getEdgeMode
IDOMCanvas, 210
IDOMPathNode, 562
getEmbedded
IDOMFontOpenType, 302
getEmbeddedPSCSAForICCBasedSpace
IColorManager, 171
getEncodeVector
IDOMStitchingFunction, 674
gGetEndPoint
IDOMLinearGradientBrush, 470
IDOMPathSegment, 581
getEnumerationCount
IFontPCL5WriteSegmentBlockEnumerator, 748
getEnumerationItemBlockSize
IFontPCL5WriteSegmentBlockEnumerator, 748
ggetEquivalentPath
IDOMGlyphs, 361
ggetEquivalentSimpleBrush
IDOMShadingPatternType2Brush, 640
IDOMShadingPatternType3Brush, 645
ggetEquivalentSimpleVisualBrush
IDOMVisualBrush, 706
ggetEquivalentTilingBrush
IDOMImageBrush, 405
IDOMVisualBrush, 707
ggetEquivalentVisualBrush
IDOMImageBrush, 407
IDOMTilingPatternBrush, 684
ggetEquivalentXPSBrush
IDOMMaskedBrush, 474
ggetExpanded
IDOMOutlineEntry, 523
ggetExponent
IDOMExponentialFunction, 273
ggetExtend
IDOMShadingPatternType2Brush, 641
IDOMShadingPatternType3Brush, 646
ggetExtraChannelType
IImageFrame, 779
ggetFactory
IRegistration, 947
ggetFieldFlags
JawsMako::IWidgetAnnotation, 994
ggetFieldId
JawsMako::IFormField, 759
ggetFieldType
JawsMako::IFormField, 760
JawsMako::IWidgetAnnotation, 996
ggetFields
JawsMako::IForm, 752
ggetFigureCollectionEnum
IDOMPathGeometry, 548
ggetFiguresCount
IDOMPathGeometry, 548
ggetFile
IDOMActionLaunch, 193
ggetFill
IDOMCharPathGroup, 223
IDOMGlyphs, 361
IDOMPathNode, 562
ggetFillRule
IDOMPathGeometry, 548
ggetFillerAtIndex
IDOMFilteredImage, 280
ggetFirstChild
IDOMNode, 492
ggetFirstImageFrame
IDOMImage, 399
ggetFitType
IDOMPageRectTarget, 532
ggetFlags
IDOMNode, 492
JawsMako::IAnnotation, 147
ggetFlattenedGeometry
IDOMPathGeometry, 548
ggetFlattenedGlyphInfo
IDOMGlyphs, 362
ggetFont
IDOMGlyphs, 362
ggetFontBaseStream
IDOMFont, 295
ggetFontDecorationFlags
IEDLFontSystemFont, 721
ggetFontFamily
IEDLFontSystemFont, 721
ggetFontFilepath
IEDLFontSystemFont, 721
ggetFontHeaderSegmentBlockEnumerator
IDOMFontPCLXL, 312
ggetFontIndex
IDOMGlyphs, 362
ggetFontLicenseFromOS2Table
IDOMFontOpenType, 302
ggetFontName
IDOMFontPCL5, 308
IDOMFontPCLXL, 313
IEDLFontSystemFont, 722
ggetFontOpenTypeTableAccessor
IDOMFontOpenType, 302
ggetFontRenderingEmSize

Generated by Doxygen
getFontSource
  IDOMFont, 296
getFontSourceType
  IDOMFontSource, 316
g.getFontStreamFilterType
  IDOMFontSourceStreamFilter, 326
g.getFontTrueTypeGlyphAccessor
  IDOMFontOpenType, 303
  IDOMFontPCL5, 309
  IDOMFontPCLXL, 313
g.getFontType
  IDOMFont, 296
g.getFontWeight
  IEDLFontSystemFont, 722
gForm
  IDOMFormInstance, 334
    JawsMako::IAnnotationAppearance, 153
    JawsMako::IDocument, 182
gFormMid
  IDOMForm, 330
gFormat15FontBlockEnumerator
  IDOMFontPCL5, 309
gFormat16FontBlockEnumerator
  IDOMFontPCL5, 309
gGetFullName
  IDOMFontOpenType, 303
gGetFunction
  IDOMShadingPatternType1Brush, 635
  IDOMShadingPatternType2Brush, 641
  IDOMShadingPatternType3Brush, 646
  IDOMShadingPatternType4567Brush, 653
gGetFunctionAtIndex
  IDOMGroupingFunction, 391
  IDOMStitchingFunction, 674
gGetFunctionType
  IDOMFunction, 341
gGetGlyph
  IDOMType3Font, 702, 703
  IDOMGlyph, 347
  IDOMGlyphIDEnumerator, 353
gGetGlyphInfoCollectionEnum
  IDOMGlyphs, 363
gGetGlyphName
  IDOMGlyph, 347, 348
gGetGlyphUnicode
  IDOMGlyph, 348
gGetGradientOrigin
  IDOMRadialGradientBrush, 606
gGetGradientStopCollectionEnum
  IDOMGradientBrush, 379
gGetGradientStopsCount
  JawsMako::ILineAnnotation, 818
  JawsMako::IShapeAnnotation, 951
gGetGrayMap
  IDOMOPI13, 510
  IDOMOPI13, 510
gGetGrayMapSize
  IDOMOPI13, 510
gGetGroup
  IDOMSoftMaskBrush, 666
gGetHasAlphaChannel
    IImageFrame, 779
gGetHasCustomAdvance
    IDOMGlyph, 348
gGetHasICCProfile
    IImageFrame, 779
gGetHeight
    IDOMFixedPage, 287
    IImageFrame, 780
gGetICCProfile
    IDOMColorSpaceICCBased, 253
    IDOMImageBrush, 407
    IImageFrame, 780
gGetIconName
  JawsMako::ITextAnnotation, 974
gGetID
  IDOMOPI13, 510
gGetId
  IDOMType3Font, 703
gGetIdByIndex
  IDOMCatalog, 216
gGetIdByNumbers
  IDOMCatalog, 217
gGetIdByURI
  IDOMCatalog, 217
gGetImageProperties
  IDOMImage, 399
gGetImageSource
  IDOMImageBrush, 407
gGetImageType
  IDOMImage, 400
  IDOMOPI13, 511
gGetIndices
  IDOMGlyphs, 363
gGetInkList
  JawsMako::IInkAnnotation, 790
gGetInputDomain
  IDOMFunction, 341
gGetInputEncode
  IDOMSampledFunction, 625
gGetInputFontSource
  IDOMFontSourceObfuscationConverter, 322
  IDOMFontSourceStreamFilter, 326
gGetIntProperty
  IDOMImageProperties, 429
gGetIntent
  JawsMako::IStampAnnotation, 956
gGetInputDomain
  IDOMFunction, 341
gGetInputEncode
  IDOMSampledFunction, 625
gGetInputFontSource
  IDOMFontSourceObfuscationConverter, 322
  IDOMFontSourceStreamFilter, 326
gGetIntProperty
  IDOMImageProperties, 429
gGetIntent
  JawsMako::IStampAnnotation, 956
  JawsMako::IShapeAnnotation, 951
gGetInteriorColor
  JawsMako::ILineAnnotation, 818
  JawsMako::IRedactionAnnotation, 936
  JawsMako::IShapeAnnotation, 951
gGetInterpolationMethod
  IDOMSampledFunction, 625
gGetIsCharPath
  IDOMOPI13, 510
  IDOMOPI13, 510

Generated by Doxygen
getPoints
    JawsMako::IPolyAnnotation, 915
getPointsCount
    IDOMPolyBezierSegment, 593
    IDOMPolyLineSegment, 595
    IDOMPolyQuadraticBezierSegment, 599
getPopupReference
    JawsMako::IMarkupAnnotation, 826
getPos
    IEDLStream, 730
getPosition
    IDOMOPI13, 511
getPostScriptCSAForICCBasedSpace
    IColorManager, 172
getPostScriptName
    IDOMFontOpenType, 305
getPrefix
    IEDLNamespace, 725
getPreviousChild
    IDOMNode, 494
getPreviousSibling
    IDOMNode, 494
getPrintingOrder
    IDOMColorSpaceDeviceN, 247
getProcessColorSpace
    IDOMColorSpaceDeviceN, 247
getProcessComponentNames
    IDOMColorSpaceDeviceN, 248
getProfileFileSpecifications
    JawsMako::IOutputIntent, 858
getProfileForSpace
    IColorManager, 173
getProfileNameForICCBasedSpace
    IColorManager, 173
getProfileVersion
    IDOMICCProfile, 396
getProfileVersionForICCBasedSpace
    IColorManager, 173
getProperty
    IDOMImageProperties, 429
    IDOMMetadata, 482
    IDOMNode, 494
getPropertyCollectionEnum
    IDOMMetadata, 482
    IDOMNode, 495
getQName
    IDOMJobTkGenericNode, 454
    IDOMJobTkNode, 459
getQNameAsString
    IDOMJobTkNode, 459
getQuadPoints
    JawsMako::ILinkAnnotation, 821
    JawsMako::IRedactionAnnotation, 936
    JawsMako::ITextMarkupAnnotation, 976
geradiusX
    IDOMArcSegment, 200
    IDOMRadialGradientBrush, 607
geradiusY
    IDOMArcSegment, 200
    IDOMRadialGradientBrush, 607
getRangeAB
    IDOMColorSpaceLAB, 259
getRawBytesPerRow
    IImageFrame, 781
    IImageFrameWriter, 786
getRect
    JawsMako::IAnnotation, 148
getRectInset
    JawsMako::ICaretAnnotation, 158
    JawsMako::IFreeTextAnnotation, 765
    JawsMako::IShapeAnnotation, 951
getReference
    JawsMako::IAnnotation, 148
getRenderTransform
    IDOMFormInstance, 335
    IDOMGlyphs, 366
    IDOMGroup, 387
    IDOMMatrix, 477
    IDOMPathGeometry, 549
    IDOMPathNode, 565
    IDOMTilingPatternBrush, 685
    IDOMTransformableBrush, 691
getRequestedFontName
    IDOMFontOpenType, 305
getResolution
    IDOMOPI13, 512
    IDOMShape, 661
getResource
    JawsMako::ISVGGenerator, 971
    JawsMako::IXAMLGenerator, 1002
getResourceDictionary
    IDOMCanvas, 211
    IDOMFixedPage, 288
getResources
    JawsMako::ISVGGenerator, 971
    JawsMako::IXAMLGenerator, 1002
getRight
    IDOMPageRectTarget, 533
getRootNode
    IDOMJobTkContent, 445
gerotationAngle
    IDOMArcSegment, 200
generatedAppearance
    JawsMako::IAnnotationAppearance, 153
getSecurityInfo
    JawsMako::IDocumentAssembly, 188
gersegmentBlockItem
    IFontHeaderWriteSegmentBlockEnumerator, 744
gersegmentBlockSize
    IFontHeaderWriteSegmentBlockEnumerator, 745
gersegmentCollectionEnum
    IDOMPathFigure, 542
gersegmentItem
    IFontHeaderWriteSegmentBlockEnumerator, 745
gersegmentCount
    IDOMPathFigure, 542
getShadingType
  IDOMShadingPatternBrush, 631
getShape
  IDOMPathGeometry, 549
  IDOMPathNode, 565
getShouldZeroWidthLinesBeVisible
  IDOMPathNode, 566
getSimplifiedGeometry
  IDOMPathGeometry, 550
getSimplifiedGradient
  IDOMRadialGradientBrush, 607
getSingleton
  IEDLClassFactory, 716
getSize
  IDOMOPI13, 512
getSnapsToDevicePixels
  IDOMPathNode, 566
getSoftMaskType
  IDOMSoftMaskBrush, 666
getSolidity
  IDOMDeviceNColorant, 270
getSoundAsWav
  JawsMako::ISoundAnnotation, 954
getSourceImage
  IDOMFilteredImage, 281
getSpreadMethod
  IDOMGradientBrush, 379
getStartPoint
  IDOMLinearGradientBrush, 470
  IDOMPathFigure, 542
getStartupDirectory
  ISession, 947
g getState
  JawsMako::IAnnotation, 148
  JawsMako::IAnnotationAppearance, 154
g getStream
  IDOMCompositeImage, 267
  IDOMFilteredImage, 281
  IDOMFontSourceFromStream, 319
  IDOMFontSourceObfuscationConverter, 323
  IDOMFontSourceStreamFilter, 326
  IDOMRecombineImage, 615
  IDOMResource, 617
g getStreamLength
  IDOMFontSourceFromStream, 319
  IDOMResource, 618
g getStroke
  IDOMPathNode, 566
g getStrokeDashCollectionEnum
  IDOMPathNode, 566
g getStrokeDashLineCap
  IDOMPathNode, 567
g getStrokeDashOffset
  IDOMPathNode, 567
g getStrokeDashsCount
  IDOMPathNode, 567
g getStrokeEndLineCap
  IDOMPathNode, 568
g getStrokeLineJoin
  IDOMPathNode, 568
g getStrokeMiterLimit
  IDOMPathNode, 568
g getStrokeMiterLimitTreatment
  IDOMPathNode, 569
g getStrokePath
  IDOMCharPathGroup, 224
g getStrokeStartLineCap
  IDOMPathNode, 569
g getStrokeThickness
  IDOMPathNode, 569
g getStructureElement
  IDOMOutlineEntry, 523
g getStructureElementReference
  IDOMForm, 330
g getStyleSimulations
  IDOMGlyphs, 367
g getSweepDirection
  IDOMArcSegment, 201
g getSymbolSetIDCode
  IDOMFontPCL5, 310
g getSymbolSetID
  IDOMFontPCL5, 310
g getSynthetic
  IDOMRawImage, 613
g getSysName
  EDLQName, 139
g getTableData
  IDOMSampledFunction, 627
g getTableDimension
  IDOMSampledFunction, 627
g getTagText
  IDOMOPI13, 513
g getTags
  IDOMOPI13, 513
g getTagsSize
  IDOMOPI13, 513
g getTarget
  IDOMOutlineEntry, 524
  JawsMako::INamedDestination, 830
g getTargetId
  IDOMInternalTarget, 432
g getTargetPage
  IDOMPageTarget, 432
g getTargetType
  IDOMActionArray, 192
  IDOMActionLaunch, 194
  IDOMExternalTarget, 276
  IDOMInternalTarget, 433
  IDOMPageRectTarget, 533
  IDOMPageTarget, 538
  IDOMTarget, 677
g getTargetUri
  IDOMExternalTarget, 276
g getTempStore
  ISession, 948
g getTemporaryDirectory

Generated by Doxygen
INDEX

IsessionId, 948
getTextColor
IDOMOutlineEntry, 524
getTextStyle
IDOMOutlineEntry, 524
getThreads
JawsMako::IDocument, 183
getThumbnail
IDOMFixedPage, 288
JawsMako::IDocumentAssembly, 188
getTileMode
IDOMImageBrush, 408
IDOMVisualBrush, 707
getTileStep
IDOMTilingPatternBrush, 686
ggetTilingType
JawsMako::IDocumentAssembly, 188
getTime
IEDLTime, 740
getTint
IDOMOPI13, 514
getTintTransform
IDOMColorSpaceDeviceN, 248
IDOMDeviceNColorant, 270
getTop
IDOMPageRectTarget, 533
getTransferFunction
IDOMSoftMaskBrush, 666
getTransforms
JawsMako::ITransformChain, 989
getTransparency
IDOMOPI13, 514
getTrimBox
IDOMFixedPage, 289
getType
JawsMako::IAnnotation, 148
getUOffset
IDOMGlyph, 349
ggetURI
IDOMCatalog, 218
getUnderlyingColorSpace
IDOMColorSpaceIndexed, 256
getUnicodeString
IDOMGlyphs, 367
getUnixParameters
IDOMActionLaunch, 194
getUri
IDOMResource, 618
getUsage
JawsMako::IAnnotationAppearance, 154
getUsers
JawsMako::IOptionalContentGroupUsage, 844
ggetVOffset
IDOMGlyph, 349
getValue
IDOMJobTkValue, 464
ggetVersion
IDOMJobTkContent, 446
getIdMOP13, 514
IDOMOPI20, 516
IDOMOPI, 504
getVerticesPerRow
IDOMShadingPatternType4567Brush, 653
ggetViewBox
IDOMImageBrush, 408
IDOMVisualBrush, 707
getViewBoxUnits
IDOMImageBrush, 409
IDOMVisualBrush, 708
getViewPort
IDOMImageBrush, 409
IDOMVisualBrush, 708
getViewPortUnits
IDOMImageBrush, 409
IDOMVisualBrush, 709
getView
IDOMTilingPatternBrush, 686
IDOMVisualBrush, 709
getWhitePoint
IDOMColorSpaceLAB, 259
getWidgets
JawsMako::IForm, 754
getWidth
IDOMFixedPage, 289
IDOMImageFrame, 781
getWinParameters
IDOMActionLaunch, 194
ggetXResolution
IDOMImageFrame, 781
getXmpPacket
JawsMako::IDocumentAssembly, 189
ggetYear
IEDLTime, 740
getZoom
IDOMPageRectTarget, 534
getF
IDOMOPI13, 509
getsGrayProfile
IColorManager, 174
getsRGBProfile
IColorManager, 174
getscRGBProfile
IColorManager, 174
GlyphID
IDOMGlyph, 345
groupsIsVisible
JawsMako::IOptionalContent, 833
hasChildNodes
IDOMNode, 495
hash
IDOMHashable, 394
IInputStream, 803
hashValue
objcclassid.h, 1076
haveMoreEnumerationItems
IFontHeaderWriteSegmentBlockEnumerator, 745

Generated by Doxygen
IFontPCL5WriteSegmentBlockEnumerator, 748
IColorManager, 164
convertColors, 166, 167
createCalibratedGrayProfile, 168
createCalibratedRGBProfile, 168
createGrayProfile, 169
get, 169
getCMYKSWOPProfile, 170
defaultIntentForICCBasedSpace, 170
defDeviceCMYKIntercept, 170
defDeviceGrayIntercept, 170
defDeviceRGBIntercept, 171
defEmbeddedPSCSAForICCBasedSpace, 171
defaultIntentForICCBasedSpace, 170
getNumComponentsForICCBasedSpace, 172
getNumComponentsForICCProfile, 172
getPostScriptCSAForICCBasedSpace, 172
getProfileForSpace, 173
getProfileVersionForICCBasedSpace, 173
getGrayProfile, 174
getRGBProfile, 174
getsRGBProfile, 174
interceptSpace, 174
setDeviceCMYKIntercept, 175
setDeviceGrayIntercept, 175
setDeviceRGBIntercept, 175
setMapDeviceGrayToCMYKBlack, 176
IDOMActionArray, 190
addAction, 191
getActionsCount, 191
getActionsEnum, 191
gerType, 192
IDOMActionLaunch, 192
getFile, 193
getMacParameters, 193
getNewWindow, 194
gerType, 194
getUnixParameters, 194
getWinParameters, 194
setFile, 195
setMacParameters, 195
setNewWindow, 195
setUnixParameters, 196
setWinParameters, 196
IDOMArcSegment, 196
classID, 199
convertToSimpleSegment, 199
getsLargeArc, 199
getPoint, 200
getRadiusX, 200
getRadiusY, 200
getRotationAngle, 200
getSweepDirection, 201
getsLargeArc, 201
setPoint, 201
setRadiusX, 202
setRadiusY, 202
setRotationAngle, 203
setSweepDirection, 203
SweepDirection, 198
IDOMArcSegment::Data, 104
IDOMAudioFile, 203
classID, 204
IDOMAudioFile::Data, 111
IDOMBrush, 205
getBrushType, 206
getOpacity, 206
setOpacity, 206
IDOMCanvas, 207
classID, 209
getAutomationPropertiesHelpText, 209
getAutomationPropertiesName, 209
getEdgeMode, 210
gLanguage, 210
getNavigateLink, 210
getResourceDictionary, 211
setAutomationPropertiesHelpText, 211
setAutomationPropertiesName, 212
setEdgeMode, 212
setLanguage, 212
setNavigateLink, 213
setResourceDictionary, 213
IDOMCanvas::Data, 107
IDOMCatalog, 214
classID, 215
createNewDOMid, 215, 216
cCount, 216
getIdByIndex, 216
getIdByNumbers, 217
getIdByURI, 217
gObject, 218
gURI, 218
registerNumbers, 218
registerObject, 220
unregisterObject, 220
IDOMCharPathGroup, 221
classID, 222
gBlendMode, 222
gCharPathType, 223
gClippedGroup, 223
gFill, 223
gStrokePath, 224
gSetBlendMode, 224
gCharPathType, 224
gClippedGroup, 226
gFill, 226
gStrokePath, 226
IDOMCharPathGroup::Data, 108
IDOMColor, 227
classID, 228
create, 228
cFromArray, 229
cFromVect, 229
getAlpha, 230
colorSpace, 230
IDX COLOR SPACES  

getComponentValue, 231  
setAlpha, 231  
setColorSpace, 231, 232  
setComponentValue, 233  
IDOMColorSpace, 233  
eColorSpaceType, 235  
equals, 236  
ggetColorSpaceType, 236  
gGetComponentRange, 236  
gGetComponentsHaveSameRange, 237  
gGetDefaultRenderingIntent, 237  
gGetNumComponents, 237  
similar, 237  
IDOMColorSpaceDeviceCMYK, 239  
classID, 240  
create, 241  
IDOMColorSpaceDeviceCMY, 238  
classID, 239  
create, 241  
IDOMColorSpaceDeviceGray, 241  
classID, 242  
create, 242  
IDOMColorSpaceDeviceN::Data, 117  
IDOMColorSpaceDeviceRGB, 249  
classID, 249  
create, 250  
IDOMColorSpaceDeviceN, 243  
classID, 245  
create, 245, 246  
getAlternateColorSpace, 246  
getColorant, 247  
gGetIsNChannel, 247  
gGetPrintingOrder, 247  
gGetProcessColorSpace, 247  
gGetProcessComponentNames, 248  
gGetTintTransform, 248  
IDOMColorSpaceICCBased, 250  
classID, 252  
create, 252  
getAlternateColorSpace, 252  
gGetICCProfile, 253  
setICCProfile, 253  
IDOMColorSpaceICCBased::Data, 114  
IDOMColorSpaceIndexed, 253  
classID, 255  
create, 255, 256  
gGetMappingFunction, 256  
gGetUnderlyingColorSpace, 256  
IDOMColorSpaceIndexed::Data, 114  
IDOMColorSpaceIndexed::Data, 116  
IDOMColorSpaceRGB::Data, 118  
IDOMColorSpaceLAB, 257  
classID, 258  
create, 258  
getBlackPoint, 259  
gGetRangeAB, 259  
gGetWhitePoint, 259  
IDOMColorSpacesGray, 262  
classID, 262  
create, 263  
IDOMColorSpacesRGB, 263  
classID, 264  
create, 264  
IDOMColorSpacescRGB, 260  
classID, 261  
create, 261  
IDOMCompositeImage, 265  
classID, 266  
create, 266  
gStream, 267  
setStream, 267  
IDOMCompositeImage::Data, 78  
IDOMDePremultiplierFilter::Data, 87  
IDOMDePremultiplyFilter, 268  
IDOMDeviceNColorant, 268  
classID, 269  
getAlternateColorSpace, 269  
gDotGainFunction, 269  
gGetName, 270  
gGetSolidity, 270  
gGetTintTransform, 270  
IDOMDeviceNColorant::Data, 115  
IDOMExponentialFunction, 271  
classID, 272  
evaluate, 272, 273  
gGetExponent, 273  
gGetOutputC0, 274  
gGetOutputC1, 274  
IDOMExponentialFunction::Data, 131  
IDOMExternalTarget, 275  
gGetIsMap, 276  
gGetTargetType, 276  
gGetTargetUri, 276  
gSetIsMap, 276  
gSetTargetUri, 277  
IDOMFilteredImage, 277  
classID, 279  
create, 279, 280  
gGetFilterAtIndex, 280  
gNumFilters, 280  
gGetSourceImage, 281  
gStream, 281  
gPushFilter, 281  
gSetStream, 282  
IDOMFilteredImage::Data, 92  
IDOMFixedPage, 282  
classID, 285  
create, 285  
gGetAnnotationManager, 285  
gGetBleedBox, 286  
gGetContentBox, 286  
gGetCropBox, 287  
gGetHeight, 287  
gGetLanguage, 287  
gGetLinkManager, 288  
gGetPageGroup, 288  
gGetResourceIdictionary, 288  
Generated by Doxygen
addFigure, 546
classID, 546
create, 547
eFillRule, 546
getBounds, 547
getFigureCollectionEnum, 548
getFiguresCount, 548
getFillRule, 548
getFlattenedGeometry, 548
getIsRect, 549
getRenderTransform, 549
getShape, 549
getSimplifiedGeometry, 550
setFillRule, 550
setRenderTransform, 551
IDOMPathGeometry::Data, 109
IDOMPathNode, 551
addStrokeDash, 558
classID, 558
createFilled, 558
createImage, 559
createStroked, 559
eStrokeLineCap, 556
eStrokeLineJoin, 556
eStrokeMiterLimitTreatment, 556
getAutomationPropertiesHelpText, 560
getAutomationPropertiesName, 561
getBlendMode, 561
getClip, 561
getEdgeMode, 562
getFill, 562
getIsDashed, 563
getLanguage, 563
getNavigateLink, 563
getOpacity, 564
getOpacityMask, 564
getPathData, 564
getRenderTransform, 565
getShape, 565
getShouldZeroWidthLinesBeVisible, 566
setLanguage, 572
setNavigateLink, 573
setOpacity, 573
setOpacityMask, 573
setPathData, 574
setRenderTransform, 574
setShapesToBeVisible, 574
setSnapsToDevicePixels, 575
setStroke, 575
setStrokeDashLineCap, 575
setStrokeDashOffset, 576
setStrokeEndLineCap, 576
setStrokeLineJoin, 577
setStrokeMiterLimit, 577
setStrokeMiterLimitTreatment, 578
setStrokeStartLineCap, 578
setStrokeThickness, 578
split, 579
IDOMPathNode::Data, 102
IDOMPathSegment, 579
getBounds, 580
getEndPoint, 581
getIsStroked, 581
setIsStroked, 581
IDOMPolyBezierSegment, 591
addPoint, 592
classID, 593
getPointEnum, 593
getPointsCount, 593
IDOMPolyBezierSegment::Data, 106
IDOMPolyLineSegment, 594
addPoint, 595
classID, 595
getPointEnum, 595
getPointsCount, 595
IDOMPolyLineSegment::Data, 104
IDOMPolyQuadraticBezierSegment, 596
addPoint, 597
classID, 598
convertToCubicBezierSegment, 598
getPointEnum, 598
getPointsCount, 599
IDOMPolyQuadraticBezierSegment::Data, 106
IDOMPostScriptCalculatorFunction, 599
classID, 600
evaluate, 600, 601
gtCalculator, 601
gtCalculatorAsPostScriptStream, 602
IDOMPostScriptCalculatorFunction::Data, 133
IDOMPrintTicket, 602
classID, 603
IDOMPrintTicket::Data, 110
IDOMRadialGradientBrush, 604
classID, 605
createShading, 606
center, 606
gtGradientOrigin, 606
gtRadiusX, 607
getRadiusY, 607
getSimplifiedGradient, 607
setCenter, 608
setGradientOrigin, 608
setRadiusX, 608
setRadiusY, 609
IDOMRadialGradientBrush::Data, 87
IDOMRawDataFile, 609
classID, 610
IDOMRawDataFile::Data, 112
IDOMRawImage, 611
classID, 612
createWriterAndImage, 612
getSynthetic, 613
IDOMRawImage::Data, 74
IDOMRecombineAlpha, 614
IDOMRecombineAlphaImage::Data, 77
IDOMRecombineImage, 614
classID, 615
getStream, 615
setStream, 616
IDOMRecombineImage::Data, 76
IDOMResource, 616
getStream, 617
getStreamLength, 618
getUri, 618
setStream, 618
setUri, 619
IDOMResourceDictionary, 619
classID, 621
get, 621
put, 621
IDOMResourceDictionary::Data, 112
IDOMSampledFunction, 622
classID, 623
evaluate, 623, 624
getBitsPerSample, 624
getInputEncode, 625
gernterpolationMethod, 625
gerOutpuDecode, 625
getTableData, 627
getTableDimension, 627
IDOMSampledFunction::Data, 130
IDOMSecurityInfo, 628
IDOMShadingPatternBrush, 628
getAntiAlias, 629
getBBBox, 630
getBackgroundColor, 630
g getColorSpace, 630
getShadingType, 631
setAntiAlias, 631
setBBBox, 632
setBackgroundColor, 631
setColorSpace, 632
IDOMShadingPatternType1Brush, 633
classID, 635
getDomain, 635
getFunction, 635
getMatrix, 636
setDomain, 636
setFunction, 637
setMatrix, 637
IDOMShadingPatternType1Brush::Data, 98
IDOMShadingPatternType2Brush, 637
classID, 639
getCoords, 639
getDomain, 640
getEquivalentSimpleBrush, 640
getExtend, 641
getFunction, 641
setCoords, 641
setDomain, 642
setExtend, 642
setFunction, 642
IDOMShadingPatternType2Brush::Data, 100
IDOMShadingPatternType3Brush, 643
classID, 644
getCoords, 645
getDomain, 645
getEquivalentSimpleBrush, 645
getExtend, 646
getFunction, 646
setCoords, 647
setDomain, 647
setExtend, 647
setFunction, 648
IDOMShadingPatternType3Brush::Data, 100
IDOMShadingPatternType4567Brush, 648
classID, 650
getBitsPerComponent, 650
getsBitsPerCoordinate, 651
getBitsPerFlag, 651
dataSource, 652
getDecode, 652
gerDecodeSize, 652
gerFunction, 653
gerVerticesPerRow, 653
setBitsPerComponent, 653
setBitsPerCoordinate, 654
setBitsPerFlag, 654
setDataSource, 654
setDecode, 656
setFunction, 656
setShadingType, 656
setVerticesPerRow, 657
IDOMShadingPatternType4567Brush::Data, 103
IDOMShape, 657
classID, 659
completelyContainsShape, 659
difference, 660
getAsImage, 660
getBounds, 661
getsIsEmpty, 661
getsRect, 661
getResolution, 661
intersect, 662
intersects, 662
isEqualTo, 662
unite, 663
IDOMShape::Data, 114
IDOMSoftMaskBrush, 663
classID, 665
create, 665
getBackdropColor, 666
group, 666
getSoftMaskType, 666
getTransferFunction, 666
IDOMSoftMaskBrush::Data, 95
IDOMSolidColorBrush, 667
classID, 668
create, 668
getColor, 669
setColor, 669
IDOMSolidColorBrush::Data, 95
IDOMSolidColorBrush::Data, 83
IDOMStandardPDFSecurityInfo, 670
ePermissionsFlags, 671
IDOMStitchingFunction, 671
classID, 673
evaluate, 673
getBoundsVector, 674
getEncodeVector, 674
getFunctionAtIndex, 674
getNumFunctions, 675
IDOMStitchingFunction::Data, 132
IDOMTIFFImage, 678
classID, 679
create, 679
eTIFFCompression, 679
encode, 680
IDOMTarget, 675
eTargetType, 676
getTargetType, 677
IDOMTilingPatternBrush, 681
classID, 683
create, 683
getBBox, 683
getEquivalentVisualBrush, 684
generate, 684
getPatternColor, 685
getPatternType, 685
getRenderTransform, 685
getTilingStep, 686
generateTilingType, 686
generate Visual, 686
setBBox, 687
setPatternType, 687
setPatternColor, 687
setRenderTransform, 689
setTilingStep, 689
setTilingType, 689
setVisual, 690
IDOMTilingPatternBrush::Data, 97
IDOMTransformableBrush, 690
generateRenderTransform, 691
setRenderTransform, 692
IDOMTransparencyGroup, 692
classID, 694
create, 694
generateBlendMode, 695
generateColorSpace, 695
generateIsIsolated, 696
generateIsKnockout, 696
generateOpacity, 696
generateOpacityMask, 696
setBlendMode, 697
setColorSpace, 697
setIsIsolated, 698
setIsKnockout, 698
setOpacity, 698
setOpacityMask, 699
IDOMTransparencyGroup::Data, 72
IDOMType3Font, 699
addGlyph, 701
classID, 701
deleteGlyphs, 701
generateBBox, 702
generateGlyph, 702, 703
generateID, 703
IDOMType3Font::Data, 125
IDOMVisualBrush, 704
classID, 705
create, 706
generateEquivalentSimpleVisualBrush, 706
generateEquivalentTilingBrush, 707
generateTileMode, 707
generateViewBox, 707
generateViewBoxUnits, 708
generateViewPort, 708
generateViewPortUnits, 709
generateVisual, 709
setTileMode, 709
setViewBox, 710
setViewBoxUnits, 710
setViewPort, 711
setViewPortUnits, 711
setVisual, 712
IDOMVisualBrush::Data, 93
IDOMVisualRoot, 712
classID, 713
IDOMWMPImage, 713
classID, 714
IEDLClassFactory, 715
createInstance, 716
findNamedClass, 716
getSingleton, 716
registerClass, 717
registerNamedClass, 717
IEDLError, 718
IEDLFontSystemFont, 718
classID, 720
getAscentDescentRatio, 720
generateAverageCharAspectRatio, 721

Generated by Doxygen
getFontDecorationFlags, 721
setFontFamily, 721
setFontFilePath, 721
setFontName, 722
setFontWeight, 722
loadFont, 723
setFontFilePath, 723
IEDLNamespace, 724
classID, 725
getNamespace, 725
getPrefix, 725
setNamespace, 726
setPrefix, 726
IEDLNamespace::Data, 128
IEDLObject, 727
clone, 728
classID, 728
init, 729
IEDLStream, 729
getPos, 730
isValid, 731
open, 731
IEDLTempStore, 732
classID, 733
createTemporaryObject, 733
createTemporaryObjectWithContents, 733
createTemporaryReaderWriter, 734
createTemporaryReaderWriterPair, 734
createTemporaryStreamWithContents, 734
IEDLTempStore::Data, 116
IEDLTempStoreObject, 735
createReader, 736
createWriter, 736
IEDLTime, 737
compare, 738
classID, 744
getSegmentBlockItem, 744
getSegmentBlockSize, 745
getSegmentItem, 745
haveMoreEnumerationItems, 745
nextEnumerationItem, 745
writeSegmentBlock, 746
setFontDecorationFlags, 721
gClassId, 747
gEnumerationCount, 748
gEnumerationItemCount, 748
haveMoreEnumerationItems, 748
nextEnumerationItem, 748
writeEnumerationItemBlock, 749
ILImageDecoder, 766
ILImageEncoder, 771
ILImageFrame, 777
gBPS, 778
gColorSpace, 778
gExtraChannelType, 779
gHasAlphaChannel, 779
gHasICCProfile, 779
gHasMask, 779
gHeight, 780
gICCProfile, 780
gMatrix, 780
getNumChannels, 780
getNumExtraChannels, 781
gRawBytesPerRow, 781
getWidth, 781
gxResolution, 781
readScanLine, 782
skipScanLines, 783
ILImageFrameReader, 783
classID, 783
gMatrix, 784
writeScanLine, 785
ILImageFrameWriter, 785
classID, 785
gRawBytesPerRow, 786
ILInputEnum, 793
ILInputEnumRC, 793
ILInputPushbackStream, 794
ILInputStream, 795
completeRead, 797
createCompositeStream, 797
createFromFile, 797
createFromFileShared, 798
createFromFlateCompressed, 799
createFromLz4Compressed, 800
createFromMemory, 800
createFromFileWithContents, 801
createFromRAUserFunc, 801
classID, 802
classID, 802
copy, 860
classID, 860
classID, 861
classID, 862
classID, 863
classID, 863
setSessionTempRootDirectory, 1073

JawsMako::CAnnotationBorder, 45
  CAnnotationBorder, 46
dash, 46
eBorderStyle, 46
width, 47

JawsMako::CQuadPoint, 54
JawsMako::CRectInset, 54
JawsMako::CTemporaryStoreParameters, 56
CTemporaryStoreParameters, 56
JawsMako::CTransformState, 69
  inUncoloredTilingBrush, 71
stateInsideBrush, 70
stateInsideNode, 71
transformPriv, 71

JawsMako::IAnnotation, 142
  addAppearance, 145
close, 146
eAnnotationType, 144
gGetAppearance, 146
getAppearances, 146
getBorder, 147
color, 147
gGetContents, 147
gGetFlags, 147
getModificationTime, 147
Rect, 148
getReference, 148
getState, 148
type, 148
matchesReference, 149
rotate, 149
setBorder, 149
color, 150
setContents, 150
setState, 151

JawsMako::IAnnotationAppearance, 152
  clone, 153
create, 153
gGetForm, 153
gGetScaledAppearance, 153
getState, 154
gGetUsage, 154

JawsMako::IAnnotationReference, 155
equals, 155
gCreateAnnotationReferenceFromTag, 156
generateXMLForDocument, 157

JawsMako::IAnnotationUtils, 156
  createAnnotationReferenceFromTag, 156
  generateXMLForDocument, 157

JawsMako::IComplexColorSimplifierTransform, 176
create, 177

JawsMako::IDocument, 179
  addNamedDestination, 181
  appendPage, 181
close, 181
create, 181
gGetForm, 182
gGetJobTicket, 182
gGetNumPages, 182
gGetOutline, 182
gGetPage, 183
gGetThreads, 183
gGetNamedPage, 183
removePage, 184
setNamedDestinations, 184
setOutline, 184

JawsMako::IDocumentAssembly, 185
  appendDocument, 186
close, 187
create, 187
gGetDocument, 187
gGetJobMetadata, 187
gGetJobTicket, 188
gGetNumDocuments, 188
gGetSecurityInfo, 188
gGetThumbnail, 188
gGetXmpPacket, 189
insertDocument, 189
removeDocument, 189
gSetThumbnail, 190

JawsMako::IFormField, 756
  addChildField, 758
  addChildWidget, 758
close, 758
gGetFieldId, 758
gGetPartialName, 758

Generated by Doxygen
removeChildField, 760
removeChildWidget, 761
setPartialName, 761
widgetInSubtree, 761, 762
JawsMako::IFormUnpackerTransform, 762
create, 763
JawsMako::IFreeTextAnnotation, 764
getCalloutLine, 765
getRectInset, 765
JawsMako::IImageDownsamplerTransform, 767
create, 769
setDownsampleMaskedImages, 770
setUseMaskResolutionForMaskedImages, 770
JawsMako::IImageEncoderTransform, 772
create, 774
eEncodeFormat, 773
setColorTIFFCompression, 774
setGrayTIFFCompression, 774
setJPEGQuality, 775
setMonoTIFFCompression, 775
setPreferredSizeColorFormat, 775
setPreferredSizeFormat, 775
setPreferredSizeGrayFormat, 776
setPreferredSizeMonoFormat, 776
setTIFFCompression, 776
JawsMako::IImageMergerTransform, 787
create, 788
setAllowMaskedResults, 788
JawsMako::ILinkAnnotation, 789
create, 790
getInkList, 790
setInkList, 790
JawsMako::IInput, 791
create, 792
open, 792
JawsMako::IJawsMako, 805
create, 806
enablePDFOutput, 807
enablePSOutput, 807
findFont, 807
JawsMako::IJawsRenderer, 808
create, 810
render, 810
renderAntiAliased, 811
renderAntiAliasedToFrameBuffer, 811
renderMonochrome, 812
renderMonochromeToFrameBuffer, 812
renderSeparations, 813
renderToFrameBuffer, 814
renderToFrameBufferPadAndReverse, 815
JawsMako::IJawsRenderer::CSpotHalftone, 55
JawsMako::IJawsRenderer::CThresholdArrayHalftone, 57
JawsMako::IJawsRenderer::CThresholdHalftone, 58
JawsMako::IJawsRenderer::IHalftone, 766
JawsMako::ILineAnnotation, 816
captionOffset, 818
getInteriorColor, 818
getLeaderLineExtensionsLength, 818
getLeaderLineLength, 818
getLeaderLineOffset, 819
getLineEndpoints, 819
setInteriorColor, 819
setLineEndpoints, 820
JawsMako::ILinkAnnotation, 820
calloutLine, 821
setQuadPoints, 821
JawsMako::IMarkedContentArtifactDetails, 822
JawsMako::IMarkedContentDetails, 823
JawsMako::IMarkedContentStructureDetails, 824
JawsMako::IMarkupAnnotation, 825
captionOffset, 826
getCreationTime, 826
getOpacity, 826
getPopupReference, 826
setAuthor, 827
setCreationTime, 827
setOpacity, 827
setPopups, 828
JawsMako::INamedDestination, 828
calloutLine, 829
captionOffset, 829
calloutLineExtensionsLength, 829
calloutLineLength, 829
calloutLineOffset, 829
captionOffset, 830
calloutLineExtensionsLength, 830
calloutLineLength, 830
calloutLineOffset, 830
calloutLineExtensionsLength, 830
calloutLineLength, 830
calloutLineOffset, 830
JawsMako::IOptionalContent, 831
addGroup, 832
forceGroupState, 832
groupIsVisible, 833
makeNodeOptional, 833
makeNodeOptionalWithNewGroup, 834
setDefaultConfiguration, 834
JawsMako::IOptionalContentConfiguration, 835
JawsMako::IOptionalContentConfiguration::COrderEntry, 837
JawsMako::IOptionalContentDetails, 837
eVisibilityPolicy, 838
getIsVisible, 839
JawsMako::IOptionalContentFixerTransform, 839
create, 840
JawsMako::IOptionalContentGroup, 841
JawsMako::IOptionalContentGroupReference, 842
JawsMako::IOptionalContentGroupUsage, 843
groups, 844
recommendedVisibilityForCategories, 845
setLanguage, 845
setPageElement, 846
setUsers, 846
JawsMako::IOptionalContentGroupUsageApplication, 846
JawsMako::IOptionalContentVisibilityExpression, 848
eVisibilityExpressionOperation, 849
evaluate, 849
JawsMako::IOutput, 850
create, 851
openWriter, 852, 853
setAllowedPermissionsFlags, 853

writeAssembly, 854, 855
JawsMako::IOutputAbort, 855
JawsMako::IOutputIntent, 857
getCheckSum, 858
getProfileFileSpecifications, 858
JawsMako::IOutputWriter, 864
beginDocument, 865
writePage, 865
JawsMako::IOverprintSimulationTransform, 866
setSimulateBlackDeviceGrayTextOverprint, 867
JawsMako::IPCL5Input, 877
create, 879
enableUnencapsulatedMode, 879
setDefaultCopies, 879
setDefaultDuplex, 879
setDefaultDuplexBindingMode, 880
setDefaultLandscape, 880
setDefaultManualFeed, 880
setDefaultPaperSize, 880
setDefaultsFromPjl, 881
setRopResolution, 881
JawsMako::IPCL5Output, 882
setEmtPjl, 883
setImageCompression, 883
setMediaSource, 884
setOpenStream, 884
setResolution, 884
setVersion, 884
JawsMako::IPCLXLInput, 885
create, 886
enableUnencapsulatedMode, 886
setDefaultCopies, 887
setDefaultDuplex, 887
setDefaultDuplexBindingMode, 887
setDefaultLandscape, 888
setDefaultManualFeed, 888
setDefaultPaperSize, 888
setDefaultFromPjl, 889
setRopResolution, 889
JawsMako::IPCLXLOutput, 890
setEmtPjl, 891
setOpenStream, 891
JawsMako::IPDFInput, 891
create, 893
scanPdfForFonts, 893
setFailOnFontFallback, 894
setPassword, 894
JawsMako::IPDFInput::CPdfFontInfo, 52
JawsMako::IPDFInput::CPdfScannedInk, 53
JawsMako::IPDFOutput, 895
ePDFVersion, 898
ePdfXDeviceNHandling, 899
ePdfXExtendedGraphicsStateHandling, 899
ePdfXOptionalContentHandling, 899
setAllowRestrictedFonts, 900
setAlwaysEmbedFonts, 900
setAutoRotatePages, 900
setColorImageMaxResolution, 901
setCompressObjects, 901
setCompressPages, 902
setConvertAllColors, 902
setConvertGray, 902
setDownsampleMaskedImages, 902
setEmbedBase14Fonts, 903
setEmbedFonts, 903
setEnableIncrementalOutput, 903
setEncryption, 904
setGrayImageMaxResolution, 905
setJPEGQuality, 905
setMonoImageMaxResolution, 906
setNeverEmbedFonts, 906
setOutputIntent, 906
setPdfXDeviceNErrorHandling, 906
setPdfXExtendedGraphicsStateErrorHandling, 907
setPdfXOptionalContentErrorHandling, 907
setPreferredColorImageCompression, 907
setPreferredGrayImageCompression, 908
setPreferredMonoImageCompression, 908
setPreferredRenderedImageCompression, 908
setProducer, 909
setReencodeImages, 909
setRenderResolution, 909
setSubsetFonts, 910
setTargetColorSpace, 910
setTargetProfile, 911
setUseMaskResolutionForMaskedImages, 911
setVersion, 911
JawsMako::IPJLParser, 912
create, 914
getAttributes, 914
parse, 914
JawsMako::IPJLParser::CPjlAttributeValue, 53
key, 53
modifier, 54
value, 54
JawsMako::IPSInjector, 918
afterBeginPageSetup, 919
afterBeginSetup, 919
afterLastByte, 920
beforeEndPageSetup, 920
beforeEndSetup, 920
beforeFirstByte, 922
beforePsHeader, 922
beforeShowpage, 922
JawsMako::IPSOutput, 923
setConvertAllObjectsToTargetColorSpace, 924
setStreamingOutput, 924
setTargetColorSpace, 924
setTargetProfile, 925
JawsMako::IPage, 867
clone, 869
create, 869
getContent, 870
getJobTicket, 870
release, 870
removeAnnotation, 870
<table>
<thead>
<tr>
<th>Class</th>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>JawsMako::IPageCropperTransform</td>
<td>create</td>
<td>871</td>
</tr>
<tr>
<td>JawsMako::IPageLayout</td>
<td>create</td>
<td>872</td>
</tr>
<tr>
<td>JawsMako::IPageLayoutData</td>
<td>create</td>
<td>874</td>
</tr>
<tr>
<td>JawsMako::IPageLayoutNode</td>
<td>create</td>
<td>876</td>
</tr>
<tr>
<td>JawsMako::IPolyAnnotation</td>
<td>getPoints</td>
<td>915</td>
</tr>
<tr>
<td>JawsMako::IPolyAnnotation</td>
<td>setPoints</td>
<td>916</td>
</tr>
<tr>
<td>JawsMako::IPopupAnnotation</td>
<td>create</td>
<td>917</td>
</tr>
<tr>
<td>JawsMako::IPopupAnnotation</td>
<td>getOpen</td>
<td>918</td>
</tr>
<tr>
<td>JawsMako::IPopupAnnotation</td>
<td>setOpen</td>
<td>918</td>
</tr>
<tr>
<td>JawsMako::IPopupAnnotation</td>
<td>create</td>
<td>917</td>
</tr>
<tr>
<td>JawsMako::IPopupAnnotation</td>
<td>getOpen</td>
<td>918</td>
</tr>
<tr>
<td>JawsMako::IPopupAnnotation</td>
<td>setOpen</td>
<td>918</td>
</tr>
<tr>
<td>JawsMako::IPopupAnnotation</td>
<td>create</td>
<td>917</td>
</tr>
<tr>
<td>JawsMako::IPopupAnnotation</td>
<td>getOpen</td>
<td>918</td>
</tr>
<tr>
<td>JawsMako::IPopupAnnotation</td>
<td>setOpen</td>
<td>918</td>
</tr>
<tr>
<td>JawsMako::IPopupAnnotation</td>
<td>create</td>
<td>917</td>
</tr>
<tr>
<td>JawsMako::IPopupAnnotation</td>
<td>getOpen</td>
<td>918</td>
</tr>
<tr>
<td>JawsMako::IPopupAnnotation</td>
<td>setOpen</td>
<td>918</td>
</tr>
<tr>
<td>JawsMako::IPopupAnnotation</td>
<td>create</td>
<td>917</td>
</tr>
<tr>
<td>JawsMako::IPopupAnnotation</td>
<td>getOpen</td>
<td>918</td>
</tr>
<tr>
<td>JawsMako::IPopupAnnotation</td>
<td>setOpen</td>
<td>918</td>
</tr>
<tr>
<td>JawsMako::IPopupAnnotation</td>
<td>create</td>
<td>917</td>
</tr>
<tr>
<td>JawsMako::IPopupAnnotation</td>
<td>getOpen</td>
<td>918</td>
</tr>
<tr>
<td>JawsMako::IPopupAnnotation</td>
<td>setOpen</td>
<td>918</td>
</tr>
<tr>
<td>JawsMako::IPopupAnnotation</td>
<td>create</td>
<td>917</td>
</tr>
<tr>
<td>JawsMako::IPopupAnnotation</td>
<td>getOpen</td>
<td>918</td>
</tr>
<tr>
<td>JawsMako::IPopupAnnotation</td>
<td>setOpen</td>
<td>918</td>
</tr>
<tr>
<td>JawsMako::IRedactionAnnotation</td>
<td>getInteriorColor</td>
<td>936</td>
</tr>
<tr>
<td>JawsMako::IRedactionAnnotation</td>
<td>getQuadPoints</td>
<td>936</td>
</tr>
<tr>
<td>JawsMako::IRedactionAnnotation</td>
<td>setInteriorColor</td>
<td>936</td>
</tr>
<tr>
<td>JawsMako::IRedactionAnnotation</td>
<td>setQuadPoints</td>
<td>937</td>
</tr>
<tr>
<td>JawsMako::IRedactorTransform</td>
<td>create</td>
<td>939</td>
</tr>
<tr>
<td>JawsMako::IRendererTransform</td>
<td>create</td>
<td>942</td>
</tr>
<tr>
<td>JawsMako::IRendererTransform</td>
<td>setGenerateMasks</td>
<td>942</td>
</tr>
<tr>
<td>JawsMako::IRendererTransform</td>
<td>setMonochromeMode</td>
<td>943</td>
</tr>
<tr>
<td>JawsMako::IRendererTransform</td>
<td>setRenderTransparentNodesOnPageGroup&lt;Mismatch&gt;</td>
<td>943</td>
</tr>
<tr>
<td>JawsMako::IRendererTransform</td>
<td>setSpotHalftone</td>
<td>943</td>
</tr>
<tr>
<td>JawsMako::IRendererTransform</td>
<td>setThresholdHalftone</td>
<td>944</td>
</tr>
<tr>
<td>JawsMako::ISVGGenerator</td>
<td>create</td>
<td>969</td>
</tr>
<tr>
<td>JawsMako::ISVGGenerator</td>
<td>generateSVG</td>
<td>970</td>
</tr>
<tr>
<td>JawsMako::ISVGGenerator</td>
<td>getResource</td>
<td>971</td>
</tr>
<tr>
<td>JawsMako::ISVGGenerator</td>
<td>getResources</td>
<td>971</td>
</tr>
<tr>
<td>JawsMako::ISVGGenerator</td>
<td>setEnableImageDownsampling</td>
<td>972</td>
</tr>
<tr>
<td>JawsMako::ISVGGenerator</td>
<td>setPageResolutionCallback</td>
<td>972</td>
</tr>
<tr>
<td>JawsMako::ISVGGenerator</td>
<td>setTargetResolutionCallback</td>
<td>972</td>
</tr>
<tr>
<td>JawsMako::IShapeAnnotation</td>
<td>getInteriorColor</td>
<td>950</td>
</tr>
<tr>
<td>JawsMako::IShapeAnnotation</td>
<td>getRectInset</td>
<td>951</td>
</tr>
<tr>
<td>JawsMako::IShapeAnnotation</td>
<td>setInteriorColor</td>
<td>951</td>
</tr>
<tr>
<td>JawsMako::ISkiaRenderer</td>
<td>create</td>
<td>952</td>
</tr>
<tr>
<td>JawsMako::ISkiaRenderer</td>
<td>drawNode</td>
<td>953</td>
</tr>
<tr>
<td>JawsMako::ISkiaRenderer</td>
<td>flushCaches</td>
<td>953</td>
</tr>
<tr>
<td>JawsMako::ISoundAnnotation</td>
<td>getSoundAsWav</td>
<td>954</td>
</tr>
<tr>
<td>JawsMako::ISoundAnnotation</td>
<td>create</td>
<td>953</td>
</tr>
<tr>
<td>JawsMako::ISoundAnnotation</td>
<td>getSoundAsWav</td>
<td>954</td>
</tr>
<tr>
<td>JawsMako::ISoundAnnotation</td>
<td>create</td>
<td>955</td>
</tr>
<tr>
<td>JawsMako::ISoundAnnotation</td>
<td>getIntent</td>
<td>956</td>
</tr>
<tr>
<td>JawsMako::ISoundAnnotation</td>
<td>getName</td>
<td>956</td>
</tr>
<tr>
<td>JawsMako::ISoundAnnotation</td>
<td>setIntent</td>
<td>957</td>
</tr>
<tr>
<td>JawsMako::ISoundAnnotation</td>
<td>setName</td>
<td>957</td>
</tr>
<tr>
<td>JawsMako::ISoundAnnotation</td>
<td>create</td>
<td>955</td>
</tr>
<tr>
<td>JawsMako::ISoundAnnotation</td>
<td>getSoundAsWav</td>
<td>954</td>
</tr>
<tr>
<td>JawsMako::ISoundAnnotation</td>
<td>create</td>
<td>956</td>
</tr>
<tr>
<td>JawsMako::ISoundAnnotation</td>
<td>getIntent</td>
<td>956</td>
</tr>
<tr>
<td>JawsMako::ISoundAnnotation</td>
<td>getName</td>
<td>956</td>
</tr>
<tr>
<td>JawsMako::ISoundAnnotation</td>
<td>setIntent</td>
<td>957</td>
</tr>
<tr>
<td>JawsMako::ISoundAnnotation</td>
<td>setName</td>
<td>957</td>
</tr>
<tr>
<td>JawsMako::ISoundAnnotation</td>
<td>create</td>
<td>959</td>
</tr>
<tr>
<td>JawsMako::ISoundAnnotation</td>
<td>getSoundAsWav</td>
<td>954</td>
</tr>
<tr>
<td>JawsMako::ISoundAnnotation</td>
<td>create</td>
<td>959</td>
</tr>
<tr>
<td>JawsMako::ITextAnnotation</td>
<td>create</td>
<td>973</td>
</tr>
<tr>
<td>JawsMako::ITextAnnotation</td>
<td>getIconName</td>
<td>974</td>
</tr>
<tr>
<td>JawsMako::ITextAnnotation</td>
<td>setOpen</td>
<td>974</td>
</tr>
<tr>
<td>JawsMako::ITextAnnotation</td>
<td>setIconName</td>
<td>975</td>
</tr>
<tr>
<td>JawsMako::ITextAnnotation</td>
<td>setOpen</td>
<td>975</td>
</tr>
<tr>
<td>JawsMako::ITextMarkupAnnotation</td>
<td>getQuadPoints</td>
<td>976</td>
</tr>
<tr>
<td>JawsMako::ITextMarkupAnnotation</td>
<td>setQuadPoints</td>
<td>976</td>
</tr>
<tr>
<td>JawsMako::ITextRun</td>
<td>getBoundsOnPage</td>
<td>977</td>
</tr>
<tr>
<td>JawsMako::ITextRun</td>
<td>getLocalBounds</td>
<td>977</td>
</tr>
<tr>
<td>JawsMako::ITextSearch</td>
<td>create</td>
<td>979</td>
</tr>
<tr>
<td>JawsMako::ITextSelect</td>
<td>create</td>
<td>980</td>
</tr>
<tr>
<td>JawsMako::ITextSelect</td>
<td>getQuadPoints</td>
<td>979</td>
</tr>
<tr>
<td>JawsMako::ITextSelect</td>
<td>setQuadPoints</td>
<td>979</td>
</tr>
<tr>
<td>JawsMako::ITransform</td>
<td>transform</td>
<td>984</td>
</tr>
<tr>
<td>JawsMako::ITransform</td>
<td>transform</td>
<td>985</td>
</tr>
<tr>
<td>JawsMako::ITransform</td>
<td>transform</td>
<td>986</td>
</tr>
<tr>
<td>JawsMako::ITransform</td>
<td>transformPage</td>
<td>987</td>
</tr>
<tr>
<td>JawsMako::ITransformChain</td>
<td>create</td>
<td>987</td>
</tr>
<tr>
<td>JawsMako::ITransformChain</td>
<td>getTransforms</td>
<td>988</td>
</tr>
<tr>
<td>JawsMako::ITransformChain</td>
<td>pushTransform</td>
<td>988</td>
</tr>
<tr>
<td>JawsMako::ITransformChain</td>
<td>pushTransformFront</td>
<td>988</td>
</tr>
<tr>
<td>JawsMako::ITransformChain</td>
<td>removeTransform</td>
<td>990</td>
</tr>
<tr>
<td>JawsMako::ITransformChain</td>
<td>transform</td>
<td>990</td>
</tr>
<tr>
<td>JawsMako::ITransformChain</td>
<td>transformPage</td>
<td>991</td>
</tr>
<tr>
<td>JawsMako::IType3UnpackerTransform</td>
<td>create</td>
<td>991</td>
</tr>
<tr>
<td>JawsMako::IType3UnpackerTransform</td>
<td>getPartialName</td>
<td>994</td>
</tr>
<tr>
<td>JawsMako::IType3UnpackerTransform</td>
<td>getPartialName</td>
<td>994</td>
</tr>
<tr>
<td>JawsMako::IType3UnpackerTransform</td>
<td>setPartialName</td>
<td>994</td>
</tr>
<tr>
<td>JawsMako::IXAMLGenerator</td>
<td>create</td>
<td>997</td>
</tr>
<tr>
<td>JawsMako::IXAMLGenerator</td>
<td>generateXAMLForAppearance</td>
<td>1000</td>
</tr>
<tr>
<td>JawsMako::IXAMLGenerator</td>
<td>generateXAMLForPageAndAnnotationAppearances</td>
<td>1000</td>
</tr>
<tr>
<td>JawsMako::IXAMLGenerator</td>
<td>generateXAMLForPageAnnotationAppearances</td>
<td>1002</td>
</tr>
<tr>
<td>JawsMako::IXAMLGenerator</td>
<td>generateXAML</td>
<td>1002</td>
</tr>
<tr>
<td>JawsMako::IXAMLGenerator</td>
<td>getResource</td>
<td>1002</td>
</tr>
<tr>
<td>JawsMako::IXAMLGenerator</td>
<td>getResources</td>
<td>1002</td>
</tr>
</tbody>
</table>

Generated by Doxygen
setColorImageMaxResolution, 1003
setGrayImageMaxResolution, 1003
setJPEGQuality, 1003
setMergeFonts, 1004
setMonoImageMaxResolution, 1004
setPreferredColorImageFormat, 1004
setPreferredGrayImageFormat, 1005
setPreferredMonoImageFormat, 1005
setRenderResolution, 1005
setSubsetFonts, 1006
setTargetColorSpace, 1006
setTargetProfile, 1006
JawsMako::IXAMLGenerator::CAnnotationXAML, 47
JawsMako::IXPSInput, 1007
create, 1007
openStreaming, 1008
JawsMako::IXPSOutput, 1009
setColorImageMaxResolution, 1010
setGrayImageMaxResolution, 1011
setJPEGQuality, 1011
setMergeFonts, 1013
setMonoImageMaxResolution, 1013
setPreferredColorImageFormat, 1013
setPreferredGrayImageFormat, 1014
setPreferredMonoImageFormat, 1014
setRenderResolution, 1014
setSubsetFonts, 1015
setTargetColorSpace, 1015
setTargetProfile, 1015
jawsmako.h, 1074
eFileFormat, 1075
key
JawsMako::IPJLParser::CPjlAttributeValue, 53
length
IRAStream, 931
loadFont
IEDLFontSystemFont, 723
loadFromFile
IDOMJobTkContent, 446
loadFromInitString
IDOMJobTkContent, 447
loadFromStream
IDOMJobTkContent, 447
makeNodeOptional
JawsMako::IOptionalContent, 833
makeNodeOptionalWithNewGroup
JawsMako::IOptionalContent, 834
matchesReference
JawsMako::IAnnotation, 149
memutils.h, 1075
modifier
JawsMako::IPJLParser::CPjlAttributeValue, 54
nextEnumerationItem
IDOMGlyphIDEnumerator, 353
IFontHeaderWriteSegmentBlockEnumerator, 745
NormalizeString
pjl.h, 1080
platform.h, 1081
platform_utils.h, 1081
ediExclusiveMakeTempDir, 1082
ediFopen, 1082
ediGetProcessId, 1084
ediMakeTempDir, 1084
ediMakeTempDirProvidingSubDirPath, 1084
ediMkdir, 1085
ediRmdir, 1085
ediSnprintf, 1086
ediSnprintfE, 1086
ediVsnprintf, 1087
postMul
CTransformMatrix, 64
preMul
CTransformMatrix, 64
psoutput.h, 1087
pushBack
IPushbackStream, 926
pushFilter
IDOMFilteredImage, 281
pushTransform
JawsMako::ITransformChain, 989
pushTransformFront
INDEX

JawsMako::ITransformChain, 989
put
IDOMResourceDictionary, 621

read
IInputStream, 804
readScanLine
IImageFrame, 782
recommendedVisibilityForCategories
JawsMako::IOptionalContentGroupUsage, 845
registerClass
IEDLClassFactory, 717
registerNamedClass
IEDLClassFactory, 717
registerNumbers
IDOMCatalog, 218
registerObject
IDOMCatalog, 220
release
JawsMako::IPage, 870
removeAnnotation
JawsMako::IPage, 870
removeChildField
JawsMako::IFormField, 760
removeChildWidget
JawsMako::IFormField, 761
RemoveCombinationCharacters
types.h, 1097
removeDocument
JawsMako::IDocumentAssembly, 189
removeField
JawsMako::IForm, 754
removePage
JawsMako::IDocument, 184
removeProperty
IDOMMetadata, 482
IDOMNode, 496
removeTransform
JawsMako::ITransformChain, 990
removeWidget
JawsMako::IForm, 755
render
JawsMako::IJawsRenderer, 810
renderAntiAliased
JawsMako::IJawsRenderer, 811
renderAntiAliasedToFrameBuffer
JawsMako::IJawsRenderer, 811
renderMonochrome
JawsMako::IJawsRenderer, 812
renderMonochromeToFrameBuffer
JawsMako::IJawsRenderer, 812
renderSeparations
JawsMako::IJawsRenderer, 813
renderToFrameBuffer
JawsMako::IJawsRenderer, 814
renderToFrameBufferPadAndReverse
JawsMako::IJawsRenderer, 815
replaceChild
IDOMNode, 497
ReplaceNonListedLigatures
types.h, 1097
revert
JawsMako::IPage, 871
rotate
CTransformMatrix, 64
JawsMako::IAnnotation, 149
run
IRunnable, 945
scale
CTransformMatrix, 65
scanPdfForFonts
JawsMako::IPDFInput, 893
set
CTransformMatrix, 65
IDOMNodeFlags, 501
setAllowMaskedResults
JawsMako::IImageMergerTransform, 788
setAllowRestrictedFonts
JawsMako::IPDFOutput, 900
setAllowedPermissionsFlags
JawsMako::IOutput, 853
setAlpha
IDOMColor, 231
setAlwaysEmbedFonts
JawsMako::IPDFOutput, 900
setAntiAlias
IDOMShadingPatternBrush, 631
setAuthor
JawsMako::IMarkupAnnotation, 827
setAutoRotatePages
JawsMako::IPDFOutput, 900
setAutomationPropertiesHelpText
IDOMCanvas, 211
IDOMPathNode, 570
setAutomationPropertiesName
IDOMCanvas, 212
IDOMPathNode, 570
setBBox
IDOMShadingPatternBrush, 632
IDOMTilingPatternBrush, 687
setBackgroundColor
IDOMShadingPatternBrush, 631
setBidiLevel
IDOMGlyphs, 368
setBitsPerComponent
IDOMShadingPatternType4567Brush, 653
setBitsPerCoordinate
IDOMShadingPatternType4567Brush, 654
setBitsPerFlag
IDOMShadingPatternType4567Brush, 654
setBleedBox
IDOMFixedPage, 290
setBlendMode
IDOMCharPathGroup, 224
IDOMFormInstance, 335
IDOMGlyphs, 368
IDOMPathNode, 571

Generated by Doxygen
IDOMTransparencyGroup, 697
setBoolProperty
IDOMImageProperties, 430
setBorder
JawsMako::IAnnotation, 149
setBottom
IDOMPageRectTarget, 534
setBounds
IDOMForm, 331
setBrush
IDOMMaskedBrush, 475
setCaretStops
IDOMGlyphs, 369
setCenter
IDOMRadialGradientBrush, 608
setCharPathType
IDOMCharPathGroup, 224
setCharacterData
IDOMJobTkGenericCharacterData, 451
setClip
IDOMGlyphs, 369
IDOMGroup, 387
IDOMPathNode, 571
setClippedGroup
IDOMCharPathGroup, 226
setColor
IDOMGradientStop, 383
IDOMSolidColorBrush, 669
JawsMako::IAnnotation, 150
setColorImageMaxResolution
JawsMako::IPDFOutput, 901
JawsMako::IXAMLGenerator, 1003
JawsMako::IXPSOutput, 1010
setColorInterpolationMode
IDOMGradientBrush, 380
setColorSpace
IDOMColor, 231, 232
IDOMShadingPatternBrush, 632
IDOMTransparencyGroup, 697
setColorTIFFCompression
JawsMako::IImageEncoderTransform, 774
setColored
IDOMGlyph, 350
setComponentValue
IDOMColor, 233
setCompressObjects
JawsMako::IPDFOutput, 901
setCompressPages
JawsMako::IPDFOutput, 902
setContent
IDOMJobTk, 437
setContentBox
IDOMFixedPage, 290
setContentPixel
JawsMako::IAnnotation, 150
setColorImageMaxResolution
JawsMako::IPDFOutput, 901
setConvertGray
JawsMako::IPDFOutput, 902
setCoordinates
IDOMShadingPatternType2Brush, 641
IDOMShadingPatternType3Brush, 647
setCreationTime
JawsMako::IMarkupAnnotation, 827
setCropBox
IDOMFixedPage, 291
setDOMId
IDOMNode, 497
setDataSource
IDOMShadingPatternType45678Brush, 654
setDay
IEDLTime, 741
setDecode
IDOMShadingPatternType45678Brush, 656
setDefaultConfiguration
JawsMako::IOptionalContent, 834
setDefaultCopies
JawsMako::IPCL5Input, 879
JawsMako::IPCLXLInput, 887
setDefaultDuplex
JawsMako::IPCL5Input, 879
JawsMako::IPCLXLInput, 887
setDefaultDuplexBindingMode
JawsMako::IPCL5Input, 880
JawsMako::IPCLXLInput, 887
setDefaultLandscape
JawsMako::IPCL5Input, 880
JawsMako::IPCLXLInput, 888
setDefaultManualFeed
JawsMako::IPCL5Input, 880
JawsMako::IPCLXLInput, 888
setDefaultPaperSize
JawsMako::IPCL5Input, 880
JawsMako::IPCLXLInput, 888
setDefaultFromPjl
JawsMako::IPCL5Input, 881
JawsMako::IPCLXLInput, 889
setDescription
IDOMOutlineEntry, 525
setDeviceCMYKIntercept
IColorManager, 175
setDeviceFontName
IDOMGlyphs, 370
setDeviceGrayIntercept
IColorManager, 175
setDeviceRGBIntercept
IColorManager, 175
setDomain
IDOMShadingPatternType1Brush, 636
IDOMShadingPatternType2Brush, 642
IDOMShadingPatternType3Brush, 647
setDownsampleMaskedImages
INDEX

JawsMako::IImageDownsamplerTransform, 770
JawsMako::IPDFOutput, 902

setDX
  CTransformMatrix, 66
setDY
  CTransformMatrix, 66

setEdgeMode
  IDOMCanvas, 212
  IDOMPathNode, 571

setEmbedBase14Fonts
  JawsMako::IPDFOutput, 903

setEmbedFonts
  JawsMako::IPDFOutput, 903

setEmbedded
  IDOMFontOpenType, 305

setCallback
  JawsMako::IPCL5Output, 883
  JawsMako::IPCLXLOutput, 891

setEnableImageDownsampling
  JawsMako::ISVGGenerator, 972

setEncrypt
  JawsMako::IPDFOutput, 904

setEndPoint
  IDOMLinearGradientBrush, 470

setExpanded
  IDOMOutlineEntry, 525

setExtend
  IDOMShadingPatternType2Brush, 642
  IDOMShadingPatternType3Brush, 647

setFactory
  ISession, 949

setFailOnFontFallback
  JawsMako::IPDFInput, 894

setFile
  IDOMActionLaunch, 195

setFill
  IDOMCharPathGroup, 226
  IDOMGlyphs, 370
  IDOMPathNode, 572

setFillRule
  IDOMPathGeometry, 550

setFitType
  IDOMPageRectTarget, 534

setFlags
  JawsMako::IAnnotation, 150

setFont
  IDOMGlyphs, 370

setFontPath
  IEDLFontSystemFont, 723

setFontObject
  IDOMGlyphs, 371

setFontRenderingEmSize
  IDOMGlyphs, 371

setFontSource
  IDOMFont, 296
IDOMGlyphs, 372
setStatus
IDOMPathSegment, 581
setJPEGQuality
JawsMako::IImageEncoderTransform, 775
JawsMako::IPDFOutput, 905
JawsMako::IXMLGenerator, 1003
JawsMako::IXPSOutput, 1011
setJobTicket
IDOMJobTkOwner, 462
setJobTkNodeType
IDOMJobTkNode, 459
setLanguage
IDOMCanvas, 212
IDOMFixedPage, 291
IDOMGlyphs, 373
IDOMOutline, 519
IDOMOutlineEntry, 525
IDOMPathNode, 572
JawsMako::IOptionalContentGroupUsage, 845
setLeft
IDOMPageRectTarget, 535
setLevel
IDOMJobTkContent, 447
setLineEndpoints
JawsMako::ILineAnnotation, 820
setMacParameters
IDOMActionLaunch, 195
setMapDeviceGrayToCMYKBlack
IColorManager, 176
setMarkedContentDetails
IDOMGroup, 387
setMatrix
IDOMForm, 331
IDOMShadingPatternType1Brush, 637
setMediaSource
JawsMako::IPCL5Output, 884
setMergeFonts
JawsMako::IXMLGenerator, 1004
JawsMako::IXPSOutput, 1013
setModificationTime
JawsMako::IAnnotation, 151
setModified
IDOMJobTkContent, 448
setMonoImageMaxResolution
JawsMako::IPDFOutput, 906
JawsMako::IXMLGenerator, 1004
JawsMako::IXPSOutput, 1013
setMonoTIFFCompression
JawsMako::IImageEncoderTransform, 775
setMonochromeMode
JawsMako::IRendererTransform, 943
setMonth
IEDLTime, 741
setName
EDLQName, 140
JawsMako::IStampAnnotation, 957
setNamedDestinations
JawsMako::IDocument, 184
setNameSpace
EDLQName, 141
IEDLNamespace, 726
setNavigateLink
IDOMCanvas, 213
IDOMGlyphs, 373
IDOMPathNode, 573
setNeedAppearances
JawsMako::IForm, 755
setNeverEmbedFonts
JawsMako::IPDFOutput, 906
setNewWindow
IDOMActionLaunch, 195
setNextSibling
IDOMNode, 497
setObjectProperty
IDOMImageProperties, 431
setOffset
IDOMGradientStop, 383
setOpacity
IDOMBrush, 206
IDOMFormInstance, 337
IDOMGlyphs, 373
IDOMPathNode, 573
IDOMTransparencyGroup, 698
JawsMako::IMarkupAnnotation, 827
setOpacityMask
IDOMFormInstance, 337
IDOMGlyphs, 374
IDOMPathNode, 573
IDOMTransparencyGroup, 699
setOpen
JawsMako::IPopupAnnotation, 918
JawsMako::ITextAnnotation, 975
setOpenStream
JawsMako::IPCL5Output, 884
JawsMako::IPCLXLOutput, 891
setOptionalContentDetails
IDOMGroup, 388
setOriginX
IDOMGlyphs, 374
setOriginY
IDOMGlyphs, 374
setOutline
IDOMGlyphs, 374
setOutline
JawsMako::IDocument, 184
setOutputIntent
JawsMako::IPDFOutput, 906
setOwner
IDOMJobTk, 437
setPageElement
JawsMako::IOptionalContentGroupUsage, 846
setPageGroup
IDOMFixedPage, 292
setPageId
IDOMPageRectTarget, 535
setPageResolutionCallback
JawsMako::ISVGGenerator, 972
setPaintType
IDOMTilingPatternBrush, 687
setParentNode
IDOMNode, 498
setPartialName
JawsMako::IFormField, 761
JawsMako::IWidgetAnnotation, 996
setPassword
JawsMako::IPDFInput, 894
setPathData
IDOMPathNode, 574
setPatternColor
IDOMTilingPatternBrush, 687
setPdfXDeviceNErrorHandling
JawsMako::IPDFOutput, 907
setPdfXExtendedGraphicsStateErrorHandling
JawsMako::IPDFOutput, 907
setPdfXOptionalContentErrorHandling
JawsMako::IPDFOutput, 907
setPoint
IDOMArcSegment, 201
setPoints
JawsMako::IPolyAnnotation, 916
setPopup
JawsMako::IMarkupAnnotation, 828
setPos
IRAStream, 931
setPreferredColorFormat
JawsMako::IImageEncoderTransform, 775
setPreferredColorImageCompression
JawsMako::IPDFOutput, 907
setPreferredColorImageFormat
JawsMako::IXAMLGenerator, 1004
JawsMako::IXPSOutput, 1013
setPreferredFormat
JawsMako::IImageEncoderTransform, 775
JawsMako::IImageEncoderTransform, 775
setPreferredGrayFormat
JawsMako::IImageEncoderTransform, 776
setPreferredGrayImageCompression
JawsMako::IPDFOutput, 908
setPreferredGrayImageFormat
JawsMako::IXAMLGenerator, 1005
JawsMako::IXPSOutput, 1014
setPreferredMonoFormat
JawsMako::IImageEncoderTransform, 776
JawsMako::IPDFOutput, 908
setPreferredMonoImageCompression
JawsMako::IPDFOutput, 908
JawsMako::IPDFOutput, 908
JawsMako::IXAMLGenerator, 1005
JawsMako::IXPSOutput, 1014
setPreferredRenderedImageCompression
JawsMako::IPDFOutput, 908
setPrefix
IEDLNamespace, 726
setPreviousSibling
IDOMNode, 498
setProducer
JawsMako::IPDFOutput, 909
setProperty
IDOMImageProperties, 431
IDOMMetadata, 483
IDOMNode, 498
setQName
IDOMJobTokenizer, 460
setQuadPoints
JawsMako::ILinkAnnotation, 821
JawsMako::IRedactionAnnotation, 937
JawsMako::ITextMarkupAnnotation, 976
setRadiusX
IDOMArcSegment, 202
IDOMRadialGradientBrush, 608
setRadiusY
IDOMArcSegment, 202
IDOMRadialGradientBrush, 609
setRect
JawsMako::IAnnotation, 151
setReencodeImages
JawsMako::IPDFOutput, 909
setRenderResolution
JawsMako::IPDFOutput, 909
JawsMako::IXAMLGenerator, 1005
JawsMako::IXPSOutput, 1014
setRenderTransform
IDOMFormInstance, 337
IDOMGlyphs, 375
IDOMGroup, 388
IDOMMatrix, 477
IDOMPathGeometry, 551
IDOMPathNode, 574
IDOMTilingPatternBrush, 689
IDOMTransformableBrush, 692
setRenderTransparentNodesOnPageGroupMismatch
JawsMako::IRendererTransform, 943
setResolution
JawsMako::IPCL5Output, 884
setResourceId
IDOMCanvas, 213
IDOMFixedPage, 292
setRight
IDOMPagingRectTarget, 535
setRopResolution
JawsMako::IPCL5Input, 881
JawsMako::IPCLXLInput, 889
setRotationAngle
IDOMArcSegment, 203
setSessionTempRootDirectory
isession.h, 1073
setShadingType
IDOMShadingPatternType4567Brush, 656
setShouldZeroWidthLinesBeVisible
IDOMPathNode, 574
setSimulateBlackDeviceGrayTextOverprint
JawsMako::IOverprintSimulationTransform, 867
setSnapsToDevicePixels
IDOMPathNode, 575
setSpotHalftone
<table>
<thead>
<tr>
<th>Function/Method</th>
<th>Source File</th>
<th>Line Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>JawsMako::IPDFOutput</td>
<td>911</td>
<td></td>
</tr>
<tr>
<td>setVerticesPerRow</td>
<td>DOMShadingPatternType4567Brush</td>
<td>657</td>
</tr>
<tr>
<td>setViewBox</td>
<td>IDOMImageBrush</td>
<td>412</td>
</tr>
<tr>
<td></td>
<td>IDOMVisualBrush</td>
<td>710</td>
</tr>
<tr>
<td>setViewBoxUnits</td>
<td>IDOMImageBrush</td>
<td>412</td>
</tr>
<tr>
<td></td>
<td>IDOMVisualBrush</td>
<td>710</td>
</tr>
<tr>
<td>setViewPort</td>
<td>IDOMImageBrush</td>
<td>413</td>
</tr>
<tr>
<td></td>
<td>IDOMVisualBrush</td>
<td>711</td>
</tr>
<tr>
<td>setViewPortUnits</td>
<td>IDOMImageBrush</td>
<td>413</td>
</tr>
<tr>
<td></td>
<td>IDOMVisualBrush</td>
<td>711</td>
</tr>
<tr>
<td>setVisual</td>
<td>IDOMTilingPatternBrush</td>
<td>690</td>
</tr>
<tr>
<td></td>
<td>IDOMVisualBrush</td>
<td>712</td>
</tr>
<tr>
<td>setWidth</td>
<td>IDOMFixedPage</td>
<td>293</td>
</tr>
<tr>
<td>setWinParameters</td>
<td>IDOMActionLaunch</td>
<td>196</td>
</tr>
<tr>
<td>setXX</td>
<td>CTransformMatrix</td>
<td>66</td>
</tr>
<tr>
<td>setXY</td>
<td>CTransformMatrix</td>
<td>66</td>
</tr>
<tr>
<td>setYear</td>
<td>IEDLTime</td>
<td>742</td>
</tr>
<tr>
<td>setYX</td>
<td>CTransformMatrix</td>
<td>67</td>
</tr>
<tr>
<td>setYY</td>
<td>CTransformMatrix</td>
<td>67</td>
</tr>
<tr>
<td>setZoom</td>
<td>IDOMPageRectTarget</td>
<td>536</td>
</tr>
<tr>
<td>setupPDFOutput</td>
<td>JawsMako::IStrokerTransform</td>
<td>960</td>
</tr>
<tr>
<td>setupPDFStyleOutput</td>
<td>JawsMako::IStrokerTransform</td>
<td>960</td>
</tr>
<tr>
<td>SignatureID</td>
<td>1017</td>
<td></td>
</tr>
<tr>
<td>similar</td>
<td>IDOMColorSpace</td>
<td>237</td>
</tr>
<tr>
<td>skiarenderer.h</td>
<td>1088</td>
<td></td>
</tr>
<tr>
<td>skip</td>
<td>InputStream</td>
<td>804</td>
</tr>
<tr>
<td></td>
<td>skipScanLines</td>
<td>783</td>
</tr>
<tr>
<td>smartptr.h</td>
<td>1088</td>
<td></td>
</tr>
<tr>
<td>split</td>
<td>IDOMGlyphs</td>
<td>376</td>
</tr>
<tr>
<td></td>
<td>IDOMPathNode</td>
<td>579</td>
</tr>
<tr>
<td>stateInsideBrush</td>
<td>JawsMako::CTransformState</td>
<td>70</td>
</tr>
<tr>
<td>stateInsideNode</td>
<td>JawsMako::CTransformState</td>
<td>71</td>
</tr>
<tr>
<td>StringToU16String</td>
<td>types.h</td>
<td>1097</td>
</tr>
<tr>
<td>StringToU32String</td>
<td>types.h</td>
<td>1098</td>
</tr>
<tr>
<td>StringToU8String</td>
<td>types.h</td>
<td>1098</td>
</tr>
<tr>
<td>structure.h</td>
<td>1098</td>
<td></td>
</tr>
<tr>
<td>svggenerator.h</td>
<td>1089</td>
<td></td>
</tr>
<tr>
<td>SweepDirection</td>
<td>DOMArcSegment</td>
<td>198</td>
</tr>
<tr>
<td>sxtof</td>
<td>IDOMImageBrush</td>
<td>412</td>
</tr>
<tr>
<td></td>
<td>IDOMVisualBrush</td>
<td>710</td>
</tr>
<tr>
<td>tagNode</td>
<td>JawsMako::IStructure</td>
<td>961</td>
</tr>
<tr>
<td>text.h</td>
<td>1090</td>
<td></td>
</tr>
<tr>
<td>ePageAnalysis</td>
<td>1091</td>
<td></td>
</tr>
<tr>
<td>ePageLayoutType</td>
<td>1091</td>
<td></td>
</tr>
<tr>
<td>IPDFLayoutNodeCollection</td>
<td>1090</td>
<td></td>
</tr>
<tr>
<td>throwEDLError</td>
<td>edlerrors.h</td>
<td>1027</td>
</tr>
<tr>
<td>toPDFDate</td>
<td>IEDLTime</td>
<td>742</td>
</tr>
<tr>
<td>toW3CDTF</td>
<td>IEDLTime</td>
<td>742</td>
</tr>
<tr>
<td>transform</td>
<td>CTransformMatrix</td>
<td>67</td>
</tr>
<tr>
<td></td>
<td>JawsMako::ITransform</td>
<td>984–986</td>
</tr>
<tr>
<td></td>
<td>JawsMako::ITransformChain</td>
<td>990</td>
</tr>
<tr>
<td>transformPage</td>
<td>JawsMako::ITransform</td>
<td>987</td>
</tr>
<tr>
<td></td>
<td>JawsMako::ITransformChain</td>
<td>991</td>
</tr>
<tr>
<td>transformPriv</td>
<td>JawsMako::CTransformState</td>
<td>71</td>
</tr>
<tr>
<td>transformRect</td>
<td>CTransformMatrix</td>
<td>68</td>
</tr>
<tr>
<td>transforms.h</td>
<td>1091</td>
<td></td>
</tr>
<tr>
<td>translate</td>
<td>CTransformMatrix</td>
<td>68</td>
</tr>
<tr>
<td>types.h</td>
<td>1093</td>
<td></td>
</tr>
<tr>
<td>eNormalizationForm</td>
<td>1095</td>
<td></td>
</tr>
<tr>
<td>eOptionalContentEvent</td>
<td>1095</td>
<td></td>
</tr>
<tr>
<td>IsCombinationChar</td>
<td>1095</td>
<td></td>
</tr>
<tr>
<td>NormalizeString</td>
<td>1096</td>
<td></td>
</tr>
<tr>
<td>NormalizeStringForCompare</td>
<td>1096</td>
<td></td>
</tr>
<tr>
<td>RemoveCombinationCharacters</td>
<td>1097</td>
<td></td>
</tr>
<tr>
<td>ReplaceNonListedLigatures</td>
<td>1097</td>
<td></td>
</tr>
<tr>
<td>StringToU16String</td>
<td>1097</td>
<td></td>
</tr>
<tr>
<td>StringToU32String</td>
<td>1098</td>
<td></td>
</tr>
<tr>
<td>StringToU8String</td>
<td>1098</td>
<td></td>
</tr>
<tr>
<td>U16StringToString</td>
<td>1098</td>
<td></td>
</tr>
<tr>
<td>U16StringToString</td>
<td>1098</td>
<td></td>
</tr>
<tr>
<td>U16StringToU32String</td>
<td>1099</td>
<td></td>
</tr>
<tr>
<td>U16StringToU8String</td>
<td>1099</td>
<td></td>
</tr>
<tr>
<td>U32StringToString</td>
<td>1099</td>
<td></td>
</tr>
<tr>
<td>U32StringToU16String</td>
<td>1101</td>
<td></td>
</tr>
<tr>
<td>U32StringToU8String</td>
<td>1101</td>
<td></td>
</tr>
<tr>
<td>U8StringToString</td>
<td>1101</td>
<td></td>
</tr>
<tr>
<td>U8StringToU16String</td>
<td>1103</td>
<td></td>
</tr>
<tr>
<td>U8StringToU32String</td>
<td>1103</td>
<td></td>
</tr>
</tbody>
</table>

Generated by Doxygen
types.h, 1099
U16StringToU8String
U32StringToString
U32StringToU16String
U32StringToU8String
U8StringToString
U8StringToU16String
U8StringToU32String

unite
IDOMShape, 663
unregisterObject
IDOMCatalog, 220
UserRAReadFunc
edlstream.h, 1033
UserStreamReadFunc
edlstream.h, 1033
UserStreamWriteFunc
edlstream.h, 1034

value
JawsMako::IPJLParser::CPjlAttributeValue, 54

walkTree
IDOMNode, 499
widgetInSubtree
JawsMako::IFormField, 761, 762
widgetInTree
JawsMako::IForm, 755, 756
width
JawsMako::CAnnotationBorder, 47
write
IOutputStream, 863
writeAssembly
JawsMako::IOutput, 854, 855
writeEnumerationItemBlock
IFontPCL5WriteSegmentBlockEnumerator, 749
writeFormatted
IOutputStream, 863
writePage
JawsMako::IOutputWriter, 865
writeScanLine
IImageFrameReader, 785
writeSegmentBlock
IFontHeaderWriteSegmentBlockEnumerator, 746
writeStream
IOutputStream, 864

xamlgenerator.h, 1103
xpsinput.h, 1104
xpsoutput.h, 1104
xx
CTransformMatrix, 68

xy
CTransformMatrix, 68
yx
CTransformMatrix, 69
yy
CTransformMatrix, 69

Generated by Doxygen