

dxflib

Generated by Doxygen 1.9.1



<b>1 Todo List</b>	<b>1</b>
<b>2 Hierarchical Index</b>	<b>3</b>
2.1 Class Hierarchy	3
<b>3 Class Index</b>	<b>5</b>
3.1 Class List	5
<b>4 Class Documentation</b>	<b>9</b>
4.1 DL_ArcAlignedTextData Struct Reference	9
4.1.1 Detailed Description	10
4.1.2 Member Data Documentation	10
4.1.2.1 alignment	10
4.1.2.2 arcHandle	10
4.1.2.3 bold	10
4.1.2.4 characerSet	10
4.1.2.5 cx	11
4.1.2.6 cy	11
4.1.2.7 cz	11
4.1.2.8 direction	11
4.1.2.9 endAngle	11
4.1.2.10 font	12
4.1.2.11 height	12
4.1.2.12 italic	12
4.1.2.13 leftOffset	12
4.1.2.14 offset	12
4.1.2.15 pitch	13
4.1.2.16 radius	13
4.1.2.17 reversedCharacterOrder	13
4.1.2.18 rightOffset	13
4.1.2.19 shxFont	13
4.1.2.20 side	14
4.1.2.21 spacing	14
4.1.2.22 startAngle	14
4.1.2.23 style	14
4.1.2.24 text	14
4.1.2.25 underline	15
4.1.2.26 wizard	15
4.1.2.27 xScaleFactor	15
4.2 DL_ArcData Struct Reference	15
4.2.1 Detailed Description	16
4.2.2 Constructor & Destructor Documentation	16
4.2.2.1 DL_ArcData()	16

4.2.3 Member Data Documentation	16
4.2.3.1 angle1	16
4.2.3.2 angle2	16
4.2.3.3 cx	17
4.2.3.4 cy	17
4.2.3.5 cz	17
4.2.3.6 radius	17
4.3 DL_AttributeData Struct Reference	17
4.3.1 Detailed Description	18
4.3.2 Constructor & Destructor Documentation	18
4.3.2.1 DL_AttributeData()	18
4.3.3 Member Data Documentation	18
4.3.3.1 tag	19
4.4 DL_Attributes Class Reference	19
4.4.1 Detailed Description	20
4.4.2 Constructor & Destructor Documentation	20
4.4.2.1 DL_Attributes() [1/2]	20
4.4.2.2 DL_Attributes() [2/2]	20
4.4.3 Member Function Documentation	21
4.4.3.1 getColor()	21
4.4.3.2 getColor24()	21
4.4.3.3 getLayer()	22
4.4.3.4 getLinetype()	22
4.4.3.5 getWidth()	22
4.4.3.6 setColor()	22
4.4.3.7 setColor24()	23
4.4.3.8 setLayer()	23
4.4.3.9 setLinetype()	23
4.5 DL_BlockData Struct Reference	23
4.5.1 Detailed Description	24
4.5.2 Constructor & Destructor Documentation	24
4.5.2.1 DL_BlockData()	24
4.5.3 Member Data Documentation	24
4.5.3.1 flags	24
4.6 DL_CircleData Struct Reference	25
4.6.1 Detailed Description	25
4.6.2 Constructor & Destructor Documentation	25
4.6.2.1 DL_CircleData()	25
4.6.3 Member Data Documentation	25
4.6.3.1 cx	26
4.6.3.2 cy	26
4.6.3.3 cz	26

4.6.3.4 radius . . . . .	26
4.7 DL_Codes Class Reference . . . . .	26
4.7.1 Detailed Description . . . . .	27
4.8 DL_ControlPointData Struct Reference . . . . .	27
4.8.1 Detailed Description . . . . .	27
4.8.2 Constructor & Destructor Documentation . . . . .	27
4.8.2.1 DL_ControlPointData() . . . . .	28
4.8.3 Member Data Documentation . . . . .	28
4.8.3.1 w . . . . .	28
4.8.3.2 x . . . . .	28
4.8.3.3 y . . . . .	28
4.8.3.4 z . . . . .	28
4.9 DL_CreationAdapter Class Reference . . . . .	29
4.9.1 Detailed Description . . . . .	31
4.9.2 Member Function Documentation . . . . .	32
4.9.2.1 addBlock() . . . . .	32
4.9.2.2 addMTextChunk() . . . . .	32
4.9.2.3 endEntity() . . . . .	32
4.9.2.4 processCodeValuePair() . . . . .	33
4.9.2.5 setVariableDouble() . . . . .	33
4.9.2.6 setVariableInt() . . . . .	33
4.9.2.7 setVariableString() . . . . .	33
4.9.2.8 setVariableVector() . . . . .	34
4.10 DL_CreationInterface Class Reference . . . . .	34
4.10.1 Detailed Description . . . . .	37
4.10.2 Member Function Documentation . . . . .	37
4.10.2.1 addBlock() . . . . .	37
4.10.2.2 addMTextChunk() . . . . .	38
4.10.2.3 endEntity() . . . . .	38
4.10.2.4 getAttributes() . . . . .	38
4.10.2.5 getExtrusion() . . . . .	38
4.10.2.6 processCodeValuePair() . . . . .	39
4.10.2.7 setVariableDouble() . . . . .	39
4.10.2.8 setVariableInt() . . . . .	39
4.10.2.9 setVariableString() . . . . .	40
4.10.2.10 setVariableVector() . . . . .	40
4.11 DL_DictionaryData Struct Reference . . . . .	40
4.11.1 Detailed Description . . . . .	41
4.12 DL_DictionaryEntryData Struct Reference . . . . .	41
4.12.1 Detailed Description . . . . .	41
4.13 DL_DimAlignedData Struct Reference . . . . .	41
4.13.1 Detailed Description . . . . .	42

4.13.2 Constructor & Destructor Documentation	42
4.13.2.1 DL_DimAlignedData()	42
4.13.3 Member Data Documentation	42
4.13.3.1 ep <sub>x</sub> 1	42
4.13.3.2 ep <sub>x</sub> 2	43
4.13.3.3 ep <sub>y</sub> 1	43
4.13.3.4 ep <sub>y</sub> 2	43
4.13.3.5 ep <sub>z</sub> 1	43
4.13.3.6 ep <sub>z</sub> 2	43
4.14 DL_DimAngular2LData Struct Reference	44
4.14.1 Detailed Description	44
4.14.2 Constructor & Destructor Documentation	44
4.14.2.1 DL_DimAngular2LData()	44
4.14.3 Member Data Documentation	45
4.14.3.1 dp <sub>x</sub> 1	45
4.14.3.2 dp <sub>x</sub> 2	45
4.14.3.3 dp <sub>x</sub> 3	45
4.14.3.4 dp <sub>x</sub> 4	45
4.14.3.5 dp <sub>y</sub> 1	45
4.14.3.6 dp <sub>y</sub> 2	46
4.14.3.7 dp <sub>y</sub> 3	46
4.14.3.8 dp <sub>y</sub> 4	46
4.14.3.9 dp <sub>z</sub> 1	46
4.14.3.10 dp <sub>z</sub> 2	46
4.14.3.11 dp <sub>z</sub> 3	46
4.14.3.12 dp <sub>z</sub> 4	47
4.15 DL_DimAngular3PData Struct Reference	47
4.15.1 Detailed Description	47
4.15.2 Constructor & Destructor Documentation	47
4.15.2.1 DL_DimAngular3PData()	48
4.15.3 Member Data Documentation	48
4.15.3.1 dp <sub>x</sub> 1	48
4.15.3.2 dp <sub>x</sub> 2	48
4.15.3.3 dp <sub>x</sub> 3	48
4.15.3.4 dp <sub>y</sub> 1	49
4.15.3.5 dp <sub>y</sub> 2	49
4.15.3.6 dp <sub>y</sub> 3	49
4.15.3.7 dp <sub>z</sub> 1	49
4.15.3.8 dp <sub>z</sub> 2	49
4.15.3.9 dp <sub>z</sub> 3	49
4.16 DL_DimDiametricData Struct Reference	50
4.16.1 Detailed Description	50

4.16.2 Constructor & Destructor Documentation	50
4.16.2.1 DL_DimDiametricData()	50
4.16.3 Member Data Documentation	50
4.16.3.1 dpx	51
4.16.3.2 dpy	51
4.16.3.3 dpz	51
4.16.3.4 leader	51
4.17 DL_DimensionData Struct Reference	51
4.17.1 Detailed Description	52
4.17.2 Constructor & Destructor Documentation	52
4.17.2.1 DL_DimensionData()	52
4.17.3 Member Data Documentation	53
4.17.3.1 attachmentPoint	53
4.17.3.2 dpx	53
4.17.3.3 dpy	53
4.17.3.4 dpz	54
4.17.3.5 lineSpacingFactor	54
4.17.3.6 lineSpacingStyle	54
4.17.3.7 mpx	54
4.17.3.8 mpy	55
4.17.3.9 mpz	55
4.17.3.10 style	55
4.17.3.11 text	55
4.17.3.12 type	55
4.18 DL_DimLinearData Struct Reference	56
4.18.1 Detailed Description	56
4.18.2 Constructor & Destructor Documentation	56
4.18.2.1 DL_DimLinearData()	56
4.18.3 Member Data Documentation	57
4.18.3.1 angle	57
4.18.3.2 dpx1	57
4.18.3.3 dpx2	57
4.18.3.4 dpy1	57
4.18.3.5 dpy2	57
4.18.3.6 dpz1	58
4.18.3.7 dpz2	58
4.18.3.8 oblique	58
4.19 DL_DimOrdinateData Struct Reference	58
4.19.1 Detailed Description	58
4.19.2 Constructor & Destructor Documentation	59
4.19.2.1 DL_DimOrdinateData()	59
4.19.3 Member Data Documentation	59

4.19.3.1 dpx1	59
4.19.3.2 dpx2	59
4.19.3.3 dpy1	59
4.19.3.4 dpy2	60
4.19.3.5 dpz1	60
4.19.3.6 dpz2	60
4.19.3.7 xtype	60
4.20 DL_DimRadialData Struct Reference	60
4.20.1 Detailed Description	61
4.20.2 Constructor & Destructor Documentation	61
4.20.2.1 DL_DimRadialData()	61
4.20.3 Member Data Documentation	61
4.20.3.1 dpx	61
4.20.3.2 dpy	61
4.20.3.3 dpz	62
4.20.3.4 leader	62
4.21 DL_Dxf Class Reference	62
4.21.1 Detailed Description	67
4.21.2 Member Function Documentation	68
4.21.2.1 addAttribute()	68
4.21.2.2 addSolid()	68
4.21.2.3 addTrace()	68
4.21.2.4 checkVariable()	69
4.21.2.5 getDimData()	69
4.21.2.6 getLibVersion()	69
4.21.2.7 getStrippedLine()	69
4.21.2.8 in() [1/2]	70
4.21.2.9 in() [2/2]	70
4.21.2.10 out()	71
4.21.2.11 processDXFGroup()	71
4.21.2.12 readDxfGroups()	72
4.21.2.13 stripWhiteSpace()	73
4.21.2.14 test()	73
4.21.2.15 write3dFace()	73
4.21.2.16 writeAppid()	74
4.21.2.17 writeArc()	74
4.21.2.18 writeBlockRecord()	74
4.21.2.19 writeCircle()	75
4.21.2.20 writeControlPoint()	75
4.21.2.21 writeDimAligned()	75
4.21.2.22 writeDimAngular2L()	76
4.21.2.23 writeDimAngular3P()	76



4.21.2.24 writeDimDiametric()	77
4.21.2.25 writeDimLinear()	77
4.21.2.26 writeDimOrdinate()	78
4.21.2.27 writeDimRadial()	78
4.21.2.28 writeDimStyle()	79
4.21.2.29 writeEllipse()	79
4.21.2.30 writeEndBlock()	79
4.21.2.31 writeFitPoint()	80
4.21.2.32 writeHatch1()	80
4.21.2.33 writeHatch2()	81
4.21.2.34 writeHatchEdge()	81
4.21.2.35 writeHatchLoop1()	81
4.21.2.36 writeHatchLoop2()	82
4.21.2.37 writeImage()	82
4.21.2.38 writeInsert()	83
4.21.2.39 writeKnot()	84
4.21.2.40 writeLayer()	84
4.21.2.41 writeLeader()	85
4.21.2.42 writeLeaderVertex()	85
4.21.2.43 writeLine()	85
4.21.2.44 writeLinetype()	86
4.21.2.45 writeMText()	86
4.21.2.46 writeObjects()	87
4.21.2.47 writeObjectsEnd()	87
4.21.2.48 writePoint()	87
4.21.2.49 writePolyline()	88
4.21.2.50 writePolylineEnd()	88
4.21.2.51 writeRay()	88
4.21.2.52 writeSolid()	89
4.21.2.53 writeSpline()	89
4.21.2.54 writeStyle()	90
4.21.2.55 writeText()	90
4.21.2.56 writeTrace()	90
4.21.2.57 writeUcs()	92
4.21.2.58 writeVertex()	92
4.21.2.59 writeView()	92
4.21.2.60 writeVPort()	93
4.21.2.61 writeXLine()	93
4.22 DL_EllipseData Struct Reference	93
4.22.1 Detailed Description	94
4.22.2 Constructor & Destructor Documentation	94
4.22.2.1 DL_EllipseData()	94

4.22.3 Member Data Documentation	94
4.22.3.1 angle1	95
4.22.3.2 angle2	95
4.22.3.3 cx	95
4.22.3.4 cy	95
4.22.3.5 cz	95
4.22.3.6 mx	96
4.22.3.7 my	96
4.22.3.8 mz	96
4.22.3.9 ratio	96
4.23 DL_Exception Class Reference	96
4.23.1 Detailed Description	97
4.24 DL_Extrusion Class Reference	97
4.24.1 Detailed Description	97
4.24.2 Constructor & Destructor Documentation	97
4.24.2.1 DL_Extrusion()	97
4.24.3 Member Function Documentation	98
4.24.3.1 getDirection() [1/2]	98
4.24.3.2 getDirection() [2/2]	98
4.24.3.3 getElevation()	98
4.25 DL_FitPointData Struct Reference	99
4.25.1 Detailed Description	99
4.25.2 Constructor & Destructor Documentation	99
4.25.2.1 DL_FitPointData()	99
4.25.3 Member Data Documentation	99
4.25.3.1 x	99
4.25.3.2 y	100
4.25.3.3 z	100
4.26 DL_GroupCodeExc Class Reference	100
4.26.1 Detailed Description	100
4.27 DL_HatchData Struct Reference	100
4.27.1 Detailed Description	101
4.27.2 Constructor & Destructor Documentation	101
4.27.2.1 DL_HatchData()	101
4.27.3 Member Data Documentation	101
4.27.3.1 angle	102
4.27.3.2 numLoops	102
4.27.3.3 originX	102
4.27.3.4 pattern	102
4.27.3.5 scale	102
4.27.3.6 solid	103
4.28 DL_HatchEdgeData Struct Reference	103

4.28.1 Detailed Description	104
4.28.2 Constructor & Destructor Documentation	104
4.28.2.1 DL_HatchEdgeData() [1/4]	104
4.28.2.2 DL_HatchEdgeData() [2/4]	104
4.28.2.3 DL_HatchEdgeData() [3/4]	105
4.28.2.4 DL_HatchEdgeData() [4/4]	105
4.28.3 Member Data Documentation	105
4.28.3.1 angle1	105
4.28.3.2 angle2	106
4.28.3.3 ccw	106
4.28.3.4 cx	106
4.28.3.5 cy	106
4.28.3.6 degree	106
4.28.3.7 mx	107
4.28.3.8 my	107
4.28.3.9 nControl	107
4.28.3.10 nFit	107
4.28.3.11 nKnots	107
4.28.3.12 radius	108
4.28.3.13 ratio	108
4.28.3.14 type	108
4.28.3.15 x1	108
4.28.3.16 x2	108
4.28.3.17 y1	109
4.28.3.18 y2	109
4.29 DL_HatchLoopData Struct Reference	109
4.29.1 Detailed Description	109
4.29.2 Constructor & Destructor Documentation	109
4.29.2.1 DL_HatchLoopData()	110
4.29.3 Member Data Documentation	110
4.29.3.1 numEdges	110
4.30 DL_ImageData Struct Reference	110
4.30.1 Detailed Description	111
4.30.2 Constructor & Destructor Documentation	111
4.30.2.1 DL_ImageData()	111
4.30.3 Member Data Documentation	111
4.30.3.1 brightness	111
4.30.3.2 contrast	112
4.30.3.3 fade	112
4.30.3.4 height	112
4.30.3.5 ipx	112
4.30.3.6 ipy	112

4.30.3.7 ipz	113
4.30.3.8 ref	113
4.30.3.9 ux	113
4.30.3.10 uy	113
4.30.3.11 uz	113
4.30.3.12 vx	114
4.30.3.13 vy	114
4.30.3.14 vz	114
4.30.3.15 width	114
4.31 DL_ImageDefData Struct Reference	114
4.31.1 Detailed Description	115
4.31.2 Constructor & Destructor Documentation	115
4.31.2.1 DL_ImageDefData()	115
4.31.3 Member Data Documentation	115
4.31.3.1 file	115
4.31.3.2 ref	115
4.32 DL_InsertData Struct Reference	116
4.32.1 Detailed Description	116
4.32.2 Constructor & Destructor Documentation	116
4.32.2.1 DL_InsertData()	116
4.32.3 Member Data Documentation	117
4.32.3.1 angle	117
4.32.3.2 cols	117
4.32.3.3 colSp	117
4.32.3.4 ipx	117
4.32.3.5 ipy	117
4.32.3.6 ipz	118
4.32.3.7 name	118
4.32.3.8 rows	118
4.32.3.9 rowSp	118
4.32.3.10 sx	118
4.32.3.11 sy	119
4.32.3.12 sz	119
4.33 DL_KnotData Struct Reference	119
4.33.1 Detailed Description	119
4.33.2 Constructor & Destructor Documentation	119
4.33.2.1 DL_KnotData()	120
4.33.3 Member Data Documentation	120
4.33.3.1 k	120
4.34 DL_LayerData Struct Reference	120
4.34.1 Detailed Description	121
4.34.2 Constructor & Destructor Documentation	121

4.34.2.1 DL_LayerData()	121
4.34.3 Member Data Documentation	121
4.34.3.1 flags	121
4.35 DL_LeaderData Struct Reference	121
4.35.1 Detailed Description	122
4.35.2 Constructor & Destructor Documentation	122
4.35.2.1 DL_LeaderData()	122
4.35.3 Member Data Documentation	122
4.35.3.1 arrowHeadFlag	123
4.35.3.2 dimScale	123
4.35.3.3 hooklineDirectionFlag	123
4.35.3.4 hooklineFlag	123
4.35.3.5 leaderCreationFlag	123
4.35.3.6 leaderPathType	124
4.35.3.7 number	124
4.35.3.8 textAnnotationHeight	124
4.35.3.9 textAnnotationWidth	124
4.36 DL_LeaderVertexData Struct Reference	124
4.36.1 Detailed Description	125
4.36.2 Constructor & Destructor Documentation	125
4.36.2.1 DL_LeaderVertexData()	125
4.36.3 Member Data Documentation	125
4.36.3.1 x	125
4.36.3.2 y	126
4.36.3.3 z	126
4.37 DL_LineData Struct Reference	126
4.37.1 Detailed Description	126
4.37.2 Constructor & Destructor Documentation	126
4.37.2.1 DL_LineData()	127
4.37.3 Member Data Documentation	127
4.37.3.1 x1	127
4.37.3.2 x2	127
4.37.3.3 y1	127
4.37.3.4 y2	128
4.37.3.5 z1	128
4.37.3.6 z2	128
4.38 DL_LinetypeData Struct Reference	128
4.38.1 Detailed Description	129
4.38.2 Constructor & Destructor Documentation	129
4.38.2.1 DL_LinetypeData()	129
4.39 DL_MTextData Struct Reference	129
4.39.1 Detailed Description	130

4.39.2 Constructor & Destructor Documentation	130
4.39.2.1 DL_MTextData()	130
4.39.3 Member Data Documentation	131
4.39.3.1 angle	131
4.39.3.2 attachmentPoint	131
4.39.3.3 dirx	131
4.39.3.4 diry	131
4.39.3.5 dirz	132
4.39.3.6 drawingDirection	132
4.39.3.7 height	132
4.39.3.8 ipx	132
4.39.3.9 ipy	132
4.39.3.10 ipz	133
4.39.3.11 lineSpacingFactor	133
4.39.3.12 lineSpacingStyle	133
4.39.3.13 style	133
4.39.3.14 text	133
4.39.3.15 width	134
4.40 DL_NullStrExc Class Reference	134
4.40.1 Detailed Description	134
4.41 DL_PointData Struct Reference	134
4.41.1 Detailed Description	135
4.41.2 Constructor & Destructor Documentation	135
4.41.2.1 DL_PointData()	135
4.41.3 Member Data Documentation	135
4.41.3.1 x	135
4.41.3.2 y	135
4.41.3.3 z	136
4.42 DL_PolylineData Struct Reference	136
4.42.1 Detailed Description	136
4.42.2 Constructor & Destructor Documentation	136
4.42.2.1 DL_PolylineData()	136
4.42.3 Member Data Documentation	137
4.42.3.1 elevation	137
4.42.3.2 flags	137
4.42.3.3 m	137
4.42.3.4 n	137
4.42.3.5 number	137
4.43 DL_RayData Struct Reference	138
4.43.1 Detailed Description	138
4.43.2 Constructor & Destructor Documentation	138
4.43.2.1 DL_RayData()	138

4.43.3 Member Data Documentation	138
4.43.3.1 bx	139
4.43.3.2 by	139
4.43.3.3 bz	139
4.43.3.4 dx	139
4.43.3.5 dy	139
4.43.3.6 dz	140
4.44 DL_SplineData Struct Reference	140
4.44.1 Detailed Description	140
4.44.2 Constructor & Destructor Documentation	140
4.44.2.1 DL_SplineData()	141
4.44.3 Member Data Documentation	141
4.44.3.1 degree	141
4.44.3.2 flags	141
4.44.3.3 nControl	141
4.44.3.4 nFit	142
4.44.3.5 nKnots	142
4.45 DL_StyleData Struct Reference	142
4.45.1 Detailed Description	143
4.46 DL_TextData Struct Reference	143
4.46.1 Detailed Description	144
4.46.2 Constructor & Destructor Documentation	144
4.46.2.1 DL_TextData()	144
4.46.3 Member Data Documentation	144
4.46.3.1 angle	144
4.46.3.2 apx	144
4.46.3.3 apy	145
4.46.3.4 apz	145
4.46.3.5 height	145
4.46.3.6 hJustification	145
4.46.3.7 ipx	145
4.46.3.8 ipy	146
4.46.3.9 ipz	146
4.46.3.10 style	146
4.46.3.11 text	146
4.46.3.12 textGenerationFlags	146
4.46.3.13 vJustification	147
4.46.3.14 xScaleFactor	147
4.47 DL_TraceData Struct Reference	147
4.47.1 Detailed Description	147
4.47.2 Constructor & Destructor Documentation	148
4.47.2.1 DL_TraceData()	148

4.47.3 Member Data Documentation	148
4.47.3.1 thickness	148
4.47.3.2 x	148
4.48 DL_VertexData Struct Reference	149
4.48.1 Detailed Description	149
4.48.2 Constructor & Destructor Documentation	149
4.48.2.1 DL_VertexData()	149
4.48.3 Member Data Documentation	149
4.48.3.1 bulge	150
4.48.3.2 x	150
4.48.3.3 y	150
4.48.3.4 z	150
4.49 DL_Writer Class Reference	150
4.49.1 Detailed Description	152
4.49.2 Constructor & Destructor Documentation	152
4.49.2.1 DL_Writer()	152
4.49.3 Member Function Documentation	153
4.49.3.1 comment()	153
4.49.3.2 dxfBool()	153
4.49.3.3 dxfEOF()	153
4.49.3.4 dxfHex()	154
4.49.3.5 dxfInt()	154
4.49.3.6 dxfReal()	154
4.49.3.7 dxfString() [1/2]	155
4.49.3.8 dxfString() [2/2]	155
4.49.3.9 entity()	155
4.49.3.10 entityAttributes()	156
4.49.3.11 getNextHandle()	156
4.49.3.12 section()	156
4.49.3.13 sectionBlockEntry()	157
4.49.3.14 sectionBlockEntryEnd()	157
4.49.3.15 sectionBlocks()	157
4.49.3.16 sectionClasses()	157
4.49.3.17 sectionEnd()	158
4.49.3.18 sectionEntities()	158
4.49.3.19 sectionHeader()	158
4.49.3.20 sectionObjects()	158
4.49.3.21 sectionTables()	159
4.49.3.22 table()	159
4.49.3.23 tableAppid()	159
4.49.3.24 tableAppidEntry()	160
4.49.3.25 tableEnd()	160



4.49.3.26 tableLayerEntry()	160
4.49.3.27 tableLayers()	160
4.49.3.28 tableLinetypeEntry()	161
4.49.3.29 tableLinetypes()	161
4.49.3.30 tableStyle()	162
4.50 DL_WriterA Class Reference	162
4.50.1 Detailed Description	163
4.50.2 Member Function Documentation	163
4.50.2.1 dxHex()	163
4.50.2.2 dxflnt()	163
4.50.2.3 dxReal()	164
4.50.2.4 dxString() [1/2]	164
4.50.2.5 dxString() [2/2]	165
4.50.2.6 openFailed()	165
4.51 DL_XLineData Struct Reference	166
4.51.1 Detailed Description	166
4.51.2 Constructor & Destructor Documentation	166
4.51.2.1 DL_XLineData()	166
4.51.3 Member Data Documentation	167
4.51.3.1 bx	167
4.51.3.2 by	167
4.51.3.3 bz	167
4.51.3.4 dx	167
4.51.3.5 dy	167
4.51.3.6 dz	167
<b>Index</b>	<b>169</b>



# Chapter 1

## Todo List

**Member `DL_Dxf::addAttribute (DL_CreationInterface *creationInterface)`**

add attrib instead of normal text

**Member `DL_Dxf::getStrippedLine (std::string &s, unsigned int size, FILE *stream, bool stripSpace=true)`**

Change function to use safer FreeBSD strl\* functions

Is it a problem if line is blank (i.e., newline only)? Then, when function returns, (s==NULL).

**Class `DL_Writer`**

Add error checking for string/entry length.

**Class `DL_WriterA`**

What if `fname` is NULL? Or `fname` can't be opened for another reason?



## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

DL_ArcAlignedTextData . . . . .	9
DL_ArcData . . . . .	15
DL_Attributes . . . . .	19
DL_BlockData . . . . .	23
DL_CircleData . . . . .	25
DL_Codes . . . . .	26
DL_ControlPointData . . . . .	27
DL_CreationInterface . . . . .	34
DL_CreationAdapter . . . . .	29
DL_DictionaryData . . . . .	40
DL_DictionaryEntryData . . . . .	41
DL_DimAlignedData . . . . .	41
DL_DimAngular2LData . . . . .	44
DL_DimAngular3PData . . . . .	47
DL_DimDiametricData . . . . .	50
DL_DimensionData . . . . .	51
DL_DimLinearData . . . . .	56
DL_DimOrdinateData . . . . .	58
DL_DimRadialData . . . . .	60
DL_Dxf . . . . .	62
DL_EllipseData . . . . .	93
DL_Exception . . . . .	96
DL_GroupCodeExc . . . . .	100
DL_NullStrExc . . . . .	134
DL_Extrusion . . . . .	97
DL_FitPointData . . . . .	99
DL_HatchData . . . . .	100
DL_HatchEdgeData . . . . .	103
DL_HatchLoopData . . . . .	109
DL_ImageData . . . . .	110
DL_ImageDefData . . . . .	114
DL_InsertData . . . . .	116
DL_KnotData . . . . .	119
DL_LayerData . . . . .	120
DL_LeaderData . . . . .	121

DL_LeaderVertexData . . . . .	124
DL_LineData . . . . .	126
DL_LinetypeData . . . . .	128
DL_MTextData . . . . .	129
DL_PointData . . . . .	134
DL_PolylineData . . . . .	136
DL_RayData . . . . .	138
DL_SplineData . . . . .	140
DL_StyleData . . . . .	142
DL_TextData . . . . .	143
DL_AttributeData . . . . .	17
DL_TraceData . . . . .	147
DL_VertexData . . . . .	149
DL_Writer . . . . .	150
DL_WriterA . . . . .	162
DL_XLineData . . . . .	166

## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">DL_ArcAlignedTextData</a>	
Arc Aligned Text Data . . . . .	9
<a href="#">DL_ArcData</a>	
Arc Data . . . . .	15
<a href="#">DL_AttributeData</a>	
Block attribute data . . . . .	17
<a href="#">DL_Attributes</a>	
Storing and passing around attributes . . . . .	19
<a href="#">DL_BlockData</a>	
Block Data . . . . .	23
<a href="#">DL_CircleData</a>	
Circle Data . . . . .	25
<a href="#">DL_Codes</a>	
Codes for colors and DXF versions . . . . .	26
<a href="#">DL_ControlPointData</a>	
Spline control point data . . . . .	27
<a href="#">DL_CreationAdapter</a>	
An abstract adapter class for receiving DXF events when a DXF file is being read . . . . .	29
<a href="#">DL_CreationInterface</a>	
Abstract class (interface) for the creation of new entities . . . . .	34
<a href="#">DL_DictionaryData</a>	
Dictionary data . . . . .	40
<a href="#">DL_DictionaryEntryData</a>	
Dictionary entry data . . . . .	41
<a href="#">DL_DimAlignedData</a>	
Aligned Dimension Data . . . . .	41
<a href="#">DL_DimAngular2LData</a>	
Angular Dimension Data . . . . .	44
<a href="#">DL_DimAngular3PData</a>	
Angular Dimension Data (3 points version) . . . . .	47
<a href="#">DL_DimDiametricData</a>	
Diametric Dimension Data . . . . .	50
<a href="#">DL_DimensionData</a>	
Generic Dimension Data . . . . .	51
<a href="#">DL_DimLinearData</a>	
Linear (rotated) Dimension Data . . . . .	56

<a href="#">DL_DimOrdinateData</a>	
Ordinate Dimension Data . . . . .	58
<a href="#">DL_DimRadialData</a>	
Radial Dimension Data . . . . .	60
<a href="#">DL_Dxf</a>	
Reading and writing of DXF files . . . . .	62
<a href="#">DL_EllipseData</a>	
Ellipse Data . . . . .	93
<a href="#">DL_Exception</a>	
Used for exception handling . . . . .	96
<a href="#">DL_Extrusion</a>	
Extrusion direction . . . . .	97
<a href="#">DL_FitPointData</a>	
Spline fit point data . . . . .	99
<a href="#">DL_GroupCodeExc</a>	
Used for exception handling . . . . .	100
<a href="#">DL_HatchData</a>	
Hatch data . . . . .	100
<a href="#">DL_HatchEdgeData</a>	
Hatch edge data . . . . .	103
<a href="#">DL_HatchLoopData</a>	
Hatch boundary path (loop) data . . . . .	109
<a href="#">DL_ImageData</a>	
Image Data . . . . .	110
<a href="#">DL_ImageDefData</a>	
Image Definition Data . . . . .	114
<a href="#">DL_InsertData</a>	
Insert Data . . . . .	116
<a href="#">DL_KnotData</a>	
Spline knot data . . . . .	119
<a href="#">DL_LayerData</a>	
Layer Data . . . . .	120
<a href="#">DL_LeaderData</a>	
Leader (arrow) . . . . .	121
<a href="#">DL_LeaderVertexData</a>	
Leader Vertex Data . . . . .	124
<a href="#">DL_LineData</a>	
Line Data . . . . .	126
<a href="#">DL_LinetypeData</a>	
Line Type Data . . . . .	128
<a href="#">DL_MTextData</a>	
MText Data . . . . .	129
<a href="#">DL_NullStrExc</a>	
Used for exception handling . . . . .	134
<a href="#">DL_PointData</a>	
Point Data . . . . .	134
<a href="#">DL_PolylineData</a>	
Polyline Data . . . . .	136
<a href="#">DL_RayData</a>	
Ray Data . . . . .	138
<a href="#">DL_SplineData</a>	
Spline Data . . . . .	140
<a href="#">DL_StyleData</a>	
Text style data . . . . .	142
<a href="#">DL_TextData</a>	
Text Data . . . . .	143
<a href="#">DL_TraceData</a>	
Trace Data / solid data / 3d face data . . . . .	147



<a href="#">DL_VertexData</a>	
Vertex Data . . . . .	149
<a href="#">DL_Writer</a>	
Defines interface for writing low level DXF constructs to a file . . . . .	150
<a href="#">DL_WriterA</a>	
Implements functions defined in <a href="#">DL_Writer</a> for writing low level DXF constructs to an ASCII format DXF file . . . . .	162
<a href="#">DL_XLineData</a>	
XLine Data . . . . .	166



## Chapter 4

# Class Documentation

### 4.1 DL\_ArcAlignedTextData Struct Reference

Arc Aligned Text Data.

```
#include <dl_entities.h>
```

#### Public Attributes

- std::string [text](#)
- std::string [font](#)
- std::string [style](#)
- double [cx](#)
- double [cy](#)
- double [cz](#)
- double [radius](#)
- double [xScaleFactor](#)
- double [height](#)
- double [spacing](#)
- double [offset](#)
- double [rightOffset](#)
- double [leftOffset](#)
- double [startAngle](#)
- double [endAngle](#)
- bool [reversedCharacterOrder](#)
- int [direction](#)
- int [alignment](#)
- int [side](#)
- bool [bold](#)
- bool [italic](#)
- bool [underline](#)
- int [characerSet](#)
- int [pitch](#)
- bool [shxFont](#)
- bool [wizard](#)
- int [arcHandle](#)

### 4.1.1 Detailed Description

Arc Aligned Text Data.

### 4.1.2 Member Data Documentation

#### 4.1.2.1 alignment

```
int DL_ArcAlignedTextData::alignment
```

Alignment: 1: fit 2: left 3: right 4: center

Referenced by DL\_Dxf::addArcAlignedText().

#### 4.1.2.2 arcHandle

```
int DL_ArcAlignedTextData::arcHandle
```

Arc handle/ID

Referenced by DL\_Dxf::addArcAlignedText().

#### 4.1.2.3 bold

```
bool DL_ArcAlignedTextData::bold
```

Bold flag

Referenced by DL\_Dxf::addArcAlignedText().

#### 4.1.2.4 characerSet

```
int DL_ArcAlignedTextData::characerSet
```

Character set value. Windows character set identifier.

Referenced by DL\_Dxf::addArcAlignedText().

#### 4.1.2.5 cx

```
double DL_ArcAlignedTextData::cx
```

X coordinate of arc center point.

Referenced by DL\_Dxf::addArcAlignedText().

#### 4.1.2.6 cy

```
double DL_ArcAlignedTextData::cy
```

Y coordinate of arc center point.

Referenced by DL\_Dxf::addArcAlignedText().

#### 4.1.2.7 cz

```
double DL_ArcAlignedTextData::cz
```

Z coordinate of arc center point.

Referenced by DL\_Dxf::addArcAlignedText().

#### 4.1.2.8 direction

```
int DL_ArcAlignedTextData::direction
```

Direction 1: outward from center 2: inward from center

Referenced by DL\_Dxf::addArcAlignedText().

#### 4.1.2.9 endAngle

```
double DL_ArcAlignedTextData::endAngle
```

End angle (radians)

Referenced by DL\_Dxf::addArcAlignedText().

#### 4.1.2.10 font

```
std::string DL_ArcAlignedTextData::font
```

Font name

Referenced by DL\_Dxf::addArcAlignedText().

#### 4.1.2.11 height

```
double DL_ArcAlignedTextData::height
```

Text height

Referenced by DL\_Dxf::addArcAlignedText().

#### 4.1.2.12 italic

```
bool DL_ArcAlignedTextData::italic
```

Italic flag

Referenced by DL\_Dxf::addArcAlignedText().

#### 4.1.2.13 leftOffset

```
double DL_ArcAlignedTextData::leftOffset
```

Left offset

Referenced by DL\_Dxf::addArcAlignedText().

#### 4.1.2.14 offset

```
double DL_ArcAlignedTextData::offset
```

Offset from arc

Referenced by DL\_Dxf::addArcAlignedText().

#### 4.1.2.15 pitch

```
int DL_ArcAlignedTextData::pitch
```

Pitch and family value. Windows pitch and character family identifier.

Referenced by DL\_Dxf::addArcAlignedText().

#### 4.1.2.16 radius

```
double DL_ArcAlignedTextData::radius
```

Arc radius.

Referenced by DL\_Dxf::addArcAlignedText().

#### 4.1.2.17 reversedCharacterOrder

```
bool DL_ArcAlignedTextData::reversedCharacterOrder
```

Reversed character order: false: normal true: reversed

Referenced by DL\_Dxf::addArcAlignedText().

#### 4.1.2.18 rightOffset

```
double DL_ArcAlignedTextData::rightOffset
```

Right offset

Referenced by DL\_Dxf::addArcAlignedText().

#### 4.1.2.19 shxFont

```
bool DL_ArcAlignedTextData::shxFont
```

Font type: false: TTF true: SHX

Referenced by DL\_Dxf::addArcAlignedText().

#### 4.1.2.20 side

```
int DL_ArcAlignedTextData::side
```

Side 1: convex 2: concave

Referenced by DL\_Dxf::addArcAlignedText().

#### 4.1.2.21 spacing

```
double DL_ArcAlignedTextData::spacing
```

Character spacing

Referenced by DL\_Dxf::addArcAlignedText().

#### 4.1.2.22 startAngle

```
double DL_ArcAlignedTextData::startAngle
```

Start angle (radians)

Referenced by DL\_Dxf::addArcAlignedText().

#### 4.1.2.23 style

```
std::string DL_ArcAlignedTextData::style
```

Style

Referenced by DL\_Dxf::addArcAlignedText().

#### 4.1.2.24 text

```
std::string DL_ArcAlignedTextData::text
```

Text string

Referenced by DL\_Dxf::addArcAlignedText().



#### 4.1.2.25 underline

```
bool DL_ArcAlignedTextData::underline
```

Underline flag

Referenced by DL\_Dxf::addArcAlignedText().

#### 4.1.2.26 wizard

```
bool DL_ArcAlignedTextData::wizard
```

Wizard flag

Referenced by DL\_Dxf::addArcAlignedText().

#### 4.1.2.27 xScaleFactor

```
double DL_ArcAlignedTextData::xScaleFactor
```

Relative X scale factor.

Referenced by DL\_Dxf::addArcAlignedText().

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

- src/dl\_entities.h

## 4.2 DL\_ArcData Struct Reference

Arc Data.

```
#include <dl_entities.h>
```

### Public Member Functions

- [DL\\_ArcData](#) (double acx, double acy, double acz, double aRadius, double aAngle1, double aAngle2)  
*Constructor.*

### Public Attributes

- double [cx](#)
- double [cy](#)
- double [cz](#)
- double [radius](#)
- double [angle1](#)
- double [angle2](#)

## 4.2.1 Detailed Description

Arc Data.

## 4.2.2 Constructor & Destructor Documentation

### 4.2.2.1 DL\_ArcData()

```
DL_ArcData::DL_ArcData (
    double acx,
    double acy,
    double acz,
    double aRadius,
    double aAngle1,
    double aAngle2 ) [inline]
```

Constructor.

Parameters: see member variables.

## 4.2.3 Member Data Documentation

### 4.2.3.1 angle1

```
double DL_ArcData::angle1
```

Startangle of arc in degrees.

Referenced by DL\_Dxf::writeArc().

### 4.2.3.2 angle2

```
double DL_ArcData::angle2
```

Endangle of arc in degrees.

Referenced by DL\_Dxf::writeArc().

#### 4.2.3.3 cx

```
double DL_ArcData::cx
```

X Coordinate of center point.

Referenced by DL\_Dxf::writeArc().

#### 4.2.3.4 cy

```
double DL_ArcData::cy
```

Y Coordinate of center point.

Referenced by DL\_Dxf::writeArc().

#### 4.2.3.5 cz

```
double DL_ArcData::cz
```

Z Coordinate of center point.

Referenced by DL\_Dxf::writeArc().

#### 4.2.3.6 radius

```
double DL_ArcData::radius
```

Radius of arc.

Referenced by DL\_Dxf::writeArc().

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

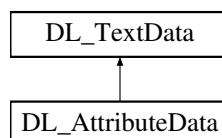
- src/dl\_entities.h

## 4.3 DL\_AttributeData Struct Reference

Block attribute data.

```
#include <dl_entities.h>
```

Inheritance diagram for DL\_AttributeData:



## Public Member Functions

- **DL\_AttributeData** (const [DL\\_TextData](#) &tData, const std::string &tag)
- **DL\_AttributeData** (double [ipx](#), double [ipy](#), double [ipz](#), double [apx](#), double [apy](#), double [apz](#), double [height](#), double [xScaleFactor](#), int [textGenerationFlags](#), int [hJustification](#), int [vJustification](#), const std::string &tag, const std::string &text, const std::string &style, double [angle](#))

*Constructor.*

## Public Attributes

- std::string [tag](#)

### 4.3.1 Detailed Description

Block attribute data.

### 4.3.2 Constructor & Destructor Documentation

#### 4.3.2.1 DL\_AttributeData()

```
DL_AttributeData::DL_AttributeData (
    double ipx,
    double ipy,
    double ipz,
    double apx,
    double apy,
    double apz,
    double height,
    double xScaleFactor,
    int textGenerationFlags,
    int hJustification,
    int vJustification,
    const std::string & tag,
    const std::string & text,
    const std::string & style,
    double angle ) [inline]
```

Constructor.

Parameters: see member variables.

### 4.3.3 Member Data Documentation

### 4.3.3.1 tag

```
std::string DL_AttributeData::tag
```

Tag.

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

- src/dl\_entities.h

## 4.4 DL\_Attributes Class Reference

Storing and passing around attributes.

```
#include <dl_attributes.h>
```

### Public Member Functions

- [DL\\_Attributes](#) ()  
*Default constructor.*
- [DL\\_Attributes](#) (const std::string &layer, int color, int width, const std::string &linetype, double linetypeScale)  
*Constructor for DXF attributes.*
- [DL\\_Attributes](#) (const std::string &layer, int color, int color24, int width, const std::string &linetype, int handle=-1)  
*Constructor for DXF attributes.*
- void [setLayer](#) (const std::string &layer)  
*Sets the layer.*
- std::string [getLayer](#) () const
- void [setColor](#) (int color)  
*Sets the color.*
- void [setColor24](#) (int color)  
*Sets the 24bit color.*
- int [getColor](#) () const
- int [getColor24](#) () const
- void [setWidth](#) (int width)  
*Sets the width.*
- int [getWidth](#) () const
- void [setLinetype](#) (const std::string &linetype)  
*Sets the line type.*
- void [setLinetypeScale](#) (double linetypeScale)  
*Sets the entity specific line type scale.*
- double [getLinetypeScale](#) () const
- std::string [getLinetype](#) () const
- void [setHandle](#) (int h)
- int [getHandle](#) () const
- void [setInPaperSpace](#) (bool on)
- bool [isInPaperSpace](#) () const

### 4.4.1 Detailed Description

Storing and passing around attributes.

Attributes are the layer name, color, width and line type.

Author

Andrew Mustun

### 4.4.2 Constructor & Destructor Documentation

#### 4.4.2.1 DL\_Attributes() [1/2]

```
DL_Attributes::DL_Attributes (
    const std::string & layer,
    int color,
    int width,
    const std::string & linetype,
    double linetypeScale ) [inline]
```

Constructor for DXF attributes.

Parameters

<i>layer</i>	Layer name for this entity or NULL for no layer (every entity should be on a named layer!).
<i>color</i>	Color number (0..256). 0 = BYBLOCK, 256 = BYLAYER.
<i>width</i>	Line thickness. Defaults to zero. -1 = BYLAYER, -2 = BYBLOCK, -3 = default width
<i>linetype</i>	Line type name or "BYLAYER" or "BYBLOCK". Defaults to "BYLAYER"

#### 4.4.2.2 DL\_Attributes() [2/2]

```
DL_Attributes::DL_Attributes (
    const std::string & layer,
    int color,
    int color24,
    int width,
    const std::string & linetype,
    int handle = -1 ) [inline]
```

Constructor for DXF attributes.

Parameters

<i>layer</i>	Layer name for this entity or NULL for no layer (every entity should be on a named layer!).
<i>color</i>	Color number (0..256). 0 = BYBLOCK, 256 = BYLAYER.

## Parameters

<i>color24</i>	24 bit color (0x00RRGGBB, see DXF reference).
<i>width</i>	Line thickness. Defaults to zero. -1 = BYLAYER, -2 = BYBLOCK, -3 = default width
<i>linetype</i>	Line type name or "BYLAYER" or "BYBLOCK". Defaults to "BYLAYER"

### 4.4.3 Member Function Documentation

#### 4.4.3.1 getColor()

```
int DL_Attributes::getColor ( ) const [inline]
```

## Returns

Color.

## See also

[DL\\_Codes](#), [dxfColors](#)

Referenced by [DL\\_Dxf::addLayer\(\)](#), [DL\\_Writer::entityAttributes\(\)](#), and [DL\\_Dxf::writeLayer\(\)](#).

#### 4.4.3.2 getColor24()

```
int DL_Attributes::getColor24 ( ) const [inline]
```

## Returns

24 bit color or -1 if no 24bit color is defined.

## See also

[DL\\_Codes](#), [dxfColors](#)

Referenced by [DL\\_Writer::entityAttributes\(\)](#), and [DL\\_Dxf::writeLayer\(\)](#).

#### 4.4.3.3 getLayer()

```
std::string DL_Attributes::getLayer ( ) const [inline]
```

##### Returns

Layer name.

Referenced by `DL_Writer::entityAttributes()`, and `DL_Dxf::writePolyline()`.

#### 4.4.3.4 getLinetype()

```
std::string DL_Attributes::getLinetype ( ) const [inline]
```

##### Returns

Line type.

Referenced by `DL_Dxf::addLayer()`, `DL_Writer::entityAttributes()`, and `DL_Dxf::writeLayer()`.

#### 4.4.3.5 getWidth()

```
int DL_Attributes::getWidth ( ) const [inline]
```

##### Returns

Width.

Referenced by `DL_Dxf::addLayer()`, `DL_Writer::entityAttributes()`, and `DL_Dxf::writeLayer()`.

#### 4.4.3.6 setColor()

```
void DL_Attributes::setColor (
    int color ) [inline]
```

Sets the color.

##### See also

[DL\\_Codes](#), `dxfColors`

Referenced by `DL_Dxf::addLayer()`.



#### 4.4.3.7 setColor24()

```
void DL_Attributes::setColor24 (
    int color ) [inline]
```

Sets the 24bit color.

See also

[DL\\_Codes](#), [dxfColors](#)

#### 4.4.3.8 setLayer()

```
void DL_Attributes::setLayer (
    const std::string & layer ) [inline]
```

Sets the layer.

If the given pointer points to NULL, the new layer name will be an empty but valid string.

#### 4.4.3.9 setLinetype()

```
void DL_Attributes::setLinetype (
    const std::string & linetype ) [inline]
```

Sets the line type.

This can be any string and is not checked to be a valid line type.

Referenced by [DL\\_Dxf::addLayer\(\)](#).

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

- [src/dl\\_attributes.h](#)

## 4.5 DL\_BlockData Struct Reference

Block Data.

```
#include <dl_entities.h>
```

### Public Member Functions

- [DL\\_BlockData](#) (const std::string &bName, int bFlags, double bbpx, double bbpy, double bbpz)  
*Constructor.*

## Public Attributes

- `std::string name`  
*Block name.*
- `int flags`  
*Block flags.*
- `double bpx`  
*X Coordinate of base point.*
- `double bpy`  
*Y Coordinate of base point.*
- `double bpz`  
*Z Coordinate of base point.*

### 4.5.1 Detailed Description

Block Data.

### 4.5.2 Constructor & Destructor Documentation

#### 4.5.2.1 DL\_BlockData()

```
DL_BlockData::DL_BlockData (
    const std::string & bName,
    int bFlags,
    double bbpx,
    double bbpy,
    double bbpz ) [inline]
```

Constructor.

Parameters: see member variables.

### 4.5.3 Member Data Documentation

#### 4.5.3.1 flags

```
int DL_BlockData::flags
```

Block flags.

(not used currently)

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

- `src/dl_entities.h`

## 4.6 DL\_CircleData Struct Reference

Circle Data.

```
#include <dl_entities.h>
```

### Public Member Functions

- [DL\\_CircleData](#) (double acx, double acy, double acz, double aRadius)  
*Constructor.*

### Public Attributes

- double [cx](#)
- double [cy](#)
- double [cz](#)
- double [radius](#)

#### 4.6.1 Detailed Description

Circle Data.

#### 4.6.2 Constructor & Destructor Documentation

##### 4.6.2.1 DL\_CircleData()

```
DL_CircleData::DL_CircleData (
    double acx,
    double acy,
    double acz,
    double aRadius ) [inline]
```

Constructor.

Parameters: see member variables.

#### 4.6.3 Member Data Documentation

#### 4.6.3.1 cx

```
double DL_CircleData::cx
```

X Coordinate of center point.

Referenced by DL\_Dxf::writeCircle().

#### 4.6.3.2 cy

```
double DL_CircleData::cy
```

Y Coordinate of center point.

Referenced by DL\_Dxf::writeCircle().

#### 4.6.3.3 cz

```
double DL_CircleData::cz
```

Z Coordinate of center point.

Referenced by DL\_Dxf::writeCircle().

#### 4.6.3.4 radius

```
double DL_CircleData::radius
```

Radius of arc.

Referenced by DL\_Dxf::writeCircle().

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

- src/dl\_entities.h

## 4.7 DL\_Codes Class Reference

Codes for colors and DXF versions.

```
#include <dl_codes.h>
```

## Public Types

- enum [color](#) {  
**black** = 250 , **green** = 3 , **red** = 1 , **brown** = 15 ,  
**yellow** = 2 , **cyan** = 4 , **magenta** = 6 , **gray** = 8 ,  
**blue** = 5 , **l\_blue** = 163 , **l\_green** = 121 , **l\_cyan** = 131 ,  
**l\_red** = 23 , **l\_magenta** = 221 , **l\_gray** = 252 , **white** = 7 ,  
**bylayer** = 256 , **byblock** = 0 }  
*Standard DXF colors.*
- enum [version](#) {  
**AC1009\_MIN** , **AC1009** , **AC1012** , **AC1014** ,  
**AC1015** }  
*Version numbers for the DXF Format.*

### 4.7.1 Detailed Description

Codes for colors and DXF versions.

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

- `src/dl_codes.h`

## 4.8 DL\_ControlPointData Struct Reference

Spline control point data.

```
#include <dl_entities.h>
```

### Public Member Functions

- [DL\\_ControlPointData](#) (double px, double py, double pz, double weight)  
*Constructor.*

### Public Attributes

- double [x](#)
- double [y](#)
- double [z](#)
- double [w](#)

### 4.8.1 Detailed Description

Spline control point data.

### 4.8.2 Constructor & Destructor Documentation

#### 4.8.2.1 DL\_ControlPointData()

```
DL_ControlPointData::DL_ControlPointData (
    double px,
    double py,
    double pz,
    double weight ) [inline]
```

Constructor.

Parameters: see member variables.

### 4.8.3 Member Data Documentation

#### 4.8.3.1 w

```
double DL_ControlPointData::w
```

Weight of control point.

#### 4.8.3.2 x

```
double DL_ControlPointData::x
```

X coordinate of the control point.

Referenced by DL\_Dxf::writeControlPoint().

#### 4.8.3.3 y

```
double DL_ControlPointData::y
```

Y coordinate of the control point.

Referenced by DL\_Dxf::writeControlPoint().

#### 4.8.3.4 z

```
double DL_ControlPointData::z
```

Z coordinate of the control point.

Referenced by DL\_Dxf::writeControlPoint().

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

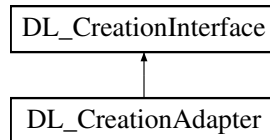
- src/dl\_entities.h

## 4.9 DL\_CreationAdapter Class Reference

An abstract adapter class for receiving DXF events when a DXF file is being read.

```
#include <dl_creationadapter.h>
```

Inheritance diagram for DL\_CreationAdapter:



### Public Member Functions

- virtual void [processCodeValuePair](#) (unsigned int, const std::string &)  
*Called for every code / value tuple of the DXF file.*
- virtual void [endSection](#) ()  
*Called when a section (entity, table entry, etc.) is finished.*
- virtual void [addLayer](#) (const [DL\\_LayerData](#) &)  
*Called for every layer.*
- virtual void [addLinetype](#) (const [DL\\_LinetypeData](#) &)  
*Called for every linetype.*
- virtual void [addLinetypeDash](#) (double)  
*Called for every dash in linetype pattern.*
- virtual void [addBlock](#) (const [DL\\_BlockData](#) &)  
*Called for every block.*
- virtual void [endBlock](#) ()  
*Called to end the current block.*
- virtual void [addTextStyle](#) (const [DL\\_StyleData](#) &)  
*Called for every text style.*
- virtual void [addPoint](#) (const [DL\\_PointData](#) &)  
*Called for every point.*
- virtual void [addLine](#) (const [DL\\_LineData](#) &)  
*Called for every line.*
- virtual void [addXLine](#) (const [DL\\_XLineData](#) &)  
*Called for every xline.*
- virtual void [addRay](#) (const [DL\\_RayData](#) &)  
*Called for every ray.*
- virtual void [addArc](#) (const [DL\\_ArcData](#) &)  
*Called for every arc.*
- virtual void [addCircle](#) (const [DL\\_CircleData](#) &)  
*Called for every circle.*
- virtual void [addEllipse](#) (const [DL\\_EllipseData](#) &)  
*Called for every ellipse.*
- virtual void [addPolyline](#) (const [DL\\_PolylineData](#) &)  
*Called for every polyline start.*
- virtual void [addVertex](#) (const [DL\\_VertexData](#) &)  
*Called for every polyline vertex.*

- virtual void [addSpline](#) (const [DL\\_SplineData](#) &)  
*Called for every spline.*
- virtual void [addControlPoint](#) (const [DL\\_ControlPointData](#) &)  
*Called for every spline control point.*
- virtual void [addFitPoint](#) (const [DL\\_FitPointData](#) &)  
*Called for every spline fit point.*
- virtual void [addKnot](#) (const [DL\\_KnotData](#) &)  
*Called for every spline knot value.*
- virtual void [addInsert](#) (const [DL\\_InsertData](#) &)  
*Called for every insert.*
- virtual void [addMText](#) (const [DL\\_MTextData](#) &)  
*Called for every multi Text entity.*
- virtual void [addMTextChunk](#) (const std::string &)  
*Called for additional text chunks for MTEXT entities.*
- virtual void [addText](#) (const [DL\\_TextData](#) &)  
*Called for every text entity.*
- virtual void [addArcAlignedText](#) (const [DL\\_ArcAlignedTextData](#) &)  
*Called for every arc aligned text entity.*
- virtual void [addAttribute](#) (const [DL\\_AttributeData](#) &)  
*Called for every block Attribute entity.*
- virtual void [addDimAlign](#) (const [DL\\_DimensionData](#) &, const [DL\\_DimAlignedData](#) &)  
*Called for every aligned dimension entity.*
- virtual void [addDimLinear](#) (const [DL\\_DimensionData](#) &, const [DL\\_DimLinearData](#) &)  
*Called for every linear or rotated dimension entity.*
- virtual void [addDimRadial](#) (const [DL\\_DimensionData](#) &, const [DL\\_DimRadialData](#) &)  
*Called for every radial dimension entity.*
- virtual void [addDimDiametric](#) (const [DL\\_DimensionData](#) &, const [DL\\_DimDiametricData](#) &)  
*Called for every diametric dimension entity.*
- virtual void [addDimAngular](#) (const [DL\\_DimensionData](#) &, const [DL\\_DimAngular2LData](#) &)  
*Called for every angular dimension (2 lines version) entity.*
- virtual void [addDimAngular3P](#) (const [DL\\_DimensionData](#) &, const [DL\\_DimAngular3PData](#) &)  
*Called for every angular dimension (3 points version) entity.*
- virtual void [addDimOrdinate](#) (const [DL\\_DimensionData](#) &, const [DL\\_DimOrdinateData](#) &)  
*Called for every ordinate dimension entity.*
- virtual void [addLeader](#) (const [DL\\_LeaderData](#) &)  
*Called for every leader start.*
- virtual void [addLeaderVertex](#) (const [DL\\_LeaderVertexData](#) &)  
*Called for every leader vertex.*
- virtual void [addHatch](#) (const [DL\\_HatchData](#) &)  
*Called for every hatch entity.*
- virtual void [addTrace](#) (const [DL\\_TraceData](#) &)  
*Called for every trace start.*
- virtual void [add3dFace](#) (const [DL\\_3dFaceData](#) &)  
*Called for every 3dface start.*
- virtual void [addSolid](#) (const [DL\\_SolidData](#) &)  
*Called for every solid start.*
- virtual void [addImage](#) (const [DL\\_ImageData](#) &)  
*Called for every image entity.*
- virtual void [linkImage](#) (const [DL\\_ImageDefData](#) &)  
*Called for every image definition.*
- virtual void [addHatchLoop](#) (const [DL\\_HatchLoopData](#) &)



- Called for every hatch loop.*

  - virtual void [addHatchEdge](#) (const [DL\\_HatchEdgeData](#) &)
- Called for every hatch edge entity.*

  - virtual void [addXRecord](#) (const std::string &)
- Called for every XRecord with the given handle.*

  - virtual void [addXRecordString](#) (int, const std::string &)
- Called for XRecords of type string.*

  - virtual void [addXRecordReal](#) (int, double)
- Called for XRecords of type double.*

  - virtual void [addXRecordInt](#) (int, int)
- Called for XRecords of type int.*

  - virtual void [addXRecordBool](#) (int, bool)
- Called for XRecords of type bool.*

  - virtual void [addXDataApp](#) (const std::string &)
- Called for every beginning of an XData section of the given application.*

  - virtual void [addXDataString](#) (int, const std::string &)
- Called for XData tuples.*

  - virtual void [addXDataReal](#) (int, double)
- Called for XData tuples.*

  - virtual void [addXDataInt](#) (int, int)
- Called for XData tuples.*

  - virtual void [addDictionary](#) (const [DL\\_DictionaryData](#) &)
- Called for dictionary objects.*

  - virtual void [addDictionaryEntry](#) (const [DL\\_DictionaryEntryData](#) &)
- Called for dictionary entries.*

  - virtual void [endEntity](#) ()
- Called after an entity has been completed.*

  - virtual void [addComment](#) (const std::string &)
- Called for every comment in the DXF file (code 999).*

  - virtual void [setVariableVector](#) (const std::string &, double, double, double, int)
- Called for every vector variable in the DXF file (e.g.*

  - virtual void [setVariableString](#) (const std::string &, const std::string &, int)
- Called for every string variable in the DXF file (e.g.*

  - virtual void [setVariableInt](#) (const std::string &, int, int)
- Called for every int variable in the DXF file (e.g.*

  - virtual void [setVariableDouble](#) (const std::string &, double, int)
- Called for every double variable in the DXF file (e.g.*

  - virtual void [endSequence](#) ()
- Called when a SEQEND occurs (when a POLYLINE or ATTRIB is done)*

## Additional Inherited Members

### 4.9.1 Detailed Description

An abstract adapter class for receiving DXF events when a DXF file is being read.

The methods in this class are empty. This class exists as convenience for creating listener objects.

Author

Andrew Mustun

## 4.9.2 Member Function Documentation

### 4.9.2.1 addBlock()

```
virtual void DL_CreationAdapter::addBlock (
    const DL_BlockData & data ) [inline], [virtual]
```

Called for every block.

Note: all entities added after this command go into this block until [endBlock\(\)](#) is called.

See also

[endBlock\(\)](#)

Implements [DL\\_CreationInterface](#).

### 4.9.2.2 addMTextChunk()

```
virtual void DL_CreationAdapter::addMTextChunk (
    const std::string & text ) [inline], [virtual]
```

Called for additional text chunks for MTEXT entities.

The chunks come at 250 character in size each. Note that those chunks come **before** the actual MTEXT entity.

Implements [DL\\_CreationInterface](#).

### 4.9.2.3 endEntity()

```
virtual void DL_CreationAdapter::endEntity ( ) [inline], [virtual]
```

Called after an entity has been completed.

Implements [DL\\_CreationInterface](#).

#### 4.9.2.4 processCodeValuePair()

```
virtual void DL_CreationAdapter::processCodeValuePair (
    unsigned int groupCode,
    const std::string & groupValue ) [inline], [virtual]
```

Called for every code / value tuple of the DXF file.

The complete DXF file contents can be handled by the implementation of this function.

Implements [DL\\_CreationInterface](#).

#### 4.9.2.5 setVariableDouble()

```
virtual void DL_CreationAdapter::setVariableDouble (
    const std::string & key,
    double value,
    int code ) [inline], [virtual]
```

Called for every double variable in the DXF file (e.g.

"\$DIMEXO").

Implements [DL\\_CreationInterface](#).

#### 4.9.2.6 setVariableInt()

```
virtual void DL_CreationAdapter::setVariableInt (
    const std::string & key,
    int value,
    int code ) [inline], [virtual]
```

Called for every int variable in the DXF file (e.g.

"\$ACADMAINTVER").

Implements [DL\\_CreationInterface](#).

#### 4.9.2.7 setVariableString()

```
virtual void DL_CreationAdapter::setVariableString (
    const std::string & key,
    const std::string & value,
    int code ) [inline], [virtual]
```

Called for every string variable in the DXF file (e.g.

"\$ACADVER").

Implements [DL\\_CreationInterface](#).

#### 4.9.2.8 setVariableVector()

```
virtual void DL_CreationAdapter::setVariableVector (
    const std::string & key,
    double v1,
    double v2,
    double v3,
    int code ) [inline], [virtual]
```

Called for every vector variable in the DXF file (e.g.

"\$EXTMIN").

Implements [DL\\_CreationInterface](#).

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

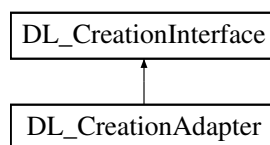
- src/dl\_creationadapter.h

## 4.10 DL\_CreationInterface Class Reference

Abstract class (interface) for the creation of new entities.

```
#include <dl_creationinterface.h>
```

Inheritance diagram for DL\_CreationInterface:



### Public Member Functions

- virtual void [processCodeValuePair](#) (unsigned int groupCode, const std::string &groupValue)=0  
*Called for every code / value tuple of the DXF file.*
- virtual void [endSection](#) ()=0  
*Called when a section (entity, table entry, etc.) is finished.*
- virtual void [addLayer](#) (const [DL\\_LayerData](#) &data)=0  
*Called for every layer.*
- virtual void [addLinetype](#) (const [DL\\_LinetypeData](#) &data)=0  
*Called for every linetype.*
- virtual void [addLinetypeDash](#) (double length)=0  
*Called for every dash in linetype pattern.*
- virtual void [addBlock](#) (const [DL\\_BlockData](#) &data)=0  
*Called for every block.*
- virtual void [endBlock](#) ()=0  
*Called to end the current block.*
- virtual void [addTextStyle](#) (const [DL\\_StyleData](#) &data)=0

- Called for every text style.*

  - virtual void [addPoint](#) (const [DL\\_PointData](#) &data)=0
- Called for every point.*

  - virtual void [addLine](#) (const [DL\\_LineData](#) &data)=0
- Called for every line.*

  - virtual void [addXLine](#) (const [DL\\_XLineData](#) &data)=0
- Called for every xline.*

  - virtual void [addRay](#) (const [DL\\_RayData](#) &data)=0
- Called for every ray.*

  - virtual void [addArc](#) (const [DL\\_ArcData](#) &data)=0
- Called for every arc.*

  - virtual void [addCircle](#) (const [DL\\_CircleData](#) &data)=0
- Called for every circle.*

  - virtual void [addEllipse](#) (const [DL\\_EllipseData](#) &data)=0
- Called for every ellipse.*

  - virtual void [addPolyline](#) (const [DL\\_PolylineData](#) &data)=0
- Called for every polyline start.*

  - virtual void [addVertex](#) (const [DL\\_VertexData](#) &data)=0
- Called for every polyline vertex.*

  - virtual void [addSpline](#) (const [DL\\_SplineData](#) &data)=0
- Called for every spline.*

  - virtual void [addControlPoint](#) (const [DL\\_ControlPointData](#) &data)=0
- Called for every spline control point.*

  - virtual void [addFitPoint](#) (const [DL\\_FitPointData](#) &data)=0
- Called for every spline fit point.*

  - virtual void [addKnot](#) (const [DL\\_KnotData](#) &data)=0
- Called for every spline knot value.*

  - virtual void [addInsert](#) (const [DL\\_InsertData](#) &data)=0
- Called for every insert.*

  - virtual void [addTrace](#) (const [DL\\_TraceData](#) &data)=0
- Called for every trace start.*

  - virtual void [add3dFace](#) (const [DL\\_3dFaceData](#) &data)=0
- Called for every 3dface start.*

  - virtual void [addSolid](#) (const [DL\\_SolidData](#) &data)=0
- Called for every solid start.*

  - virtual void [addMText](#) (const [DL\\_MTextData](#) &data)=0
- Called for every multi Text entity.*

  - virtual void [addMTextChunk](#) (const std::string &text)=0
- Called for additional text chunks for MTEXT entities.*

  - virtual void [addText](#) (const [DL\\_TextData](#) &data)=0
- Called for every text entity.*

  - virtual void [addArcAlignedText](#) (const [DL\\_ArcAlignedTextData](#) &data)=0
- Called for every arc aligned text entity.*

  - virtual void [addAttribute](#) (const [DL\\_AttributeData](#) &data)=0
- Called for every block Attribute entity.*

  - virtual void [addDimAlign](#) (const [DL\\_DimensionData](#) &data, const [DL\\_DimAlignedData](#) &edata)=0
- Called for every aligned dimension entity.*

  - virtual void [addDimLinear](#) (const [DL\\_DimensionData](#) &data, const [DL\\_DimLinearData](#) &edata)=0
- Called for every linear or rotated dimension entity.*

  - virtual void [addDimRadial](#) (const [DL\\_DimensionData](#) &data, const [DL\\_DimRadialData](#) &edata)=0
- Called for every radial dimension entity.*

- virtual void [addDimDiametric](#) (const [DL\\_DimensionData](#) &data, const [DL\\_DimDiametricData](#) &edata)=0  
*Called for every diametric dimension entity.*
- virtual void [addDimAngular](#) (const [DL\\_DimensionData](#) &data, const [DL\\_DimAngular2LData](#) &edata)=0  
*Called for every angular dimension (2 lines version) entity.*
- virtual void [addDimAngular3P](#) (const [DL\\_DimensionData](#) &data, const [DL\\_DimAngular3PData](#) &edata)=0  
*Called for every angular dimension (3 points version) entity.*
- virtual void [addDimOrdinate](#) (const [DL\\_DimensionData](#) &data, const [DL\\_DimOrdinateData](#) &edata)=0  
*Called for every ordinate dimension entity.*
- virtual void [addLeader](#) (const [DL\\_LeaderData](#) &data)=0  
*Called for every leader start.*
- virtual void [addLeaderVertex](#) (const [DL\\_LeaderVertexData](#) &data)=0  
*Called for every leader vertex.*
- virtual void [addHatch](#) (const [DL\\_HatchData](#) &data)=0  
*Called for every hatch entity.*
- virtual void [addImage](#) (const [DL\\_ImageData](#) &data)=0  
*Called for every image entity.*
- virtual void [linkImage](#) (const [DL\\_ImageDefData](#) &data)=0  
*Called for every image definition.*
- virtual void [addHatchLoop](#) (const [DL\\_HatchLoopData](#) &data)=0  
*Called for every hatch loop.*
- virtual void [addHatchEdge](#) (const [DL\\_HatchEdgeData](#) &data)=0  
*Called for every hatch edge entity.*
- virtual void [addXRecord](#) (const std::string &handle)=0  
*Called for every XRecord with the given handle.*
- virtual void [addXRecordString](#) (int code, const std::string &value)=0  
*Called for XRecords of type string.*
- virtual void [addXRecordReal](#) (int code, double value)=0  
*Called for XRecords of type double.*
- virtual void [addXRecordInt](#) (int code, int value)=0  
*Called for XRecords of type int.*
- virtual void [addXRecordBool](#) (int code, bool value)=0  
*Called for XRecords of type bool.*
- virtual void [addXDataApp](#) (const std::string &appld)=0  
*Called for every beginning of an XData section of the given application.*
- virtual void [addXDataString](#) (int code, const std::string &value)=0  
*Called for XData tuples.*
- virtual void [addXDataReal](#) (int code, double value)=0  
*Called for XData tuples.*
- virtual void [addXDataInt](#) (int code, int value)=0  
*Called for XData tuples.*
- virtual void [addDictionary](#) (const [DL\\_DictionaryData](#) &data)=0  
*Called for dictionary objects.*
- virtual void [addDictionaryEntry](#) (const [DL\\_DictionaryEntryData](#) &data)=0  
*Called for dictionary entries.*
- virtual void [endEntity](#) ()=0  
*Called after an entity has been completed.*
- virtual void [addComment](#) (const std::string &comment)=0  
*Called for every comment in the DXF file (code 999).*
- virtual void [setVariableVector](#) (const std::string &key, double v1, double v2, double v3, int code)=0  
*Called for every vector variable in the DXF file (e.g.*
- virtual void [setVariableString](#) (const std::string &key, const std::string &value, int code)=0

- Called for every string variable in the DXF file (e.g.*
  - virtual void [setVariableInt](#) (const std::string &key, int value, int code)=0
- Called for every int variable in the DXF file (e.g.*
  - virtual void [setVariableDouble](#) (const std::string &key, double value, int code)=0
- Called for every double variable in the DXF file (e.g.*
  - virtual void [endSequence](#) ()=0
- Called when a SEQEND occurs (when a POLYLINE or ATTRIB is done)*
  - void [setAttributes](#) (const [DL\\_Attributes](#) &attrib)
- Sets the current attributes for entities.*
  - [DL\\_Attributes](#) [getAttributes](#) ()
- Sets the current attributes for entities.*
  - void [setExtrusion](#) (double dx, double dy, double dz, double elevation)
- Sets the current attributes for entities.*
  - [DL\\_Extrusion](#) \* [getExtrusion](#) ()

## Protected Attributes

- [DL\\_Attributes](#) **attributes**
- [DL\\_Extrusion](#) \* **extrusion**

### 4.10.1 Detailed Description

Abstract class (interface) for the creation of new entities.

Inherit your class which takes care of the entities in the processed DXF file from this interface.

Double arrays passed to your implementation contain 3 double values for x, y, z coordinates unless stated differently.

Author

Andrew Mustun

### 4.10.2 Member Function Documentation

#### 4.10.2.1 addBlock()

```
virtual void DL_CreationInterface::addBlock (
    const DL\_BlockData & data ) [pure virtual]
```

Called for every block.

Note: all entities added after this command go into this block until [endBlock\(\)](#) is called.

See also

[endBlock\(\)](#)

Implemented in [DL\\_CreationAdapter](#).

Referenced by [DL\\_Dxf::addBlock\(\)](#).

#### 4.10.2.2 addMTextChunk()

```
virtual void DL_CreationInterface::addMTextChunk (
    const std::string & text ) [pure virtual]
```

Called for additional text chunks for MTEXT entities.

The chunks come at 250 character in size each. Note that those chunks come **before** the actual MTEXT entity.

Implemented in [DL\\_CreationAdapter](#).

Referenced by DL\_Dxf::handleMTextData().

#### 4.10.2.3 endEntity()

```
virtual void DL_CreationInterface::endEntity ( ) [pure virtual]
```

Called after an entity has been completed.

Implemented in [DL\\_CreationAdapter](#).

Referenced by DL\_Dxf::addHatch(), DL\_Dxf::addImage(), DL\_Dxf::addImageDef(), DL\_Dxf::addLeader(), DL\_Dxf::addPolyline(), DL\_Dxf::addSpline(), and DL\_Dxf::endEntity().

#### 4.10.2.4 getAttributes()

```
DL_Attributes DL_CreationInterface::getAttributes ( ) [inline]
```

##### Returns

the current attributes used for new entities.

Referenced by DL\_Dxf::addLayer().

#### 4.10.2.5 getExtrusion()

```
DL_Extrusion* DL_CreationInterface::getExtrusion ( ) [inline]
```

##### Returns

the current attributes used for new entities.



#### 4.10.2.6 processCodeValuePair()

```
virtual void DL_CreationInterface::processCodeValuePair (
    unsigned int groupCode,
    const std::string & groupValue ) [pure virtual]
```

Called for every code / value tuple of the DXF file.

The complete DXF file contents can be handled by the implementation of this function.

Implemented in [DL\\_CreationAdapter](#).

Referenced by `DL_Dxf::readDxfGroups()`.

#### 4.10.2.7 setVariableDouble()

```
virtual void DL_CreationInterface::setVariableDouble (
    const std::string & key,
    double value,
    int code ) [pure virtual]
```

Called for every double variable in the DXF file (e.g.

"\$DIMEXO").

Implemented in [DL\\_CreationAdapter](#).

Referenced by `DL_Dxf::addSetting()`.

#### 4.10.2.8 setVariableInt()

```
virtual void DL_CreationInterface::setVariableInt (
    const std::string & key,
    int value,
    int code ) [pure virtual]
```

Called for every int variable in the DXF file (e.g.

"\$ACADMAINTVER").

Implemented in [DL\\_CreationAdapter](#).

Referenced by `DL_Dxf::addSetting()`.

#### 4.10.2.9 setVariableString()

```
virtual void DL_CreationInterface::setVariableString (
    const std::string & key,
    const std::string & value,
    int code ) [pure virtual]
```

Called for every string variable in the DXF file (e.g.

"\$ACADVER").

Implemented in [DL\\_CreationAdapter](#).

Referenced by DL\_Dxf::addSetting().

#### 4.10.2.10 setVariableVector()

```
virtual void DL_CreationInterface::setVariableVector (
    const std::string & key,
    double v1,
    double v2,
    double v3,
    int code ) [pure virtual]
```

Called for every vector variable in the DXF file (e.g.

"\$EXTMIN").

Implemented in [DL\\_CreationAdapter](#).

Referenced by DL\_Dxf::addSetting().

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

- src/dl\_creationinterface.h

## 4.11 DL\_DictionaryData Struct Reference

Dictionary data.

```
#include <dl_entities.h>
```

### Public Member Functions

- **DL\_DictionaryData** (const std::string &handle)

### Public Attributes

- std::string **handle**

### 4.11.1 Detailed Description

Dictionary data.

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

- src/dl\_entities.h

## 4.12 DL\_DictionaryEntryData Struct Reference

Dictionary entry data.

```
#include <dl_entities.h>
```

### Public Member Functions

- **DL\_DictionaryEntryData** (const std::string &name, const std::string &handle)

### Public Attributes

- std::string **name**
- std::string **handle**

### 4.12.1 Detailed Description

Dictionary entry data.

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

- src/dl\_entities.h

## 4.13 DL\_DimAlignedData Struct Reference

Aligned Dimension Data.

```
#include <dl_entities.h>
```

### Public Member Functions

- [DL\\_DimAlignedData](#) (double depx1, double depy1, double depz1, double depx2, double depy2, double depz2)

*Constructor.*

## Public Attributes

- double [epx1](#)
- double [epy1](#)
- double [epz1](#)
- double [epx2](#)
- double [epy2](#)
- double [epz2](#)

### 4.13.1 Detailed Description

Aligned Dimension Data.

### 4.13.2 Constructor & Destructor Documentation

#### 4.13.2.1 DL\_DimAlignedData()

```
DL_DimAlignedData::DL_DimAlignedData (
    double depx1,
    double depy1,
    double depz1,
    double depx2,
    double depy2,
    double depz2 ) [inline]
```

Constructor.

Parameters: see member variables.

### 4.13.3 Member Data Documentation

#### 4.13.3.1 [epx1](#)

```
double DL_DimAlignedData::epx1
```

X Coordinate of Extension point 1.

Referenced by `DL_Dxf::writeDimAligned()`.

#### 4.13.3.2 ep<sub>x</sub>2

double DL\_DimAlignedData::ep<sub>x</sub>2

X Coordinate of Extension point 2.

Referenced by DL\_Dxf::writeDimAligned().

#### 4.13.3.3 ep<sub>y</sub>1

double DL\_DimAlignedData::ep<sub>y</sub>1

Y Coordinate of Extension point 1.

Referenced by DL\_Dxf::writeDimAligned().

#### 4.13.3.4 ep<sub>y</sub>2

double DL\_DimAlignedData::ep<sub>y</sub>2

Y Coordinate of Extension point 2.

Referenced by DL\_Dxf::writeDimAligned().

#### 4.13.3.5 ep<sub>z</sub>1

double DL\_DimAlignedData::ep<sub>z</sub>1

Z Coordinate of Extension point 1.

#### 4.13.3.6 ep<sub>z</sub>2

double DL\_DimAlignedData::ep<sub>z</sub>2

Z Coordinate of Extension point 2.

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

- src/dl\_entities.h

## 4.14 DL\_DimAngular2LData Struct Reference

Angular Dimension Data.

```
#include <dl_entities.h>
```

### Public Member Functions

- [DL\\_DimAngular2LData](#) (double ddp<sub>x</sub>1, double ddp<sub>y</sub>1, double ddp<sub>z</sub>1, double ddp<sub>x</sub>2, double ddp<sub>y</sub>2, double ddp<sub>z</sub>2, double ddp<sub>x</sub>3, double ddp<sub>y</sub>3, double ddp<sub>z</sub>3, double ddp<sub>x</sub>4, double ddp<sub>y</sub>4, double ddp<sub>z</sub>4)  
*Constructor.*

### Public Attributes

- double [dpx1](#)
- double [dpy1](#)
- double [dpz1](#)
- double [dpx2](#)
- double [dpy2](#)
- double [dpz2](#)
- double [dpx3](#)
- double [dpy3](#)
- double [dpz3](#)
- double [dpx4](#)
- double [dpy4](#)
- double [dpz4](#)

#### 4.14.1 Detailed Description

Angular Dimension Data.

#### 4.14.2 Constructor & Destructor Documentation

##### 4.14.2.1 DL\_DimAngular2LData()

```
DL_DimAngular2LData::DL_DimAngular2LData (
    double ddpx1,
    double ddpy1,
    double ddpz1,
    double ddpx2,
    double ddpy2,
    double ddpz2,
    double ddpx3,
    double ddpy3,
    double ddpz3,
    double ddpx4,
    double ddpy4,
    double ddpz4 ) [inline]
```

Constructor.

Parameters: see member variables.

### 4.14.3 Member Data Documentation

#### 4.14.3.1 dpx1

```
double DL_DimAngular2LData::dpx1
```

X Coordinate of definition point 1.

Referenced by DL\_Dxf::writeDimAngular2L().

#### 4.14.3.2 dpx2

```
double DL_DimAngular2LData::dpx2
```

X Coordinate of definition point 2.

Referenced by DL\_Dxf::writeDimAngular2L().

#### 4.14.3.3 dpx3

```
double DL_DimAngular2LData::dpx3
```

X Coordinate of definition point 3.

Referenced by DL\_Dxf::writeDimAngular2L().

#### 4.14.3.4 dpx4

```
double DL_DimAngular2LData::dpx4
```

X Coordinate of definition point 4.

Referenced by DL\_Dxf::writeDimAngular2L().

#### 4.14.3.5 dpy1

```
double DL_DimAngular2LData::dpy1
```

Y Coordinate of definition point 1.

Referenced by DL\_Dxf::writeDimAngular2L().

#### 4.14.3.6 dpy2

```
double DL_DimAngular2LData::dpy2
```

Y Coordinate of definition point 2.

Referenced by DL\_Dxf::writeDimAngular2L().

#### 4.14.3.7 dpy3

```
double DL_DimAngular2LData::dpy3
```

Y Coordinate of definition point 3.

Referenced by DL\_Dxf::writeDimAngular2L().

#### 4.14.3.8 dpy4

```
double DL_DimAngular2LData::dpy4
```

Y Coordinate of definition point 4.

Referenced by DL\_Dxf::writeDimAngular2L().

#### 4.14.3.9 dpz1

```
double DL_DimAngular2LData::dpz1
```

Z Coordinate of definition point 1.

#### 4.14.3.10 dpz2

```
double DL_DimAngular2LData::dpz2
```

Z Coordinate of definition point 2.

#### 4.14.3.11 dpz3

```
double DL_DimAngular2LData::dpz3
```

Z Coordinate of definition point 3.



#### 4.14.3.12 dpz4

```
double DL_DimAngular2LData::dpz4
```

Z Coordinate of definition point 4.

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

- src/dl\_entities.h

## 4.15 DL\_DimAngular3PData Struct Reference

Angular Dimension Data (3 points version).

```
#include <dl_entities.h>
```

### Public Member Functions

- [DL\\_DimAngular3PData](#) (double ddp<sub>x</sub>1, double ddp<sub>y</sub>1, double ddp<sub>z</sub>1, double ddp<sub>x</sub>2, double ddp<sub>y</sub>2, double ddp<sub>z</sub>2, double ddp<sub>x</sub>3, double ddp<sub>y</sub>3, double ddp<sub>z</sub>3)

*Constructor.*

### Public Attributes

- double [dpx1](#)
- double [dpy1](#)
- double [dpz1](#)
- double [dpx2](#)
- double [dpy2](#)
- double [dpz2](#)
- double [dpx3](#)
- double [dpy3](#)
- double [dpz3](#)

#### 4.15.1 Detailed Description

Angular Dimension Data (3 points version).

#### 4.15.2 Constructor & Destructor Documentation

#### 4.15.2.1 DL\_DimAngular3PData()

```
DL_DimAngular3PData::DL_DimAngular3PData (
    double ddp1,
    double ddp2,
    double ddp3,
    double ddp4,
    double ddp5,
    double ddp6,
    double ddp7,
    double ddp8,
    double ddp9 ) [inline]
```

Constructor.

Parameters: see member variables.

### 4.15.3 Member Data Documentation

#### 4.15.3.1 dpx1

```
double DL_DimAngular3PData::dpx1
```

X Coordinate of definition point 1 (extension line 1 end).

Referenced by DL\_Dxf::writeDimAngular3P().

#### 4.15.3.2 dpx2

```
double DL_DimAngular3PData::dpx2
```

X Coordinate of definition point 2 (extension line 2 end).

Referenced by DL\_Dxf::writeDimAngular3P().

#### 4.15.3.3 dpx3

```
double DL_DimAngular3PData::dpx3
```

X Coordinate of definition point 3 (center).

Referenced by DL\_Dxf::writeDimAngular3P().

#### 4.15.3.4 dpy1

```
double DL_DimAngular3PData::dpy1
```

Y Coordinate of definition point 1.

Referenced by DL\_Dxf::writeDimAngular3P().

#### 4.15.3.5 dpy2

```
double DL_DimAngular3PData::dpy2
```

Y Coordinate of definition point 2.

Referenced by DL\_Dxf::writeDimAngular3P().

#### 4.15.3.6 dpy3

```
double DL_DimAngular3PData::dpy3
```

Y Coordinate of definition point 3.

Referenced by DL\_Dxf::writeDimAngular3P().

#### 4.15.3.7 dpz1

```
double DL_DimAngular3PData::dpz1
```

Z Coordinate of definition point 1.

#### 4.15.3.8 dpz2

```
double DL_DimAngular3PData::dpz2
```

Z Coordinate of definition point 2.

#### 4.15.3.9 dpz3

```
double DL_DimAngular3PData::dpz3
```

Z Coordinate of definition point 3.

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

- src/dl\_entities.h

## 4.16 DL\_DimDiametricData Struct Reference

Diametric Dimension Data.

```
#include <dl_entities.h>
```

### Public Member Functions

- [DL\\_DimDiametricData](#) (double ddp<sub>x</sub>, double ddp<sub>y</sub>, double ddp<sub>z</sub>, double dleader)  
*Constructor.*

### Public Attributes

- double [dpx](#)
- double [dpy](#)
- double [dpz](#)
- double [leader](#)

#### 4.16.1 Detailed Description

Diametric Dimension Data.

#### 4.16.2 Constructor & Destructor Documentation

##### 4.16.2.1 DL\_DimDiametricData()

```
DL_DimDiametricData::DL_DimDiametricData (  
    double ddpx,  
    double ddpy,  
    double ddpz,  
    double dleader ) [inline]
```

Constructor.

Parameters: see member variables.

#### 4.16.3 Member Data Documentation

#### 4.16.3.1 dpx

```
double DL_DimDiametricData::dpx
```

X Coordinate of definition point (DXF 15).

Referenced by DL\_Dxf::writeDimDiametric().

#### 4.16.3.2 dpy

```
double DL_DimDiametricData::dpy
```

Y Coordinate of definition point (DXF 25).

Referenced by DL\_Dxf::writeDimDiametric().

#### 4.16.3.3 dpz

```
double DL_DimDiametricData::dpz
```

Z Coordinate of definition point (DXF 35).

#### 4.16.3.4 leader

```
double DL_DimDiametricData::leader
```

Leader length

Referenced by DL\_Dxf::writeDimDiametric().

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

- src/dl\_entities.h

## 4.17 DL\_DimensionData Struct Reference

Generic Dimension Data.

```
#include <dl_entities.h>
```

### Public Member Functions

- [DL\\_DimensionData](#) (double [dpx](#), double [dpy](#), double [dpz](#), double [mpx](#), double [mpy](#), double [mpz](#), int [type](#), int [attachmentPoint](#), int [lineSpacingStyle](#), double [lineSpacingFactor](#), const std::string &[text](#), const std::string &[style](#), double [angle](#), double [linearFactor](#)=1.0, double [dimScale](#)=1.0)  
*Constructor.*

## Public Attributes

- double [dpx](#)
- double [dpy](#)
- double [dpz](#)
- double [mpx](#)
- double [mpy](#)
- double [mpz](#)
- int [type](#)  
*Dimension type.*
- int [attachmentPoint](#)  
*Attachment point.*
- int [lineSpacingStyle](#)  
*Line spacing style.*
- double [lineSpacingFactor](#)  
*Line spacing factor.*
- std::string [text](#)  
*Text string.*
- std::string [style](#)
- double [angle](#)  
*Rotation angle of dimension text away from default orientation.*
- double [linearFactor](#)  
*Linear factor style override.*
- double [dimScale](#)  
*Dimension scale (dimscale) style override.*
- bool **arrow1Flipped**
- bool **arrow2Flipped**

### 4.17.1 Detailed Description

Generic Dimension Data.

### 4.17.2 Constructor & Destructor Documentation

#### 4.17.2.1 DL\_DimensionData()

```
DL_DimensionData::DL_DimensionData (
    double dpx,
    double dpy,
    double dpz,
    double mpx,
    double mpy,
    double mpz,
    int type,
    int attachmentPoint,
    int lineSpacingStyle,
    double lineSpacingFactor,
    const std::string & text,
```

```
const std::string & style,  
double angle,  
double linearFactor = 1.0,  
double dimScale = 1.0 ) [inline]
```

Constructor.

Parameters: see member variables.

### 4.17.3 Member Data Documentation

#### 4.17.3.1 attachmentPoint

```
int DL_DimensionData::attachmentPoint
```

Attachment point.

1 = Top left, 2 = Top center, 3 = Top right, 4 = Middle left, 5 = Middle center, 6 = Middle right, 7 = Bottom left, 8 = Bottom center, 9 = Bottom right,

Referenced by DL\_Dxf::writeDimAligned(), DL\_Dxf::writeDimAngular2L(), DL\_Dxf::writeDimAngular3P(), DL\_Dxf::writeDimDiametric(), DL\_Dxf::writeDimLinear(), DL\_Dxf::writeDimOrdinate(), and DL\_Dxf::writeDimRadial().

#### 4.17.3.2 dpx

```
double DL_DimensionData::dpx
```

X Coordinate of definition point.

Referenced by DL\_Dxf::writeDimAligned(), DL\_Dxf::writeDimAngular2L(), DL\_Dxf::writeDimAngular3P(), DL\_Dxf::writeDimDiametric(), DL\_Dxf::writeDimLinear(), DL\_Dxf::writeDimOrdinate(), and DL\_Dxf::writeDimRadial().

#### 4.17.3.3 dpy

```
double DL_DimensionData::dpy
```

Y Coordinate of definition point.

Referenced by DL\_Dxf::writeDimAligned(), DL\_Dxf::writeDimAngular2L(), DL\_Dxf::writeDimAngular3P(), DL\_Dxf::writeDimDiametric(), DL\_Dxf::writeDimLinear(), DL\_Dxf::writeDimOrdinate(), and DL\_Dxf::writeDimRadial().

#### 4.17.3.4 dpz

```
double DL_DimensionData::dpz
```

Z Coordinate of definition point.

Referenced by `DL_Dxf::writeDimAligned()`, `DL_Dxf::writeDimAngular2L()`, `DL_Dxf::writeDimAngular3P()`, `DL_Dxf::writeDimDiametric()`, `DL_Dxf::writeDimLinear()`, `DL_Dxf::writeDimOrdinate()`, and `DL_Dxf::writeDimRadial()`.

#### 4.17.3.5 lineSpacingFactor

```
double DL_DimensionData::lineSpacingFactor
```

Line spacing factor.

0.25 .. 4.0

Referenced by `DL_Dxf::writeDimAligned()`, `DL_Dxf::writeDimAngular2L()`, `DL_Dxf::writeDimAngular3P()`, `DL_Dxf::writeDimDiametric()`, `DL_Dxf::writeDimLinear()`, `DL_Dxf::writeDimOrdinate()`, and `DL_Dxf::writeDimRadial()`.

#### 4.17.3.6 lineSpacingStyle

```
int DL_DimensionData::lineSpacingStyle
```

Line spacing style.

1 = at least, 2 = exact

Referenced by `DL_Dxf::writeDimAligned()`, `DL_Dxf::writeDimAngular2L()`, `DL_Dxf::writeDimAngular3P()`, `DL_Dxf::writeDimDiametric()`, `DL_Dxf::writeDimLinear()`, `DL_Dxf::writeDimOrdinate()`, and `DL_Dxf::writeDimRadial()`.

#### 4.17.3.7 mpx

```
double DL_DimensionData::mpx
```

X Coordinate of middle point of the text.

Referenced by `DL_Dxf::writeDimAligned()`, `DL_Dxf::writeDimAngular2L()`, `DL_Dxf::writeDimAngular3P()`, `DL_Dxf::writeDimDiametric()`, `DL_Dxf::writeDimLinear()`, `DL_Dxf::writeDimOrdinate()`, and `DL_Dxf::writeDimRadial()`.



#### 4.17.3.8 mpy

```
double DL_DimensionData::mpy
```

Y Coordinate of middle point of the text.

Referenced by `DL_Dxf::writeDimAligned()`, `DL_Dxf::writeDimAngular2L()`, `DL_Dxf::writeDimAngular3P()`, `DL_Dxf::writeDimDiametric()`, `DL_Dxf::writeDimLinear()`, `DL_Dxf::writeDimOrdinate()`, and `DL_Dxf::writeDimRadial()`.

#### 4.17.3.9 mpz

```
double DL_DimensionData::mpz
```

Z Coordinate of middle point of the text.

#### 4.17.3.10 style

```
std::string DL_DimensionData::style
```

Dimension style (font name).

#### 4.17.3.11 text

```
std::string DL_DimensionData::text
```

Text string.

Text string entered explicitly by user or null or "<>" for the actual measurement or " " (one blank space). for supressing the text.

Referenced by `DL_Dxf::writeDimAligned()`, `DL_Dxf::writeDimAngular2L()`, `DL_Dxf::writeDimAngular3P()`, `DL_Dxf::writeDimDiametric()`, `DL_Dxf::writeDimLinear()`, `DL_Dxf::writeDimOrdinate()`, and `DL_Dxf::writeDimRadial()`.

#### 4.17.3.12 type

```
int DL_DimensionData::type
```

Dimension type.

0 Rotated, horizontal, or vertical

1 Aligned

2 Angular

3 Diametric

4 Radius

5 Angular 3-point

6 Ordinate

64 Ordinate type. This is a bit value (bit 7) used only with integer value 6. If set, ordinate is X-type; if not set, ordinate is Y-type

128 This is a bit value (bit 8) added to the other group 70 values if the dimension text has been positioned at a user-defined location rather than at the default location

Referenced by `DL_Dxf::writeDimAligned()`, `DL_Dxf::writeDimAngular2L()`, `DL_Dxf::writeDimAngular3P()`, `DL_Dxf::writeDimDiametric()`, `DL_Dxf::writeDimLinear()`, `DL_Dxf::writeDimOrdinate()`, and `DL_Dxf::writeDimRadial()`.

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

- `src/dl_entities.h`

## 4.18 DL\_DimLinearData Struct Reference

Linear (rotated) Dimension Data.

```
#include <dl_entities.h>
```

### Public Member Functions

- [DL\\_DimLinearData](#) (double ddp<sub>x</sub>1, double ddp<sub>y</sub>1, double ddp<sub>z</sub>1, double ddp<sub>x</sub>2, double ddp<sub>y</sub>2, double ddp<sub>z</sub>2, double dAngle, double dOblique)

*Constructor.*

### Public Attributes

- double [dpx1](#)
- double [dpy1](#)
- double [dpz1](#)
- double [dpx2](#)
- double [dpy2](#)
- double [dpz2](#)
- double [angle](#)
- double [oblique](#)

### 4.18.1 Detailed Description

Linear (rotated) Dimension Data.

### 4.18.2 Constructor & Destructor Documentation

#### 4.18.2.1 DL\_DimLinearData()

```
DL_DimLinearData::DL_DimLinearData (
    double ddpx1,
    double ddpy1,
    double ddpz1,
    double ddpx2,
    double ddpy2,
    double ddpz2,
    double dAngle,
    double dOblique ) [inline]
```

Constructor.

Parameters: see member variables.

### 4.18.3 Member Data Documentation

#### 4.18.3.1 angle

```
double DL_DimLinearData::angle
```

Rotation angle (angle of dimension line) in degrees.

Referenced by DL\_Dxf::writeDimLinear().

#### 4.18.3.2 dpx1

```
double DL_DimLinearData::dpx1
```

X Coordinate of Extension point 1.

Referenced by DL\_Dxf::writeDimLinear().

#### 4.18.3.3 dpx2

```
double DL_DimLinearData::dpx2
```

X Coordinate of Extension point 2.

Referenced by DL\_Dxf::writeDimLinear().

#### 4.18.3.4 dpy1

```
double DL_DimLinearData::dpy1
```

Y Coordinate of Extension point 1.

Referenced by DL\_Dxf::writeDimLinear().

#### 4.18.3.5 dpy2

```
double DL_DimLinearData::dpy2
```

Y Coordinate of Extension point 2.

Referenced by DL\_Dxf::writeDimLinear().

#### 4.18.3.6 dpz1

```
double DL_DimLinearData::dpz1
```

Z Coordinate of Extension point 1.

#### 4.18.3.7 dpz2

```
double DL_DimLinearData::dpz2
```

Z Coordinate of Extension point 2.

#### 4.18.3.8 oblique

```
double DL_DimLinearData::oblique
```

Oblique angle in degrees.

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

- `src/dl_entities.h`

## 4.19 DL\_DimOrdinateData Struct Reference

Ordinate Dimension Data.

```
#include <dl_entities.h>
```

### Public Member Functions

- [DL\\_DimOrdinateData](#) (double ddp<sub>x</sub>1, double ddp<sub>y</sub>1, double ddp<sub>z</sub>1, double ddp<sub>x</sub>2, double ddp<sub>y</sub>2, double ddp<sub>z</sub>2, bool dxtype)  
*Constructor.*

### Public Attributes

- double [dpx1](#)
- double [dpy1](#)
- double [dpz1](#)
- double [dpx2](#)
- double [dpy2](#)
- double [dpz2](#)
- bool [xtype](#)

#### 4.19.1 Detailed Description

Ordinate Dimension Data.

## 4.19.2 Constructor & Destructor Documentation

### 4.19.2.1 DL\_DimOrdinateData()

```
DL_DimOrdinateData::DL_DimOrdinateData (
    double ddpx1,
    double ddpy1,
    double ddpz1,
    double ddpx2,
    double ddpy2,
    double ddpz2,
    bool dxtype ) [inline]
```

Constructor.

Parameters: see member variables.

## 4.19.3 Member Data Documentation

### 4.19.3.1 dpx1

```
double DL_DimOrdinateData::dpx1
```

X Coordinate of definition point 1.

Referenced by DL\_Dxf::writeDimOrdinate().

### 4.19.3.2 dpx2

```
double DL_DimOrdinateData::dpx2
```

X Coordinate of definition point 2.

Referenced by DL\_Dxf::writeDimOrdinate().

### 4.19.3.3 dpy1

```
double DL_DimOrdinateData::dpy1
```

Y Coordinate of definition point 1.

Referenced by DL\_Dxf::writeDimOrdinate().

#### 4.19.3.4 dpy2

```
double DL_DimOrdinateData::dpy2
```

Y Coordinate of definition point 2.

Referenced by DL\_Dxf::writeDimOrdinate().

#### 4.19.3.5 dpz1

```
double DL_DimOrdinateData::dpz1
```

Z Coordinate of definition point 1.

#### 4.19.3.6 dpz2

```
double DL_DimOrdinateData::dpz2
```

Z Coordinate of definition point 2.

#### 4.19.3.7 xtype

```
bool DL_DimOrdinateData::xtype
```

True if the dimension indicates the X-value, false for Y-value

Referenced by DL\_Dxf::writeDimOrdinate().

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

- src/dl\_entities.h

## 4.20 DL\_DimRadialData Struct Reference

Radial Dimension Data.

```
#include <dl_entities.h>
```

### Public Member Functions

- [DL\\_DimRadialData](#) (double ddp<sub>x</sub>, double ddp<sub>y</sub>, double ddp<sub>z</sub>, double dleader)  
*Constructor.*

## Public Attributes

- double [dpx](#)
- double [dpy](#)
- double [dpz](#)
- double [leader](#)

### 4.20.1 Detailed Description

Radial Dimension Data.

### 4.20.2 Constructor & Destructor Documentation

#### 4.20.2.1 DL\_DimRadialData()

```
DL_DimRadialData::DL_DimRadialData (
    double ddpx,
    double ddpy,
    double ddpz,
    double dleader ) [inline]
```

Constructor.

Parameters: see member variables.

### 4.20.3 Member Data Documentation

#### 4.20.3.1 dpx

```
double DL_DimRadialData::dpx
```

X Coordinate of definition point.

Referenced by DL\_Dxf::writeDimRadial().

#### 4.20.3.2 dpy

```
double DL_DimRadialData::dpy
```

Y Coordinate of definition point.

Referenced by DL\_Dxf::writeDimRadial().

#### 4.20.3.3 dpz

```
double DL_DimRadialData::dpz
```

Z Coordinate of definition point.

#### 4.20.3.4 leader

```
double DL_DimRadialData::leader
```

Leader length

Referenced by DL\_Dxf::writeDimRadial().

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

- src/dl\_entities.h

## 4.21 DL\_Dxf Class Reference

Reading and writing of DXF files.

```
#include <dl_dxf.h>
```

### Public Member Functions

- [DL\\_Dxf](#) ()  
*Default constructor.*
- [~DL\\_Dxf](#) ()  
*Destructor.*
- bool [in](#) (const std::string &file, [DL\\_CreationInterface](#) \*creationInterface)  
*Reads the given file and calls the appropriate functions in the given creation interface for every entity found in the file.*
- bool [readDxfGroups](#) (FILE \*fp, [DL\\_CreationInterface](#) \*creationInterface)  
*Reads a group couplet from a DXF file.*
- bool [readDxfGroups](#) (std::istream &stream, [DL\\_CreationInterface](#) \*creationInterface)  
*Same as above but for input streams.*
- bool [in](#) (std::istream &stream, [DL\\_CreationInterface](#) \*creationInterface)  
*Reads a DXF file from an existing stream.*
- bool [processDXFGroup](#) ([DL\\_CreationInterface](#) \*creationInterface, int groupCode, const std::string &group↵ Value)  
*Processes a group (pair of group code and value).*
- void [addSetting](#) ([DL\\_CreationInterface](#) \*creationInterface)  
*Adds a variable from the DXF file.*
- void [addLayer](#) ([DL\\_CreationInterface](#) \*creationInterface)  
*Adds a layer that was read from the file via the creation interface.*
- void [addLinetype](#) ([DL\\_CreationInterface](#) \*creationInterface)  
*Adds a linetype that was read from the file via the creation interface.*
- void [addBlock](#) ([DL\\_CreationInterface](#) \*creationInterface)



- Adds a block that was read from the file via the creation interface.*

  - void **endBlock** (DL\_CreationInterface \*creationInterface)
- Ends a block that was read from the file via the creation interface.*

  - void **addTextStyle** (DL\_CreationInterface \*creationInterface)
- Adds a point entity that was read from the file via the creation interface.*

  - void **addPoint** (DL\_CreationInterface \*creationInterface)
- Adds a line entity that was read from the file via the creation interface.*

  - void **addLine** (DL\_CreationInterface \*creationInterface)
- Adds an xline entity that was read from the file via the creation interface.*

  - void **addXLine** (DL\_CreationInterface \*creationInterface)
- Adds a ray entity that was read from the file via the creation interface.*

  - void **addRay** (DL\_CreationInterface \*creationInterface)
- Adds a polyline entity that was read from the file via the creation interface.*

  - void **addPolyline** (DL\_CreationInterface \*creationInterface)
- Adds a polyline vertex entity that was read from the file via the creation interface.*

  - void **addVertex** (DL\_CreationInterface \*creationInterface)
- Adds a spline entity that was read from the file via the creation interface.*

  - void **addSpline** (DL\_CreationInterface \*creationInterface)
- Adds an arc entity that was read from the file via the creation interface.*

  - void **addArc** (DL\_CreationInterface \*creationInterface)
- Adds a circle entity that was read from the file via the creation interface.*

  - void **addCircle** (DL\_CreationInterface \*creationInterface)
- Adds an ellipse entity that was read from the file via the creation interface.*

  - void **addEllipse** (DL\_CreationInterface \*creationInterface)
- Adds an insert entity that was read from the file via the creation interface.*

  - void **addInsert** (DL\_CreationInterface \*creationInterface)
- Adds a trace entity (4 edge closed polyline) that was read from the file via the creation interface.*

  - void **addTrace** (DL\_CreationInterface \*creationInterface)
- Adds a 3dface entity that was read from the file via the creation interface.*

  - void **add3dFace** (DL\_CreationInterface \*creationInterface)
- Adds a solid entity (filled trace) that was read from the file via the creation interface.*

  - void **addSolid** (DL\_CreationInterface \*creationInterface)
- Adds an MText entity that was read from the file via the creation interface.*

  - void **addMText** (DL\_CreationInterface \*creationInterface)
- Adds a text entity that was read from the file via the creation interface.*

  - void **addText** (DL\_CreationInterface \*creationInterface)
- Adds an arc aligned text entity that was read from the file via the creation interface.*

  - void **addArcAlignedText** (DL\_CreationInterface \*creationInterface)
- Adds an attrib entity that was read from the file via the creation interface.*

  - void **addAttribute** (DL\_CreationInterface \*creationInterface)
- Adds a linear dimension entity that was read from the file via the creation interface.*

  - **DL\_DimensionData** getDimData ()
- Adds an aligned dimension entity that was read from the file via the creation interface.*

  - void **addDimLinear** (DL\_CreationInterface \*creationInterface)
- Adds a radial dimension entity that was read from the file via the creation interface.*

  - void **addDimAligned** (DL\_CreationInterface \*creationInterface)
- Adds a diametric dimension entity that was read from the file via the creation interface.*

  - void **addDimRadial** (DL\_CreationInterface \*creationInterface)
- Adds an angular dimension entity that was read from the file via the creation interface.*

  - void **addDimDiametric** (DL\_CreationInterface \*creationInterface)
- Adds an angular dimension entity that was read from the file via the creation interface.*

  - void **addDimAngular** (DL\_CreationInterface \*creationInterface)

- void **addDimAngular3P** ([DL\\_CreationInterface](#) \*creationInterface)  
*Adds an angular dimension entity that was read from the file via the creation interface.*
- void **addDimOrdinate** ([DL\\_CreationInterface](#) \*creationInterface)  
*Adds an ordinate dimension entity that was read from the file via the creation interface.*
- void **addLeader** ([DL\\_CreationInterface](#) \*creationInterface)  
*Adds a leader entity that was read from the file via the creation interface.*
- void **addHatch** ([DL\\_CreationInterface](#) \*creationInterface)  
*Adds a hatch entity that was read from the file via the creation interface.*
- void **addHatchLoop** ()
- void **addHatchEdge** ()
- bool **handleHatchData** ([DL\\_CreationInterface](#) \*creationInterface)  
*Handles all hatch data.*
- void **addImage** ([DL\\_CreationInterface](#) \*creationInterface)  
*Adds an image entity that was read from the file via the creation interface.*
- void **addImageDef** ([DL\\_CreationInterface](#) \*creationInterface)  
*Adds an image definition that was read from the file via the creation interface.*
- void **addComment** ([DL\\_CreationInterface](#) \*creationInterface, const std::string &comment)  
*Adds a comment from the DXF file.*
- void **addDictionary** ([DL\\_CreationInterface](#) \*creationInterface)
- void **addDictionaryEntry** ([DL\\_CreationInterface](#) \*creationInterface)
- bool **handleXRecordData** ([DL\\_CreationInterface](#) \*creationInterface)  
*Handles all XRecord data.*
- bool **handleDictionaryData** ([DL\\_CreationInterface](#) \*creationInterface)  
*Handles all dictionary data.*
- bool **handleXData** ([DL\\_CreationInterface](#) \*creationInterface)  
*Handles XData for all object types.*
- bool **handleMTextData** ([DL\\_CreationInterface](#) \*creationInterface)  
*Handles additional MText data.*
- bool **handleLWPolylineData** ([DL\\_CreationInterface](#) \*creationInterface)  
*Handles additional polyline data.*
- bool **handleSplineData** ([DL\\_CreationInterface](#) \*creationInterface)  
*Handles additional spline data.*
- bool **handleLeaderData** ([DL\\_CreationInterface](#) \*creationInterface)  
*Handles additional leader data.*
- bool **handleLinetypeData** ([DL\\_CreationInterface](#) \*creationInterface)  
*Handles all dashes in linetype pattern.*
- void **endEntity** ([DL\\_CreationInterface](#) \*creationInterface)  
*Ends some special entities like hatches or old style polylines.*
- void **endSequence** ([DL\\_CreationInterface](#) \*creationInterface)  
*Ends a sequence and notifies the creation interface.*
- [DL\\_WriterA](#) \* **out** (const char \*file, [DL\\_Codes::version](#) version=[DL\\_VERSION\\_2000](#))  
*Converts the given string into an int.*
- void **writeHeader** ([DL\\_WriterA](#) &dw)  
*Writes a DXF header to the file currently opened by the given DXF writer object.*
- void **writePoint** ([DL\\_WriterA](#) &dw, const [DL\\_PointData](#) &data, const [DL\\_Attributes](#) &attrib)  
*Writes a point entity to the file.*
- void **writeLine** ([DL\\_WriterA](#) &dw, const [DL\\_LineData](#) &data, const [DL\\_Attributes](#) &attrib)  
*Writes a line entity to the file.*
- void **writeXLine** ([DL\\_WriterA](#) &dw, const [DL\\_XLineData](#) &data, const [DL\\_Attributes](#) &attrib)  
*Writes an x line entity to the file.*
- void **writeRay** ([DL\\_WriterA](#) &dw, const [DL\\_RayData](#) &data, const [DL\\_Attributes](#) &attrib)

- Writes a ray entity to the file.*

  - void [writePolyline](#) (DL\_WriterA &dw, const DL\_PolylineData &data, const DL\_Attributes &attrib)
- Writes a polyline entity to the file.*

  - void [writeVertex](#) (DL\_WriterA &dw, const DL\_VertexData &data)
- Writes a single vertex of a polyline to the file.*

  - void [writePolylineEnd](#) (DL\_WriterA &dw)
- Writes the polyline end.*

  - void [writeSpline](#) (DL\_WriterA &dw, const DL\_SplineData &data, const DL\_Attributes &attrib)
- Writes a spline entity to the file.*

  - void [writeControlPoint](#) (DL\_WriterA &dw, const DL\_ControlPointData &data)
- Writes a single control point of a spline to the file.*

  - void [writeFitPoint](#) (DL\_WriterA &dw, const DL\_FitPointData &data)
- Writes a single fit point of a spline to the file.*

  - void [writeKnot](#) (DL\_WriterA &dw, const DL\_KnotData &data)
- Writes a single knot of a spline to the file.*

  - void [writeCircle](#) (DL\_WriterA &dw, const DL\_CircleData &data, const DL\_Attributes &attrib)
- Writes a circle entity to the file.*

  - void [writeArc](#) (DL\_WriterA &dw, const DL\_ArcData &data, const DL\_Attributes &attrib)
- Writes an arc entity to the file.*

  - void [writeEllipse](#) (DL\_WriterA &dw, const DL\_EllipseData &data, const DL\_Attributes &attrib)
- Writes an ellipse entity to the file.*

  - void [writeSolid](#) (DL\_WriterA &dw, const DL\_SolidData &data, const DL\_Attributes &attrib)
- Writes a solid entity to the file.*

  - void [writeTrace](#) (DL\_WriterA &dw, const DL\_TraceData &data, const DL\_Attributes &attrib)
- Writes a trace entity to the file.*

  - void [write3dFace](#) (DL\_WriterA &dw, const DL\_3dFaceData &data, const DL\_Attributes &attrib)
- Writes a 3d face entity to the file.*

  - void [writeInsert](#) (DL\_WriterA &dw, const DL\_InsertData &data, const DL\_Attributes &attrib)
- Writes an insert to the file.*

  - void [writeMText](#) (DL\_WriterA &dw, const DL\_MTextData &data, const DL\_Attributes &attrib)
- Writes a multi text entity to the file.*

  - void [writeText](#) (DL\_WriterA &dw, const DL\_TextData &data, const DL\_Attributes &attrib)
- Writes a text entity to the file.*

  - void [writeAttribute](#) (DL\_WriterA &dw, const DL\_AttributeData &data, const DL\_Attributes &attrib)
- Writes a dimension style overrides entity to the file.*

  - void [writeDimStyleOverrides](#) (DL\_WriterA &dw, const DL\_DimensionData &data)
- Writes a dimension aligned entity to the file.*

  - void [writeDimAligned](#) (DL\_WriterA &dw, const DL\_DimensionData &data, const DL\_DimAlignedData &edata, const DL\_Attributes &attrib)
- Writes a linear dimension entity to the file.*

  - void [writeDimLinear](#) (DL\_WriterA &dw, const DL\_DimensionData &data, const DL\_DimLinearData &edata, const DL\_Attributes &attrib)
- Writes a radial dimension entity to the file.*

  - void [writeDimRadial](#) (DL\_WriterA &dw, const DL\_DimensionData &data, const DL\_DimRadialData &edata, const DL\_Attributes &attrib)
- Writes a diametric dimension entity to the file.*

  - void [writeDimDiametric](#) (DL\_WriterA &dw, const DL\_DimensionData &data, const DL\_DimDiametricData &edata, const DL\_Attributes &attrib)
- Writes an angular dimension entity to the file.*

  - void [writeDimAngular2L](#) (DL\_WriterA &dw, const DL\_DimensionData &data, const DL\_DimAngular2LData &edata, const DL\_Attributes &attrib)
- Writes an angular dimension entity to the file.*

  - void [writeDimAngular3P](#) (DL\_WriterA &dw, const DL\_DimensionData &data, const DL\_DimAngular3PData &edata, const DL\_Attributes &attrib)

- Writes an angular dimension entity (3 points version) to the file.*

  - void `writeDimOrdinate` (`DL_WriterA` &dw, const `DL_DimensionData` &data, const `DL_DimOrdinateData` &edata, const `DL_Attributes` &attrib)
- Writes an ordinate dimension entity to the file.*

  - void `writeLeader` (`DL_WriterA` &dw, const `DL_LeaderData` &data, const `DL_Attributes` &attrib)
- Writes a leader entity to the file.*

  - void `writeLeaderVertex` (`DL_WriterA` &dw, const `DL_LeaderVertexData` &data)
- Writes a single vertex of a leader to the file.*

  - void `writeLeaderEnd` (`DL_WriterA` &dw, const `DL_LeaderData` &data)
- Writes the beginning of a hatch entity to the file.*

  - void `writeHatch1` (`DL_WriterA` &dw, const `DL_HatchData` &data, const `DL_Attributes` &attrib)
- Writes the end of a hatch entity to the file.*

  - void `writeHatch2` (`DL_WriterA` &dw, const `DL_HatchData` &data, const `DL_Attributes` &attrib)
- Writes the beginning of a hatch loop to the file.*

  - void `writeHatchLoop1` (`DL_WriterA` &dw, const `DL_HatchLoopData` &data)
- Writes the end of a hatch loop to the file.*

  - void `writeHatchLoop2` (`DL_WriterA` &dw, const `DL_HatchLoopData` &data)
- Writes the beginning of a hatch entity to the file.*

  - void `writeHatchEdge` (`DL_WriterA` &dw, const `DL_HatchEdgeData` &data)
- Writes an image entity.*

  - unsigned long `writeImage` (`DL_WriterA` &dw, const `DL_ImageData` &data, const `DL_Attributes` &attrib)
- Writes an image definition entity.*

  - void `writeImageDef` (`DL_WriterA` &dw, int handle, const `DL_ImageData` &data)
- Writes a layer to the file.*

  - void `writeLayer` (`DL_WriterA` &dw, const `DL_LayerData` &data, const `DL_Attributes` &attrib)
- Writes a line type to the file.*

  - void `writeLinetype` (`DL_WriterA` &dw, const `DL_LinetypeData` &data)
- Writes the APPID section to the DXF file.*

  - void `writeAppid` (`DL_WriterA` &dw, const std::string &name)
- Writes a block's definition (no entities) to the DXF file.*

  - void `writeBlock` (`DL_WriterA` &dw, const `DL_BlockData` &data)
- Writes a block end.*

  - void `writeEndBlock` (`DL_WriterA` &dw, const std::string &name)
- Writes a viewport section.*

  - void `writeVPort` (`DL_WriterA` &dw)
- Writes a style section.*

  - void `writeStyle` (`DL_WriterA` &dw, const `DL_StyleData` &style)
- Writes a view section.*

  - void `writeView` (`DL_WriterA` &dw)
- Writes a ucs section.*

  - void `writeUcs` (`DL_WriterA` &dw)
- Writes a dimstyle section.*

  - void `writeDimStyle` (`DL_WriterA` &dw, double dimasz, double dimexe, double dimexo, double dimgap, double dimtxt)
- Writes a blockrecord section.*

  - void `writeBlockRecord` (`DL_WriterA` &dw)
- Writes a single block record with the given name.*

  - void `writeBlockRecord` (`DL_WriterA` &dw, const std::string &name)
- Writes a objects section.*

  - void `writeObjects` (`DL_WriterA` &dw, const std::string &appDictionaryName="")
- Writes an app dictionary section.*

  - void `writeAppDictionary` (`DL_WriterA` &dw)

- unsigned long **writeDictionaryEntry** (DL\_WriterA &dw, const std::string &name)
- void **writeXRecord** (DL\_WriterA &dw, int handle, int value)
- void **writeXRecord** (DL\_WriterA &dw, int handle, double value)
- void **writeXRecord** (DL\_WriterA &dw, int handle, bool value)
- void **writeXRecord** (DL\_WriterA &dw, int handle, const std::string &value)
- void **writeObjectsEnd** (DL\_WriterA &dw)  
*Writes the end of the objects section.*
- void **writeComment** (DL\_WriterA &dw, const std::string &comment)  
*Writes a comment to the DXF file.*
- DL\_Codes::version **getVersion** ()
- int **getLibVersion** (const std::string &str)
- bool **hasValue** (int code)
- int **getIntValue** (int code, int def)
- int **toInt** (const std::string &str)
- int **getInt16Value** (int code, int def)
- int **toInt16** (const std::string &str)
- bool **toBool** (const std::string &str)
- std::string **getStringValue** (int code, const std::string &def)
- double **getRealValue** (int code, double def)
- double **toReal** (const std::string &str)

## Static Public Member Functions

- static bool **getStrippedLine** (std::string &s, unsigned int size, FILE \*stream, bool stripSpace=true)  
*Reads line from file & strips whitespace at start and newline at end.*
- static bool **getStrippedLine** (std::string &s, unsigned int size, std::istream &stream, bool stripSpace=true)  
*Same as above but for input streams.*
- static bool **stripWhiteSpace** (char \*\*s, bool stripSpaces=true)  
*Strips leading whitespace and trailing Carriage Return (CR) and Line Feed (LF) from NULL terminated string.*
- static bool **checkVariable** (const char \*var, DL\_Codes::version version)  
*Converts the given string into a double or returns the given default valud (def) if value is NULL or empty.*
- static void **test** ()  
*Converts the given string into a double or returns the given default valud (def) if value is NULL or empty.*

### 4.21.1 Detailed Description

Reading and writing of DXF files.

This class can read in a DXF file and calls methods from the interface DL\_EntityContainer to add the entities to the container provided by the user of the library.

It can also be used to write DXF files to a certain extent.

When saving entities, special values for colors and linetypes can be used:

Special colors are 0 (=BYBLOCK) and 256 (=BYLAYER). Special linetypes are "BYLAYER" and "BYBLOCK".

#### Author

Andrew Mustun

## 4.21.2 Member Function Documentation

### 4.21.2.1 addAttribute()

```
void DL_Dxf::addAttribute (
    DL_CreationInterface * creationInterface )
```

Adds an attrib entity that was read from the file via the creation interface.

**Todo** add attrib instead of normal text

References DL\_CreationInterface::addAttribute().

Referenced by processDXFGroup().

### 4.21.2.2 addSolid()

```
void DL_Dxf::addSolid (
    DL_CreationInterface * creationInterface )
```

Adds a solid entity (filled trace) that was read from the file via the creation interface.

Author

AHM

References DL\_CreationInterface::addSolid(), and DL\_TraceData::x.

Referenced by processDXFGroup().

### 4.21.2.3 addTrace()

```
void DL_Dxf::addTrace (
    DL_CreationInterface * creationInterface )
```

Adds a trace entity (4 edge closed polyline) that was read from the file via the creation interface.

Author

AHM

References DL\_CreationInterface::addTrace(), and DL\_TraceData::x.

Referenced by processDXFGroup().

#### 4.21.2.4 checkVariable()

```
bool DL_Dxf::checkVariable (
    const char * var,
    DL_Codes::version version ) [static]
```

Converts the given string into a double or returns the given default value (def) if value is NULL or empty.

Checks if the given variable is known by the given DXF version.

Converts the given string into an int or returns the given default value (def) if value is NULL or empty. Converts the given string into a string or returns the given default value (def) if value is NULL or empty.

#### 4.21.2.5 getDimData()

```
DL_DimensionData DL_Dxf::getDimData ( )
```

##### Returns

dimension data from current values.

Referenced by addDimAligned(), addDimAngular(), addDimAngular3P(), addDimDiametric(), addDimLinear(), addDimOrdinate(), and addDimRadial().

#### 4.21.2.6 getLibVersion()

```
int DL_Dxf::getLibVersion (
    const std::string & str )
```

##### Returns

the library version as int (4 bytes, each byte one version number). e.g. if str = "2.0.2.0" getLibVersion returns 0x02000200

Referenced by processDXFGroup().

#### 4.21.2.7 getStrippedLine()

```
bool DL_Dxf::getStrippedLine (
    std::string & s,
    unsigned int size,
    FILE * fp,
    bool stripSpace = true ) [static]
```

Reads line from file & strips whitespace at start and newline at end.

## Parameters

<i>s</i>	Output Pointer to character array that chopped line will be returned in.
<i>size</i>	Size of <i>s</i> . (Including space for NULL.)
<i>fp</i>	Input Handle of input file.

## Return values

<i>true</i>	if line could be read
<i>false</i>	if <i>fp</i> is already at end of file

**Todo** Change function to use safer FreeBSD strl\* functions

Is it a problem if line is blank (i.e., newline only)? Then, when function returns, (s==NULL).

References stripWhiteSpace().

Referenced by readDxfGroups().

#### 4.21.2.8 in() [1/2]

```
bool DL_Dxf::in (
    const std::string & file,
    DL_CreationInterface * creationInterface )
```

Reads the given file and calls the appropriate functions in the given creation interface for every entity found in the file.

## Parameters

<i>file</i>	Input Path and name of file to read
<i>creationInterface</i>	Pointer to the class which takes care of the entities in the file.

## Return values

<i>true</i>	If <i>file</i> could be opened.
<i>false</i>	If <i>file</i> could not be opened.

References readDxfGroups().

#### 4.21.2.9 in() [2/2]

```
bool DL_Dxf::in (
```



```
std::istream & stream,
DL_CreationInterface * creationInterface )
```

Reads a DXF file from an existing stream.

#### Parameters

<i>stream</i>	The input stream.
<i>creationInterface</i>	Pointer to the class which takes care of the entities in the file.

#### Return values

<i>true</i>	If <code>file</code> could be opened.
<i>false</i>	If <code>file</code> could not be opened.

References `readDxfGroups()`.

#### 4.21.2.10 out()

```
DL_WriterA * DL_Dxf::out (
    const char * file,
    DL_Codes::version version = DL_VERSION_2000 )
```

Converts the given string into an int.

`ok` is set to false if there was an error.

Opens the given file for writing and returns a pointer to the dxf writer. This pointer needs to be passed on to other writing functions.

#### Parameters

<i>file</i>	Full path of the file to open.
-------------	--------------------------------

#### Returns

Pointer to an ascii dxf writer object.

References `DL_WriterA::openFailed()`.

#### 4.21.2.11 processDXFGroup()

```
bool DL_Dxf::processDXFGroup (
    DL_CreationInterface * creationInterface,
    int groupCode,
    const std::string & groupValue )
```

Processes a group (pair of group code and value).

## Parameters

<i>creationInterface</i>	Handle to class that creates entities and other CAD data from DXF group codes
<i>groupCode</i>	Constant indicating the data type of the group.
<i>groupValue</i>	The data value.

## Return values

<i>true</i>	if done processing current entity and new entity begun
<i>false</i>	if not done processing current entity

References `add3dFace()`, `addArc()`, `addArcAlignedText()`, `addAttribute()`, `addBlock()`, `addCircle()`, `addComment()`, `addDimAligned()`, `addDimAngular()`, `addDimAngular3P()`, `addDimDiametric()`, `addDimLinear()`, `addDimOrdinate()`, `addDimRadial()`, `addEllipse()`, `addImage()`, `addImageDef()`, `addInsert()`, `addLayer()`, `addLeader()`, `addLine()`, `addLinetype()`, `addMText()`, `addPoint()`, `addPolyline()`, `addRay()`, `addSetting()`, `addSolid()`, `addSpline()`, `addText()`, `addTrace()`, `addVertex()`, `addXLine()`, `endBlock()`, `endEntity()`, `DL_CreationInterface::endSection()`, `endSequence()`, `getLibVersion()`, `handleDictionaryData()`, `handleHatchData()`, `handleLeaderData()`, `handleLinetypeData()`, `handleLWPolylineData()`, `handleMTextData()`, `handleSplineData()`, `handleXData()`, `handleXRecordData()`, `DL_CreationInterface::setAttributes()`, `DL_CreationInterface::setExtrusion()`, and `DL_Attributes::setLinetypeScale()`.

Referenced by `readDxfGroups()`.

## 4.21.2.12 readDxfGroups()

```
bool DL_Dxf::readDxfGroups (
    FILE * fp,
    DL_CreationInterface * creationInterface )
```

Reads a group couplet from a DXF file.

Calls another function to process it.

A group couplet consists of two lines that represent a single piece of data. An integer constant on the first line indicates the type of data. The value is on the next line.

This function reads a couplet, determines the type of data, and passes the value to the the appropriate handler function of `creationInterface`.

`fp` is advanced so that the next call to `readDXFGroups()` reads the next couplet in the file.

## Parameters

<i>fp</i>	Handle of input file
<i>creationInterface</i>	Handle of class which processes entities in the file

## Return values

<i>true</i>	If EOF not reached.
<i>false</i>	If EOF reached.

References `getStrippedLine()`, `DL_CreationInterface::processCodeValuePair()`, and `processDXFGroup()`.

Referenced by in().

#### 4.21.2.13 stripWhiteSpace()

```
bool DL_Dxf::stripWhiteSpace (
    char ** s,
    bool stripSpace = true ) [static]
```

Strips leading whitespace and trailing Carriage Return (CR) and Line Feed (LF) from NULL terminated string.

##### Parameters

<i>s</i>	Input and output. NULL terminates string.
----------	---

##### Return values

<i>true</i>	if <i>s</i> is non-NULL
<i>false</i>	if <i>s</i> is NULL

Referenced by getStrippedLine(), and test().

#### 4.21.2.14 test()

```
void DL_Dxf::test ( ) [static]
```

Converts the given string into a double or returns the given default value (def) if value is NULL or empty.

Some test routines.

References stripWhiteSpace().

#### 4.21.2.15 write3dFace()

```
void DL_Dxf::write3dFace (
    DL_WriterA & dw,
    const DL_3dFaceData & data,
    const DL_Attributes & attrib )
```

Writes a 3d face entity to the file.

##### Parameters

<i>dw</i>	DXF writer
<i>data</i>	Entity data from the file
<i>attrib</i>	Attributes

References `DL_WriterA::dxflnt()`, `DL_Writer::entity()`, `DL_Writer::entityAttributes()`, and `DL_TraceData::x`.

#### 4.21.2.16 writeAppid()

```
void DL_Dxf::writeAppid (
    DL_WriterA & dw,
    const std::string & name )
```

Writes the APPID section to the DXF file.

##### Parameters

<i>name</i>	Application name
-------------	------------------

References `DL_WriterA::dxflnt()`, `DL_WriterA::dxflnt()`, and `DL_Writer::tableAppidEntry()`.

#### 4.21.2.17 writeArc()

```
void DL_Dxf::writeArc (
    DL_WriterA & dw,
    const DL_ArcData & data,
    const DL_Attributes & attrib )
```

Writes an arc entity to the file.

##### Parameters

<i>dw</i>	DXF writer
<i>data</i>	Entity data from the file
<i>attrib</i>	Attributes

References `DL_ArcData::angle1`, `DL_ArcData::angle2`, `DL_ArcData::cx`, `DL_ArcData::cy`, `DL_ArcData::cz`, `DL_WriterA::dxflnt()`, `DL_WriterA::dxflnt()`, `DL_Writer::entity()`, `DL_Writer::entityAttributes()`, and `DL_ArcData::radius`.

#### 4.21.2.18 writeBlockRecord()

```
void DL_Dxf::writeBlockRecord (
    DL_WriterA & dw )
```

Writes a blockrecord section.

This section is needed in `DL_VERSION_R13`. Note that this method currently only writes a faked BLOCKRECORD section to make the file readable by Autocad.

References `DL_WriterA::dxflnt()`, `DL_WriterA::dxflnt()`, and `DL_WriterA::dxflnt()`.

**4.21.2.19 writeCircle()**

```
void DL_Dxf::writeCircle (
    DL_WriterA & dw,
    const DL_CircleData & data,
    const DL_Attributes & attrib )
```

Writes a circle entity to the file.

**Parameters**

<i>dw</i>	DXF writer
<i>data</i>	Entity data from the file
<i>attrib</i>	Attributes

References DL\_CircleData::cx, DL\_CircleData::cy, DL\_CircleData::cz, DL\_WriterA::dxfReal(), DL\_WriterA::dxfString(), DL\_Writer::entity(), DL\_Writer::entityAttributes(), and DL\_CircleData::radius.

**4.21.2.20 writeControlPoint()**

```
void DL_Dxf::writeControlPoint (
    DL_WriterA & dw,
    const DL_ControlPointData & data )
```

Writes a single control point of a spline to the file.

**Parameters**

<i>dw</i>	DXF writer
<i>data</i>	Entity data from the file
<i>attrib</i>	Attributes

References DL\_WriterA::dxfReal(), DL\_ControlPointData::x, DL\_ControlPointData::y, and DL\_ControlPointData::z.

**4.21.2.21 writeDimAligned()**

```
void DL_Dxf::writeDimAligned (
    DL_WriterA & dw,
    const DL_DimensionData & data,
    const DL_DimAlignedData & edata,
    const DL_Attributes & attrib )
```

Writes an aligned dimension entity to the file.

**Parameters**

<i>dw</i>	DXF writer
<i>data</i>	Generic dimension data for from the file
<i>edata</i>	Specific aligned dimension data from the file
<i>attrib</i>	Attributes

References `DL_DimensionData::angle`, `DL_DimensionData::attachmentPoint`, `DL_DimensionData::dpx`, `DL_DimensionData::dpy`, `DL_DimensionData::dpz`, `DL_WriterA::dxflnt()`, `DL_WriterA::dxflReal()`, `DL_WriterA::dxflString()`, `DL_Writer::entity()`, `DL_Writer::entityAttributes()`, `DL_DimAlignedData::epx1`, `DL_DimAlignedData::epx2`, `DL_DimAlignedData::epy1`, `DL_DimAlignedData::epy2`, `DL_DimensionData::lineSpacingFactor`, `DL_DimensionData::lineSpacingStyle`, `DL_DimensionData::mpx`, `DL_DimensionData::mpy`, `DL_DimensionData::text`, and `DL_DimensionData::type`.

#### 4.21.2.22 writeDimAngular2L()

```
void DL_Dxf::writeDimAngular2L (
    DL_WriterA & dw,
    const DL_DimensionData & data,
    const DL_DimAngular2LData & edata,
    const DL_Attributes & attrib )
```

Writes an angular dimension entity to the file.

##### Parameters

<i>dw</i>	DXF writer
<i>data</i>	Generic dimension data for from the file
<i>data</i>	Specific angular dimension data from the file
<i>attrib</i>	Attributes

References `DL_DimensionData::angle`, `DL_DimensionData::attachmentPoint`, `DL_DimensionData::dpx`, `DL_DimAngular2LData::dpx1`, `DL_DimAngular2LData::dpx2`, `DL_DimAngular2LData::dpx3`, `DL_DimAngular2LData::dpx4`, `DL_DimensionData::dpy`, `DL_DimAngular2LData::dpy1`, `DL_DimAngular2LData::dpy2`, `DL_DimAngular2LData::dpy3`, `DL_DimAngular2LData::dpy4`, `DL_DimensionData::dpz`, `DL_WriterA::dxflnt()`, `DL_WriterA::dxflReal()`, `DL_WriterA::dxflString()`, `DL_Writer::entity()`, `DL_Writer::entityAttributes()`, `DL_DimensionData::lineSpacingFactor`, `DL_DimensionData::lineSpacingStyle`, `DL_DimensionData::mpx`, `DL_DimensionData::mpy`, `DL_DimensionData::text`, and `DL_DimensionData::type`.

#### 4.21.2.23 writeDimAngular3P()

```
void DL_Dxf::writeDimAngular3P (
    DL_WriterA & dw,
    const DL_DimensionData & data,
    const DL_DimAngular3PData & edata,
    const DL_Attributes & attrib )
```

Writes an angular dimension entity (3 points version) to the file.

##### Parameters

<i>dw</i>	DXF writer
<i>data</i>	Generic dimension data for from the file
<i>data</i>	Specific angular dimension data from the file
<i>attrib</i>	Attributes

References DL\_DimensionData::angle, DL\_DimensionData::attachmentPoint, DL\_DimensionData::dpx, DL\_DimAngular3PData::dpx1, DL\_DimAngular3PData::dpx2, DL\_DimAngular3PData::dpx3, DL\_DimensionData::dpy, DL\_DimAngular3PData::dpy1, DL\_DimAngular3PData::dpy2, DL\_DimAngular3PData::dpy3, DL\_DimensionData::dpz, DL\_WriterA::dxflnt(), DL\_WriterA::dxflReal(), DL\_WriterA::dxflString(), DL\_Writer::entity(), DL\_Writer::entityAttributes(), DL\_DimensionData::lineSpacingFactor, DL\_DimensionData::lineSpacingStyle, DL\_DimensionData::mpx, DL\_DimensionData::mpy, DL\_DimensionData::text, and DL\_DimensionData::type.

#### 4.21.2.24 writeDimDiametric()

```
void DL_Dxf::writeDimDiametric (
    DL_WriterA & dw,
    const DL_DimensionData & data,
    const DL_DimDiametricData & edata,
    const DL_Attributes & attrib )
```

Writes a diametric dimension entity to the file.

##### Parameters

<i>dw</i>	DXF writer
<i>data</i>	Generic dimension data for from the file
<i>data</i>	Specific diametric dimension data from the file
<i>attrib</i>	Attributes

References DL\_DimensionData::angle, DL\_DimensionData::attachmentPoint, DL\_DimensionData::dpx, DL\_DimDiametricData::dpx, DL\_DimensionData::dpy, DL\_DimDiametricData::dpy, DL\_DimensionData::dpz, DL\_WriterA::dxflnt(), DL\_WriterA::dxflReal(), DL\_WriterA::dxflString(), DL\_Writer::entity(), DL\_Writer::entityAttributes(), DL\_DimDiametricData::leader, DL\_DimensionData::lineSpacingFactor, DL\_DimensionData::lineSpacingStyle, DL\_DimensionData::mpx, DL\_DimensionData::mpy, DL\_DimensionData::text, and DL\_DimensionData::type.

#### 4.21.2.25 writeDimLinear()

```
void DL_Dxf::writeDimLinear (
    DL_WriterA & dw,
    const DL_DimensionData & data,
    const DL_DimLinearData & edata,
    const DL_Attributes & attrib )
```

Writes a linear dimension entity to the file.

##### Parameters

<i>dw</i>	DXF writer
<i>data</i>	Generic dimension data for from the file
<i>data</i>	Specific linear dimension data from the file
<i>attrib</i>	Attributes

References `DL_DimensionData::angle`, `DL_DimLinearData::angle`, `DL_DimensionData::attachmentPoint`, `DL_DimensionData::dpx`, `DL_DimLinearData::dpx1`, `DL_DimLinearData::dpx2`, `DL_DimensionData::dpy`, `DL_DimLinearData::dpy1`, `DL_DimLinearData::dpy2`, `DL_DimensionData::dpz`, `DL_WriterA::dxflnt()`, `DL_WriterA::dxflReal()`, `DL_WriterA::dxflString()`, `DL_Writer::entity()`, `DL_Writer::entityAttributes()`, `DL_DimensionData::lineSpacingFactor`, `DL_DimensionData::lineSpacingStyle`, `DL_DimensionData::mpx`, `DL_DimensionData::mpy`, `DL_DimensionData::text`, and `DL_DimensionData::type`.

#### 4.21.2.26 writeDimOrdinate()

```
void DL_Dxf::writeDimOrdinate (
    DL_WriterA & dw,
    const DL_DimensionData & data,
    const DL_DimOrdinateData & edata,
    const DL_Attributes & attrib )
```

Writes an ordinate dimension entity to the file.

##### Parameters

<i>dw</i>	DXF writer
<i>data</i>	Generic dimension data for from the file
<i>data</i>	Specific ordinate dimension data from the file
<i>attrib</i>	Attributes

References `DL_DimensionData::attachmentPoint`, `DL_DimensionData::dpx`, `DL_DimOrdinateData::dpx1`, `DL_DimOrdinateData::dpx2`, `DL_DimensionData::dpy`, `DL_DimOrdinateData::dpy1`, `DL_DimOrdinateData::dpy2`, `DL_DimensionData::dpz`, `DL_WriterA::dxflnt()`, `DL_WriterA::dxflReal()`, `DL_WriterA::dxflString()`, `DL_Writer::entity()`, `DL_Writer::entityAttributes()`, `DL_DimensionData::lineSpacingFactor`, `DL_DimensionData::lineSpacingStyle`, `DL_DimensionData::mpx`, `DL_DimensionData::mpy`, `DL_DimensionData::text`, `DL_DimensionData::type`, and `DL_DimOrdinateData::xtype`.

#### 4.21.2.27 writeDimRadial()

```
void DL_Dxf::writeDimRadial (
    DL_WriterA & dw,
    const DL_DimensionData & data,
    const DL_DimRadialData & edata,
    const DL_Attributes & attrib )
```

Writes a radial dimension entity to the file.

##### Parameters

<i>dw</i>	DXF writer
<i>data</i>	Generic dimension data for from the file
<i>data</i>	Specific radial dimension data from the file
<i>attrib</i>	Attributes



References `DL_DimensionData::angle`, `DL_DimensionData::attachmentPoint`, `DL_DimensionData::dpx`, `DL_DimRadialData::dpx`, `DL_DimensionData::dpy`, `DL_DimRadialData::dpy`, `DL_DimensionData::dpz`, `DL_WriterA::dxfInt()`, `DL_WriterA::dxfReal()`, `DL_WriterA::dxfString()`, `DL_Writer::entity()`, `DL_Writer::entityAttributes()`, `DL_DimRadialData::leader`, `DL_DimensionData::lineSpacingFactor`, `DL_DimensionData::lineSpacingStyle`, `DL_DimensionData::mpx`, `DL_DimensionData::mpy`, `DL_DimensionData::text`, and `DL_DimensionData::type`.

#### 4.21.2.28 writeDimStyle()

```
void DL_Dxf::writeDimStyle (
    DL_WriterA & dw,
    double dimasz,
    double dimexe,
    double dimexo,
    double dimgap,
    double dimtxt )
```

Writes a dimstyle section.

This section is needed in `DL_VERSION_R13`. Note that this method currently only writes a faked DIMSTYLE section to make the file readable by AutoCAD.

References `DL_WriterA::dxfHex()`, `DL_WriterA::dxfInt()`, `DL_WriterA::dxfReal()`, and `DL_WriterA::dxfString()`.

#### 4.21.2.29 writeEllipse()

```
void DL_Dxf::writeEllipse (
    DL_WriterA & dw,
    const DL_EllipseData & data,
    const DL_Attributes & attrib )
```

Writes an ellipse entity to the file.

##### Parameters

<i>dw</i>	DXF writer
<i>data</i>	Entity data from the file
<i>attrib</i>	Attributes

References `DL_EllipseData::angle1`, `DL_EllipseData::angle2`, `DL_EllipseData::cx`, `DL_EllipseData::cy`, `DL_EllipseData::cz`, `DL_WriterA::dxfReal()`, `DL_WriterA::dxfString()`, `DL_Writer::entity()`, `DL_Writer::entityAttributes()`, `DL_EllipseData::mx`, `DL_EllipseData::my`, `DL_EllipseData::mz`, and `DL_EllipseData::ratio`.

#### 4.21.2.30 writeEndBlock()

```
void DL_Dxf::writeEndBlock (
    DL_WriterA & dw,
    const std::string & name )
```

Writes a block end.

#### Parameters

<i>name</i>	Block name
-------------	------------

References `DL_Writer::sectionBlockEntryEnd()`.

#### 4.21.2.31 writeFitPoint()

```
void DL_Dxf::writeFitPoint (
    DL_WriterA & dw,
    const DL_FitPointData & data )
```

Writes a single fit point of a spline to the file.

#### Parameters

<i>dw</i>	DXF writer
<i>data</i>	Entity data from the file
<i>attrib</i>	Attributes

References `DL_WriterA::dxfReal()`, `DL_FitPointData::x`, `DL_FitPointData::y`, and `DL_FitPointData::z`.

#### 4.21.2.32 writeHatch1()

```
void DL_Dxf::writeHatch1 (
    DL_WriterA & dw,
    const DL_HatchData & data,
    const DL_Attributes & attrib )
```

Writes the beginning of a hatch entity to the file.

This must be followed by one or more `writeHatchLoop()` calls and a `writeHatch2()` call.

#### Parameters

<i>dw</i>	DXF writer
<i>data</i>	Entity data.
<i>attrib</i>	Attributes

References `DL_WriterA::dxfInt()`, `DL_WriterA::dxfReal()`, `DL_WriterA::dxfString()`, `DL_Writer::entity()`, `DL_WriterA::entityAttributes()`, `DL_HatchData::numLoops`, `DL_HatchData::pattern`, and `DL_HatchData::solid`.

**4.21.2.33 writeHatch2()**

```
void DL_Dxf::writeHatch2 (
    DL_WriterA & dw,
    const DL_HatchData & data,
    const DL_Attributes & attrib )
```

Writes the end of a hatch entity to the file.

**Parameters**

<i>dw</i>	DXF writer
<i>data</i>	Entity data.
<i>attrib</i>	Attributes

References DL\_HatchData::angle, DL\_WriterA::dxflnt(), DL\_WriterA::dxflReal(), DL\_WriterA::dxflString(), DL\_HatchData::originX, DL\_HatchData::scale, and DL\_HatchData::solid.

**4.21.2.34 writeHatchEdge()**

```
void DL_Dxf::writeHatchEdge (
    DL_WriterA & dw,
    const DL_HatchEdgeData & data )
```

Writes the beginning of a hatch entity to the file.

**Parameters**

<i>dw</i>	DXF writer
<i>data</i>	Entity data.
<i>attrib</i>	Attributes

References DL\_HatchEdgeData::angle1, DL\_HatchEdgeData::angle2, DL\_HatchEdgeData::ccw, DL\_HatchEdgeData::cx, DL\_HatchEdgeData::cy, DL\_HatchEdgeData::degree, DL\_Writer::dxflBool(), DL\_WriterA::dxflInt(), DL\_WriterA::dxflReal(), DL\_HatchEdgeData::mx, DL\_HatchEdgeData::my, DL\_HatchEdgeData::nControl, DL\_HatchEdgeData::nFit, DL\_HatchEdgeData::nKnots, DL\_HatchEdgeData::radius, DL\_HatchEdgeData::ratio, DL\_HatchEdgeData::type, DL\_HatchEdgeData::x1, DL\_HatchEdgeData::x2, DL\_HatchEdgeData::y1, and DL\_HatchEdgeData::y2.

**4.21.2.35 writeHatchLoop1()**

```
void DL_Dxf::writeHatchLoop1 (
    DL_WriterA & dw,
    const DL_HatchLoopData & data )
```

Writes the beginning of a hatch loop to the file.

This must happen after writing the beginning of a hatch entity.

## Parameters

<i>dw</i>	DXF writer
<i>data</i>	Entity data.
<i>attrib</i>	Attributes

References `DL_WriterA::dxflnt()`, and `DL_HatchLoopData::numEdges`.

**4.21.2.36 writeHatchLoop2()**

```
void DL_Dxf::writeHatchLoop2 (
    DL_WriterA & dw,
    const DL_HatchLoopData & data )
```

Writes the end of a hatch loop to the file.

## Parameters

<i>dw</i>	DXF writer
<i>data</i>	Entity data.
<i>attrib</i>	Attributes

References `DL_WriterA::dxflnt()`.

**4.21.2.37 writeImage()**

```
unsigned long DL_Dxf::writeImage (
    DL_WriterA & dw,
    const DL_ImageData & data,
    const DL_Attributes & attrib )
```

Writes an image entity.

## Returns

IMAGEDEF handle. Needed for the IMAGEDEF counterpart.

References `DL_ImageData::brightness`, `DL_ImageData::contrast`, `DL_WriterA::dxflnt()`, `DL_WriterA::dxflReal()`, `DL_WriterA::dxflString()`, `DL_Writer::entity()`, `DL_Writer::entityAttributes()`, `DL_ImageData::fade`, `DL_WriterA::handle()`, `DL_ImageData::height`, `DL_ImageData::ipx`, `DL_ImageData::ipy`, `DL_ImageData::ipz`, `DL_ImageData::ux`, `DL_ImageData::uy`, `DL_ImageData::uz`, `DL_ImageData::vx`, `DL_ImageData::vy`, `DL_ImageData::vz`, and `DL_ImageData::width`.

#### 4.21.2.38 writeInsert()

```
void DL_Dxf::writeInsert (
    DL_WriterA & dw,
    const DL_InsertData & data,
    const DL_Attributes & attrib )
```

Writes an insert to the file.

## Parameters

<i>dw</i>	DXF writer
<i>data</i>	Entity data from the file
<i>attrib</i>	Attributes

References DL\_InsertData::angle, DL\_InsertData::cols, DL\_InsertData::colSp, DL\_WriterA::dxflnt(), DL\_WriterA::dxflReal(), DL\_WriterA::dxflString(), DL\_Writer::entity(), DL\_Writer::entityAttributes(), DL\_InsertData::ipx, DL\_InsertData::ipy, DL\_InsertData::ipz, DL\_InsertData::name, DL\_InsertData::rows, DL\_InsertData::rowSp, DL\_InsertData::sx, and DL\_InsertData::sy.

**4.21.2.39 writeKnot()**

```
void DL_Dxf::writeKnot (
    DL_WriterA & dw,
    const DL_KnotData & data )
```

Writes a single knot of a spline to the file.

## Parameters

<i>dw</i>	DXF writer
<i>data</i>	Entity data from the file
<i>attrib</i>	Attributes

References DL\_WriterA::dxflReal(), and DL\_KnotData::k.

**4.21.2.40 writeLayer()**

```
void DL_Dxf::writeLayer (
    DL_WriterA & dw,
    const DL_LayerData & data,
    const DL_Attributes & attrib )
```

Writes a layer to the file.

Layers are stored in the tables section of a DXF file.

## Parameters

<i>dw</i>	DXF writer
<i>data</i>	Entity data from the file
<i>attrib</i>	Attributes

References DL\_WriterA::dxflHex(), DL\_WriterA::dxflnt(), DL\_WriterA::dxflString(), DL\_LayerData::flags, DL\_

Attributes::getColor(), DL\_Attributes::getColor24(), DL\_Attributes::getLinetype(), DL\_Attributes::getWidth(), DL\_LayerData::name, DL\_LayerData::off, and DL\_Writer::tableLayerEntry().

#### 4.21.2.41 writeLeader()

```
void DL_Dxf::writeLeader (
    DL_WriterA & dw,
    const DL_LeaderData & data,
    const DL_Attributes & attrib )
```

Writes a leader entity to the file.

##### Parameters

<i>dw</i>	DXF writer
<i>data</i>	Entity data from the file
<i>attrib</i>	Attributes

##### See also

[writeVertex](#)

References DL\_LeaderData::arrowHeadFlag, DL\_WriterA::dxflnt(), DL\_WriterA::dxflreal(), DL\_WriterA::dxflstring(), DL\_Writer::entity(), DL\_Writer::entityAttributes(), DL\_LeaderData::hooklineDirectionFlag, DL\_LeaderData::hooklineFlag, DL\_LeaderData::leaderCreationFlag, DL\_LeaderData::leaderPathType, DL\_LeaderData::number, DL\_LeaderData::textAnnotationHeight, and DL\_LeaderData::textAnnotationWidth.

#### 4.21.2.42 writeLeaderVertex()

```
void DL_Dxf::writeLeaderVertex (
    DL_WriterA & dw,
    const DL_LeaderVertexData & data )
```

Writes a single vertex of a leader to the file.

##### Parameters

<i>dw</i>	DXF writer
<i>data</i>	Entity data

References DL\_WriterA::dxflreal(), DL\_LeaderVertexData::x, and DL\_LeaderVertexData::y.

#### 4.21.2.43 writeLine()

```
void DL_Dxf::writeLine (
```

```
DL_WriterA & dw,
const DL_LineData & data,
const DL_Attributes & attrib )
```

Writes a line entity to the file.

#### Parameters

<i>dw</i>	DXF writer
<i>data</i>	Entity data from the file
<i>attrib</i>	Attributes

References DL\_WriterA::dxfString(), DL\_Writer::entity(), DL\_Writer::entityAttributes(), DL\_LineData::x1, DL\_LineData::x2, DL\_LineData::y1, DL\_LineData::y2, DL\_LineData::z1, and DL\_LineData::z2.

#### 4.21.2.44 writeLinetype()

```
void DL_Dxf::writeLinetype (
    DL_WriterA & dw,
    const DL_LinetypeData & data )
```

Writes a line type to the file.

Line types are stored in the tables section of a DXF file.

References DL\_LinetypeData::description, DL\_WriterA::dxflnt(), DL\_WriterA::dxflreal(), DL\_WriterA::dxflstring(), DL\_LinetypeData::flags, DL\_LinetypeData::name, DL\_LinetypeData::numberOfDashes, DL\_LinetypeData::pattern, DL\_LinetypeData::patternLength, and DL\_Writer::tableLinetypeEntry().

#### 4.21.2.45 writeMText()

```
void DL_Dxf::writeMText (
    DL_WriterA & dw,
    const DL_MTextData & data,
    const DL_Attributes & attrib )
```

Writes a multi text entity to the file.

#### Parameters

<i>dw</i>	DXF writer
<i>data</i>	Entity data from the file
<i>attrib</i>	Attributes

References DL\_MTextData::angle, DL\_MTextData::attachmentPoint, DL\_MTextData::drawingDirection, DL\_WriterA::dxflnt(), DL\_WriterA::dxflreal(), DL\_WriterA::dxflstring(), DL\_Writer::entity(), DL\_Writer::entityAttributes(),



DL\_MTextData::height, DL\_MTextData::ipx, DL\_MTextData::ipy, DL\_MTextData::ipz, DL\_MTextData::lineSpacing←Factor, DL\_MTextData::lineSpacingStyle, DL\_MTextData::style, DL\_MTextData::text, and DL\_MTextData::width.

#### 4.21.2.46 writeObjects()

```
void DL_Dxf::writeObjects (
    DL_WriterA & dw,
    const std::string & appDictionaryName = "" )
```

Writes a objects section.

This section is needed in DL\_VERSION\_R13. Note that this method currently only writes a faked OBJECTS section to make the file readable by Aut\*cad.

References DL\_WriterA::dxflHex(), DL\_WriterA::dxflInt(), DL\_WriterA::dxflReal(), DL\_WriterA::dxflString(), DL\_WriterA::getNextHandle(), and DL\_WriterA::handle().

#### 4.21.2.47 writeObjectsEnd()

```
void DL_Dxf::writeObjectsEnd (
    DL_WriterA & dw )
```

Writes the end of the objects section.

This section is needed in DL\_VERSION\_R13. Note that this method currently only writes a faked OBJECTS section to make the file readable by Aut\*cad.

References DL\_WriterA::dxfString().

#### 4.21.2.48 writePoint()

```
void DL_Dxf::writePoint (
    DL_WriterA & dw,
    const DL_PointData & data,
    const DL_Attributes & attrib )
```

Writes a point entity to the file.

## Parameters

<i>dw</i>	DXF writer
<i>data</i>	Entity data from the file
<i>attrib</i>	Attributes

References DL\_WriterA::dxfString(), DL\_Writer::entity(), DL\_Writer::entityAttributes(), DL\_PointData::x, DL\_Point↵

Data::y, and DL\_PointData::z.

#### 4.21.2.49 writePolyline()

```
void DL_Dxf::writePolyline (
    DL_WriterA & dw,
    const DL_PolylineData & data,
    const DL_Attributes & attrib )
```

Writes a polyline entity to the file.

##### Parameters

<i>dw</i>	DXF writer
<i>data</i>	Entity data from the file
<i>attrib</i>	Attributes

##### See also

[writeVertex](#)

References DL\_WriterA::dxflnt(), DL\_WriterA::dxflString(), DL\_Writer::entity(), DL\_Writer::entityAttributes(), DL\_PolylineData::flags, DL\_Attributes::getLayer(), and DL\_PolylineData::number.

#### 4.21.2.50 writePolylineEnd()

```
void DL_Dxf::writePolylineEnd (
    DL_WriterA & dw )
```

Writes the polyline end.

Only needed for DXF R12.

References DL\_Writer::entity().

#### 4.21.2.51 writeRay()

```
void DL_Dxf::writeRay (
    DL_WriterA & dw,
    const DL_RayData & data,
    const DL_Attributes & attrib )
```

Writes a ray entity to the file.

## Parameters

<i>dw</i>	DXF writer
<i>data</i>	Entity data from the file
<i>attrib</i>	Attributes

References DL\_RayData::bx, DL\_RayData::by, DL\_RayData::bz, DL\_RayData::dx, DL\_WriterA::dxfString(), DL\_RayData::dy, DL\_RayData::dz, DL\_Writer::entity(), and DL\_Writer::entityAttributes().

**4.21.2.52 writeSolid()**

```
void DL_Dxf::writeSolid (
    DL_WriterA & dw,
    const DL_SolidData & data,
    const DL_Attributes & attrib )
```

Writes a solid entity to the file.

## Parameters

<i>dw</i>	DXF writer
<i>data</i>	Entity data from the file
<i>attrib</i>	Attributes

References DL\_WriterA::dxfReal(), DL\_WriterA::dxfString(), DL\_Writer::entity(), DL\_Writer::entityAttributes(), DL\_TraceData::thickness, and DL\_TraceData::x.

**4.21.2.53 writeSpline()**

```
void DL_Dxf::writeSpline (
    DL_WriterA & dw,
    const DL_SplineData & data,
    const DL_Attributes & attrib )
```

Writes a spline entity to the file.

## Parameters

<i>dw</i>	DXF writer
<i>data</i>	Entity data from the file
<i>attrib</i>	Attributes

## See also

[writeControlPoint](#)

References `DL_SplineData::degree`, `DL_WriterA::dxflnt()`, `DL_WriterA::dxfString()`, `DL_Writer::entity()`, `DL_Writer::entityAttributes()`, `DL_SplineData::flags`, `DL_SplineData::nControl`, `DL_SplineData::nFit`, and `DL_SplineData::nKnots`.

#### 4.21.2.54 writeStyle()

```
void DL_Dxf::writeStyle (
    DL_WriterA & dw,
    const DL_StyleData & style )
```

Writes a style section.

This section is needed in `DL_VERSION_R13`.

References `DL_StyleData::bigFontFile`, `DL_WriterA::dxflnt()`, `DL_WriterA::dxfReal()`, `DL_WriterA::dxfString()`, `DL_StyleData::fixedTextHeight`, `DL_StyleData::flags`, `DL_Writer::handle()`, `DL_StyleData::lastHeightUsed`, `DL_StyleData::name`, `DL_StyleData::obliqueAngle`, `DL_StyleData::primaryFontFile`, `DL_StyleData::textGenerationFlags`, and `DL_StyleData::widthFactor`.

#### 4.21.2.55 writeText()

```
void DL_Dxf::writeText (
    DL_WriterA & dw,
    const DL_TextData & data,
    const DL_Attributes & attrib )
```

Writes a text entity to the file.

##### Parameters

<i>dw</i>	DXF writer
<i>data</i>	Entity data from the file
<i>attrib</i>	Attributes

References `DL_TextData::angle`, `DL_TextData::apx`, `DL_TextData::apy`, `DL_TextData::apz`, `DL_WriterA::dxflnt()`, `DL_WriterA::dxfReal()`, `DL_WriterA::dxfString()`, `DL_Writer::entity()`, `DL_Writer::entityAttributes()`, `DL_TextData::height`, `DL_TextData::hJustification`, `DL_TextData::ipx`, `DL_TextData::ipy`, `DL_TextData::ipz`, `DL_TextData::style`, `DL_TextData::text`, `DL_TextData::textGenerationFlags`, `DL_TextData::vJustification`, and `DL_TextData::xScaleFactor`.

#### 4.21.2.56 writeTrace()

```
void DL_Dxf::writeTrace (
    DL_WriterA & dw,
```

```
const DL_TraceData & data,  
const DL_Attributes & attrib )
```

Writes a trace entity to the file.

## Parameters

<i>dw</i>	DXF writer
<i>data</i>	Entity data from the file
<i>attrib</i>	Attributes

References DL\_WriterA::dxReal(), DL\_WriterA::dxString(), DL\_Writer::entity(), DL\_Writer::entityAttributes(), DL\_↵\_TraceData::thickness, and DL\_TraceData::x.

**4.21.2.57 writeUcs()**

```
void DL_Dxf::writeUcs (
    DL_WriterA & dw )
```

Writes a ucs section.

This section is needed in DL\_VERSION\_R13. Note that this method currently only writes a faked UCS section to make the file readable by Aut\*cad.

References DL\_WriterA::dxHex(), DL\_WriterA::dxflnt(), and DL\_WriterA::dxString().

**4.21.2.58 writeVertex()**

```
void DL_Dxf::writeVertex (
    DL_WriterA & dw,
    const DL_VertexData & data )
```

Writes a single vertex of a polyline to the file.

## Parameters

<i>dw</i>	DXF writer
<i>data</i>	Entity data from the file
<i>attrib</i>	Attributes

References DL\_VertexData::bulge, DL\_WriterA::dxReal(), DL\_WriterA::dxString(), DL\_Writer::entity(), DL\_↵\_VertexData::x, DL\_VertexData::y, and DL\_VertexData::z.

**4.21.2.59 writeView()**

```
void DL_Dxf::writeView (
    DL_WriterA & dw )
```

Writes a view section.

This section is needed in DL\_VERSION\_R13. Note that this method currently only writes a faked VIEW section to make the file readable by Aut\*cad.

References DL\_WriterA::dxHex(), DL\_WriterA::dxflnt(), and DL\_WriterA::dxString().

#### 4.21.2.60 writeVPort()

```
void DL_Dxf::writeVPort (
    DL_WriterA & dw )
```

Writes a viewport section.

This section is needed in DL\_VERSION\_R13. Note that this method currently only writes a faked VPORT section to make the file readable by Aut\*cad.

References DL\_WriterA::dxHex(), DL\_WriterA::dxflnt(), DL\_WriterA::dxReal(), DL\_WriterA::dxString(), and DL\_WriterA::handle().

#### 4.21.2.61 writeXLine()

```
void DL_Dxf::writeXLine (
    DL_WriterA & dw,
    const DL_XLineData & data,
    const DL_Attributes & attrib )
```

Writes an x line entity to the file.

##### Parameters

<i>dw</i>	DXF writer
<i>data</i>	Entity data from the file
<i>attrib</i>	Attributes

References DL\_XLineData::bx, DL\_XLineData::by, DL\_XLineData::bz, DL\_XLineData::dx, DL\_WriterA::dxString(), DL\_XLineData::dy, DL\_XLineData::dz, DL\_WriterA::entity(), and DL\_WriterA::entityAttributes().

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

- src/dl\_dxf.h
- src/dl\_dxf.cpp

## 4.22 DL\_EllipseData Struct Reference

Ellipse Data.

```
#include <dl_entities.h>
```

## Public Member Functions

- [DL\\_EllipseData](#) (double [cx](#), double [cy](#), double [cz](#), double [mx](#), double [my](#), double [mz](#), double [ratio](#), double [angle1](#), double [angle2](#))

*Constructor.*

## Public Attributes

- double [cx](#)
- double [cy](#)
- double [cz](#)
- double [mx](#)
- double [my](#)
- double [mz](#)
- double [ratio](#)
- double [angle1](#)
- double [angle2](#)

### 4.22.1 Detailed Description

Ellipse Data.

### 4.22.2 Constructor & Destructor Documentation

#### 4.22.2.1 DL\_EllipseData()

```
DL_EllipseData::DL_EllipseData (  
    double cx,  
    double cy,  
    double cz,  
    double mx,  
    double my,  
    double mz,  
    double ratio,  
    double angle1,  
    double angle2 ) [inline]
```

Constructor.

Parameters: see member variables.

### 4.22.3 Member Data Documentation



#### 4.22.3.1 angle1

```
double DL_EllipseData::angle1
```

Startangle of ellipse in rad.

Referenced by DL\_Dxf::writeEllipse().

#### 4.22.3.2 angle2

```
double DL_EllipseData::angle2
```

Endangle of ellipse in rad.

Referenced by DL\_Dxf::writeEllipse().

#### 4.22.3.3 cx

```
double DL_EllipseData::cx
```

X Coordinate of center point.

Referenced by DL\_Dxf::writeEllipse().

#### 4.22.3.4 cy

```
double DL_EllipseData::cy
```

Y Coordinate of center point.

Referenced by DL\_Dxf::writeEllipse().

#### 4.22.3.5 cz

```
double DL_EllipseData::cz
```

Z Coordinate of center point.

Referenced by DL\_Dxf::writeEllipse().

#### 4.22.3.6 mx

```
double DL_EllipseData::mx
```

X coordinate of the endpoint of the major axis.

Referenced by DL\_Dxf::writeEllipse().

#### 4.22.3.7 my

```
double DL_EllipseData::my
```

Y coordinate of the endpoint of the major axis.

Referenced by DL\_Dxf::writeEllipse().

#### 4.22.3.8 mz

```
double DL_EllipseData::mz
```

Z coordinate of the endpoint of the major axis.

Referenced by DL\_Dxf::writeEllipse().

#### 4.22.3.9 ratio

```
double DL_EllipseData::ratio
```

Ratio of minor axis to major axis..

Referenced by DL\_Dxf::writeEllipse().

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

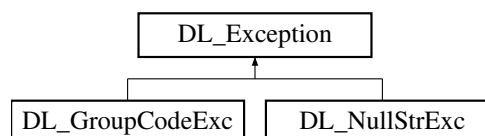
- src/dl\_entities.h

## 4.23 DL\_Exception Class Reference

Used for exception handling.

```
#include <dl_exception.h>
```

Inheritance diagram for DL\_Exception:



### 4.23.1 Detailed Description

Used for exception handling.

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

- `src/dl_exception.h`

## 4.24 DL\_Extrusion Class Reference

Extrusion direction.

```
#include <dl_extrusion.h>
```

### Public Member Functions

- [DL\\_Extrusion](#) ()  
*Default constructor.*
- [~DL\\_Extrusion](#) ()  
*Destructor.*
- [DL\\_Extrusion](#) (double dx, double dy, double dz, double elevation)  
*Constructor for DXF extrusion.*
- void [setDirection](#) (double dx, double dy, double dz)  
*Sets the direction vector.*
- double \* [getDirection](#) () const
- void [getDirection](#) (double dir[]) const
- void [setElevation](#) (double elevation)  
*Sets the elevation.*
- double [getElevation](#) () const
- [DL\\_Extrusion operator=](#) (const [DL\\_Extrusion](#) &extru)  
*Copies extrusion (deep copies) from another extrusion object.*

### 4.24.1 Detailed Description

Extrusion direction.

Author

Andrew Mustun

### 4.24.2 Constructor & Destructor Documentation

#### 4.24.2.1 DL\_Extrusion()

```
DL_Extrusion::DL_Extrusion (
    double dx,
    double dy,
    double dz,
    double elevation ) [inline]
```

Constructor for DXF extrusion.

## Parameters

<i>direction</i>	Vector of axis along which the entity shall be extruded this is also the Z axis of the Entity coordinate system
<i>elevation</i>	Distance of the entities XY plane from the origin of the world coordinate system

### 4.24.3 Member Function Documentation

#### 4.24.3.1 `getDirection()` [1/2]

```
double* DL_Extrusion::getDirection ( ) const [inline]
```

## Returns

direction vector.

#### 4.24.3.2 `getDirection()` [2/2]

```
void DL_Extrusion::getDirection (
    double dir[ ] ) const [inline]
```

## Returns

direction vector.

#### 4.24.3.3 `getElevation()`

```
double DL_Extrusion::getElevation ( ) const [inline]
```

## Returns

Elevation.

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

- `src/dl_extrusion.h`

## 4.25 DL\_FitPointData Struct Reference

Spline fit point data.

```
#include <dl_entities.h>
```

### Public Member Functions

- [DL\\_FitPointData](#) (double [x](#), double [y](#), double [z](#))  
*Constructor.*

### Public Attributes

- double [x](#)
- double [y](#)
- double [z](#)

#### 4.25.1 Detailed Description

Spline fit point data.

#### 4.25.2 Constructor & Destructor Documentation

##### 4.25.2.1 DL\_FitPointData()

```
DL_FitPointData::DL_FitPointData (
    double x,
    double y,
    double z ) [inline]
```

Constructor.

Parameters: see member variables.

#### 4.25.3 Member Data Documentation

##### 4.25.3.1 [x](#)

```
double DL_FitPointData::x
```

X coordinate of the fit point.

Referenced by [DL\\_Dxf::writeFitPoint\(\)](#).

#### 4.25.3.2 y

```
double DL_FitPointData::y
```

Y coordinate of the fit point.

Referenced by DL\_Dxf::writeFitPoint().

#### 4.25.3.3 z

```
double DL_FitPointData::z
```

Z coordinate of the fit point.

Referenced by DL\_Dxf::writeFitPoint().

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

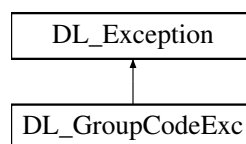
- src/dl\_entities.h

## 4.26 DL\_GroupCodeExc Class Reference

Used for exception handling.

```
#include <dl_exception.h>
```

Inheritance diagram for DL\_GroupCodeExc:



### 4.26.1 Detailed Description

Used for exception handling.

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

- src/dl\_exception.h

## 4.27 DL\_HatchData Struct Reference

Hatch data.

```
#include <dl_entities.h>
```

## Public Member Functions

- [DL\\_HatchData](#) ()  
*Default constructor.*
- [DL\\_HatchData](#) (int [numLoops](#), bool [solid](#), double [scale](#), double [angle](#), const std::string &[pattern](#), double [originX](#)=0.0, double [originY](#)=0.0)  
*Constructor.*

## Public Attributes

- int [numLoops](#)
- bool [solid](#)
- double [scale](#)
- double [angle](#)
- std::string [pattern](#)
- double [originX](#)
- double [originY](#)

### 4.27.1 Detailed Description

Hatch data.

### 4.27.2 Constructor & Destructor Documentation

#### 4.27.2.1 DL\_HatchData()

```
DL_HatchData::DL_HatchData (  
    int numLoops,  
    bool solid,  
    double scale,  
    double angle,  
    const std::string & pattern,  
    double originX = 0.0,  
    double originY = 0.0 ) [inline]
```

Constructor.

Parameters: see member variables.

### 4.27.3 Member Data Documentation

#### 4.27.3.1 angle

```
double DL_HatchData::angle
```

Pattern angle in degrees

Referenced by DL\_Dxf::writeHatch2().

#### 4.27.3.2 numLoops

```
int DL_HatchData::numLoops
```

Number of boundary paths (loops).

Referenced by DL\_Dxf::writeHatch1().

#### 4.27.3.3 originX

```
double DL_HatchData::originX
```

Pattern origin

Referenced by DL\_Dxf::writeHatch2().

#### 4.27.3.4 pattern

```
std::string DL_HatchData::pattern
```

Pattern name.

Referenced by DL\_Dxf::writeHatch1().

#### 4.27.3.5 scale

```
double DL_HatchData::scale
```

Pattern scale or spacing

Referenced by DL\_Dxf::writeHatch2().



## 4.27.3.6 solid

```
bool DL_HatchData::solid
```

Solid fill flag (true=solid, false=pattern).

Referenced by DL\_Dxf::writeHatch1(), and DL\_Dxf::writeHatch2().

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

- src/dl\_entities.h

## 4.28 DL\_HatchEdgeData Struct Reference

Hatch edge data.

```
#include <dl_entities.h>
```

## Public Member Functions

- [DL\\_HatchEdgeData](#) ()  
*Default constructor.*
- [DL\\_HatchEdgeData](#) (double [x1](#), double [y1](#), double [x2](#), double [y2](#))  
*Constructor for a line edge.*
- [DL\\_HatchEdgeData](#) (double [cx](#), double [cy](#), double [radius](#), double [angle1](#), double [angle2](#), bool [ccw](#))  
*Constructor for an arc edge.*
- [DL\\_HatchEdgeData](#) (double [cx](#), double [cy](#), double [mx](#), double [my](#), double [ratio](#), double [angle1](#), double [angle2](#), bool [ccw](#))  
*Constructor for an ellipse arc edge.*
- [DL\\_HatchEdgeData](#) (unsigned int [degree](#), bool rational, bool periodic, unsigned int [nKnots](#), unsigned int [nControl](#), unsigned int [nFit](#), const std::vector< double > &knots, const std::vector< std::vector< double > > &controlPoints, const std::vector< std::vector< double > > &fitPoints, const std::vector< double > &weights, double startTangentX, double startTangentY, double endTangentX, double endTangentY)  
*Constructor for a spline edge.*

## Public Attributes

- bool [defined](#)  
*Set to true if this edge is fully defined.*
- int [type](#)  
*Edge type.*
- double [x1](#)
- double [y1](#)
- double [x2](#)
- double [y2](#)
- double [cx](#)
- double [cy](#)
- double [radius](#)
- double [angle1](#)

- double [angle2](#)
- bool [ccw](#)
- double [mx](#)
- double [my](#)
- double [ratio](#)
- unsigned int [degree](#)
- bool **rational**
- bool **periodic**
- unsigned int [nKnots](#)
- unsigned int [nControl](#)
- unsigned int [nFit](#)
- std::vector< std::vector< double > > **controlPoints**
- std::vector< double > **knots**
- std::vector< double > **weights**
- std::vector< std::vector< double > > **fitPoints**
- double **startTangentX**
- double **startTangentY**
- double **endTangentX**
- double **endTangentY**
- std::vector< std::vector< double > > [vertices](#)

*Polyline boundary vertices (x y [bulge])*

### 4.28.1 Detailed Description

Hatch edge data.

### 4.28.2 Constructor & Destructor Documentation

#### 4.28.2.1 DL\_HatchEdgeData() [1/4]

```
DL_HatchEdgeData::DL_HatchEdgeData (
    double x1,
    double y1,
    double x2,
    double y2 ) [inline]
```

Constructor for a line edge.

Parameters: see member variables.

#### 4.28.2.2 DL\_HatchEdgeData() [2/4]

```
DL_HatchEdgeData::DL_HatchEdgeData (
    double cx,
    double cy,
    double radius,
    double angle1,
    double angle2,
    bool ccw ) [inline]
```

Constructor for an arc edge.

Parameters: see member variables.

**4.28.2.3 DL\_HatchEdgeData() [3/4]**

```
DL_HatchEdgeData::DL_HatchEdgeData (
    double cx,
    double cy,
    double mx,
    double my,
    double ratio,
    double angle1,
    double angle2,
    bool ccw ) [inline]
```

Constructor for an ellipse arc edge.

Parameters: see member variables.

**4.28.2.4 DL\_HatchEdgeData() [4/4]**

```
DL_HatchEdgeData::DL_HatchEdgeData (
    unsigned int degree,
    bool rational,
    bool periodic,
    unsigned int nKnots,
    unsigned int nControl,
    unsigned int nFit,
    const std::vector< double > & knots,
    const std::vector< std::vector< double > > & controlPoints,
    const std::vector< std::vector< double > > & fitPoints,
    const std::vector< double > & weights,
    double startTangentX,
    double startTangentY,
    double endTangentX,
    double endTangentY ) [inline]
```

Constructor for a spline edge.

Parameters: see member variables.

**4.28.3 Member Data Documentation****4.28.3.1 angle1**

```
double DL_HatchEdgeData::angle1
```

Start angle of arc or ellipse arc.

Referenced by DL\_Dxf::handleHatchData(), and DL\_Dxf::writeHatchEdge().

#### 4.28.3.2 angle2

```
double DL_HatchEdgeData::angle2
```

End angle of arc or ellipse arc.

Referenced by DL\_Dxf::handleHatchData(), and DL\_Dxf::writeHatchEdge().

#### 4.28.3.3 ccw

```
bool DL_HatchEdgeData::ccw
```

Counterclockwise flag for arc or ellipse arc.

Referenced by DL\_Dxf::handleHatchData(), and DL\_Dxf::writeHatchEdge().

#### 4.28.3.4 cx

```
double DL_HatchEdgeData::cx
```

Center point of arc or ellipse arc (X).

Referenced by DL\_Dxf::handleHatchData(), and DL\_Dxf::writeHatchEdge().

#### 4.28.3.5 cy

```
double DL_HatchEdgeData::cy
```

Center point of arc or ellipse arc (Y).

Referenced by DL\_Dxf::handleHatchData(), and DL\_Dxf::writeHatchEdge().

#### 4.28.3.6 degree

```
unsigned int DL_HatchEdgeData::degree
```

Spline degree

Referenced by DL\_Dxf::handleHatchData(), and DL\_Dxf::writeHatchEdge().

#### 4.28.3.7 mx

```
double DL_HatchEdgeData::mx
```

Major axis end point (X).

Referenced by DL\_Dxf::handleHatchData(), and DL\_Dxf::writeHatchEdge().

#### 4.28.3.8 my

```
double DL_HatchEdgeData::my
```

Major axis end point (Y).

Referenced by DL\_Dxf::handleHatchData(), and DL\_Dxf::writeHatchEdge().

#### 4.28.3.9 nControl

```
unsigned int DL_HatchEdgeData::nControl
```

Number of control points.

Referenced by DL\_Dxf::handleHatchData(), and DL\_Dxf::writeHatchEdge().

#### 4.28.3.10 nFit

```
unsigned int DL_HatchEdgeData::nFit
```

Number of fit points.

Referenced by DL\_Dxf::handleHatchData(), and DL\_Dxf::writeHatchEdge().

#### 4.28.3.11 nKnots

```
unsigned int DL_HatchEdgeData::nKnots
```

Number of knots.

Referenced by DL\_Dxf::handleHatchData(), and DL\_Dxf::writeHatchEdge().

#### 4.28.3.12 radius

```
double DL_HatchEdgeData::radius
```

Arc radius.

Referenced by DL\_Dxf::handleHatchData(), and DL\_Dxf::writeHatchEdge().

#### 4.28.3.13 ratio

```
double DL_HatchEdgeData::ratio
```

Axis ratio

Referenced by DL\_Dxf::handleHatchData(), and DL\_Dxf::writeHatchEdge().

#### 4.28.3.14 type

```
int DL_HatchEdgeData::type
```

Edge type.

1=line, 2=arc, 3=elliptic arc, 4=spline.

Referenced by DL\_Dxf::handleHatchData(), and DL\_Dxf::writeHatchEdge().

#### 4.28.3.15 x1

```
double DL_HatchEdgeData::x1
```

Start point (X).

Referenced by DL\_Dxf::handleHatchData(), and DL\_Dxf::writeHatchEdge().

#### 4.28.3.16 x2

```
double DL_HatchEdgeData::x2
```

End point (X).

Referenced by DL\_Dxf::handleHatchData(), and DL\_Dxf::writeHatchEdge().

#### 4.28.3.17 y1

```
double DL_HatchEdgeData::y1
```

Start point (Y).

Referenced by DL\_Dxf::handleHatchData(), and DL\_Dxf::writeHatchEdge().

#### 4.28.3.18 y2

```
double DL_HatchEdgeData::y2
```

End point (Y).

Referenced by DL\_Dxf::handleHatchData(), and DL\_Dxf::writeHatchEdge().

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

- src/dl\_entities.h

## 4.29 DL\_HatchLoopData Struct Reference

Hatch boundary path (loop) data.

```
#include <dl_entities.h>
```

### Public Member Functions

- [DL\\_HatchLoopData](#) ()  
*Default constructor.*
- [DL\\_HatchLoopData](#) (int hNumEdges)  
*Constructor.*

### Public Attributes

- int [numEdges](#)

#### 4.29.1 Detailed Description

Hatch boundary path (loop) data.

#### 4.29.2 Constructor & Destructor Documentation

#### 4.29.2.1 DL\_HatchLoopData()

```
DL_HatchLoopData::DL_HatchLoopData (
    int hNumEdges ) [inline]
```

Constructor.

Parameters: see member variables.

### 4.29.3 Member Data Documentation

#### 4.29.3.1 numEdges

```
int DL_HatchLoopData::numEdges
```

Number of edges in this loop.

Referenced by DL\_Dxf::writeHatchLoop1().

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

- src/dl\_entities.h

## 4.30 DL\_ImageData Struct Reference

Image Data.

```
#include <dl_entities.h>
```

### Public Member Functions

- [DL\\_ImageData](#) (const std::string &ref, double iipx, double iipy, double iipz, double iux, double iuy, double iuz, double ivx, double ivy, double ivz, int iwidth, int iheight, int ibrightness, int icontrast, int ifade)  
*Constructor.*

### Public Attributes

- std::string [ref](#)
- double [ipx](#)
- double [ipy](#)
- double [ipz](#)
- double [ux](#)
- double [uy](#)
- double [uz](#)
- double [vx](#)
- double [vy](#)
- double [vz](#)
- int [width](#)
- int [height](#)
- int [brightness](#)
- int [contrast](#)
- int [fade](#)



### 4.30.1 Detailed Description

Image Data.

### 4.30.2 Constructor & Destructor Documentation

#### 4.30.2.1 DL\_ImageData()

```
DL_ImageData::DL_ImageData (
    const std::string & iref,
    double iipx,
    double iipy,
    double iipz,
    double iux,
    double iuy,
    double iuz,
    double ivx,
    double ivy,
    double ivz,
    int iwidth,
    int iheight,
    int ibrightness,
    int icontrast,
    int ifade ) [inline]
```

Constructor.

Parameters: see member variables.

### 4.30.3 Member Data Documentation

#### 4.30.3.1 brightness

```
int DL_ImageData::brightness
```

Brightness (0..100, default = 50).

Referenced by DL\_Dxf::writeImage().

#### 4.30.3.2 contrast

```
int DL_ImageData::contrast
```

Contrast (0..100, default = 50).

Referenced by DL\_Dxf::writeImage().

#### 4.30.3.3 fade

```
int DL_ImageData::fade
```

Fade (0..100, default = 0).

Referenced by DL\_Dxf::writeImage().

#### 4.30.3.4 height

```
int DL_ImageData::height
```

Height of image in pixel.

Referenced by DL\_Dxf::writeImage(), and DL\_Dxf::writeImageDef().

#### 4.30.3.5 ipx

```
double DL_ImageData::ipx
```

X Coordinate of insertion point.

Referenced by DL\_Dxf::writeImage().

#### 4.30.3.6 ipy

```
double DL_ImageData::ipy
```

Y Coordinate of insertion point.

Referenced by DL\_Dxf::writeImage().

#### 4.30.3.7 ipz

```
double DL_ImageData::ipz
```

Z Coordinate of insertion point.

Referenced by DL\_Dxf::writeImage().

#### 4.30.3.8 ref

```
std::string DL_ImageData::ref
```

Reference to the image file (unique, used to refer to the image def object).

Referenced by DL\_Dxf::writeImageDef().

#### 4.30.3.9 ux

```
double DL_ImageData::ux
```

X Coordinate of u vector along bottom of image.

Referenced by DL\_Dxf::writeImage().

#### 4.30.3.10 uy

```
double DL_ImageData::uy
```

Y Coordinate of u vector along bottom of image.

Referenced by DL\_Dxf::writeImage().

#### 4.30.3.11 uz

```
double DL_ImageData::uz
```

Z Coordinate of u vector along bottom of image.

Referenced by DL\_Dxf::writeImage().

#### 4.30.3.12 vx

```
double DL_ImageData::vx
```

X Coordinate of v vector along left side of image.

Referenced by DL\_Dxf::writeImage().

#### 4.30.3.13 vy

```
double DL_ImageData::vy
```

Y Coordinate of v vector along left side of image.

Referenced by DL\_Dxf::writeImage().

#### 4.30.3.14 vz

```
double DL_ImageData::vz
```

Z Coordinate of v vector along left side of image.

Referenced by DL\_Dxf::writeImage().

#### 4.30.3.15 width

```
int DL_ImageData::width
```

Width of image in pixel.

Referenced by DL\_Dxf::writeImage(), and DL\_Dxf::writeImageDef().

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

- src/dl\_entities.h

## 4.31 DL\_ImageDefData Struct Reference

Image Definition Data.

```
#include <dl_entities.h>
```

## Public Member Functions

- [DL\\_ImageDefData](#) (const std::string &iref, const std::string &ifile)  
*Constructor.*

## Public Attributes

- std::string [ref](#)
- std::string [file](#)

### 4.31.1 Detailed Description

Image Definition Data.

### 4.31.2 Constructor & Destructor Documentation

#### 4.31.2.1 DL\_ImageDefData()

```
DL_ImageDefData::DL_ImageDefData (
    const std::string & iref,
    const std::string & ifile ) [inline]
```

Constructor.

Parameters: see member variables.

### 4.31.3 Member Data Documentation

#### 4.31.3.1 file

```
std::string DL_ImageDefData::file
```

Image file

#### 4.31.3.2 ref

```
std::string DL_ImageDefData::ref
```

Reference to the image file (unique, used to refer to the image def object).

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

- src/dl\_entities.h

## 4.32 DL\_InsertData Struct Reference

Insert Data.

```
#include <dl_entities.h>
```

### Public Member Functions

- [DL\\_InsertData](#) (const std::string &[name](#), double [ipx](#), double [ipy](#), double [ipz](#), double [sx](#), double [sy](#), double [sz](#), double [angle](#), int [cols](#), int [rows](#), double [colSp](#), double [rowSp](#))  
*Constructor.*

### Public Attributes

- std::string [name](#)
- double [ipx](#)
- double [ipy](#)
- double [ipz](#)
- double [sx](#)
- double [sy](#)
- double [sz](#)
- double [angle](#)
- int [cols](#)
- int [rows](#)
- double [colSp](#)
- double [rowSp](#)

### 4.32.1 Detailed Description

Insert Data.

### 4.32.2 Constructor & Destructor Documentation

#### 4.32.2.1 DL\_InsertData()

```
DL_InsertData::DL_InsertData (  
    const std::string & name,  
    double ipx,  
    double ipy,  
    double ipz,  
    double sx,  
    double sy,  
    double sz,  
    double angle,  
    int cols,  
    int rows,  
    double colSp,  
    double rowSp ) [inline]
```

Constructor.

Parameters: see member variables.

### 4.32.3 Member Data Documentation

#### 4.32.3.1 angle

```
double DL_InsertData::angle
```

Rotation angle in degrees.

Referenced by DL\_Dxf::writeInsert().

#### 4.32.3.2 cols

```
int DL_InsertData::cols
```

Number of columns if we insert an array of the block or 1.

Referenced by DL\_Dxf::writeInsert().

#### 4.32.3.3 colSp

```
double DL_InsertData::colSp
```

Values for the spacing between cols.

Referenced by DL\_Dxf::writeInsert().

#### 4.32.3.4 ipx

```
double DL_InsertData::ipx
```

X Coordinate of insertion point.

Referenced by DL\_Dxf::writeInsert().

#### 4.32.3.5 ipy

```
double DL_InsertData::ipy
```

Y Coordinate of insertion point.

Referenced by DL\_Dxf::writeInsert().

#### 4.32.3.6 ipz

```
double DL_InsertData::ipz
```

Z Coordinate of insertion point.

Referenced by DL\_Dxf::writeInsert().

#### 4.32.3.7 name

```
std::string DL_InsertData::name
```

Name of the referred block.

Referenced by DL\_Dxf::writeInsert().

#### 4.32.3.8 rows

```
int DL_InsertData::rows
```

Number of rows if we insert an array of the block or 1.

Referenced by DL\_Dxf::writeInsert().

#### 4.32.3.9 rowSp

```
double DL_InsertData::rowSp
```

Values for the spacing between rows.

Referenced by DL\_Dxf::writeInsert().

#### 4.32.3.10 sx

```
double DL_InsertData::sx
```

X Scale factor.

Referenced by DL\_Dxf::writeInsert().



#### 4.32.3.11 sy

```
double DL_InsertData::sy
```

Y Scale factor.

Referenced by DL\_Dxf::writeInsert().

#### 4.32.3.12 sz

```
double DL_InsertData::sz
```

Z Scale factor.

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

- src/dl\_entities.h

## 4.33 DL\_KnotData Struct Reference

Spline knot data.

```
#include <dl_entities.h>
```

### Public Member Functions

- [DL\\_KnotData](#) (double pk)  
*Constructor.*

### Public Attributes

- double [k](#)

#### 4.33.1 Detailed Description

Spline knot data.

#### 4.33.2 Constructor & Destructor Documentation

#### 4.33.2.1 DL\_KnotData()

```
DL_KnotData::DL_KnotData (
    double pk ) [inline]
```

Constructor.

Parameters: see member variables.

### 4.33.3 Member Data Documentation

#### 4.33.3.1 k

```
double DL_KnotData::k
```

Knot value.

Referenced by DL\_Dxf::writeKnot().

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

- src/dl\_entities.h

## 4.34 DL\_LayerData Struct Reference

Layer Data.

```
#include <dl_entities.h>
```

### Public Member Functions

- [DL\\_LayerData](#) (const std::string &[name](#), int [flags](#), bool [off](#)=false)  
*Constructor.*

### Public Attributes

- std::string [name](#)  
*Layer name.*
- int [flags](#)  
*Layer flags.*
- bool [off](#)  
*Layer is off.*

### 4.34.1 Detailed Description

Layer Data.

### 4.34.2 Constructor & Destructor Documentation

#### 4.34.2.1 DL\_LayerData()

```
DL_LayerData::DL_LayerData (
    const std::string & name,
    int flags,
    bool off = false ) [inline]
```

Constructor.

Parameters: see member variables.

### 4.34.3 Member Data Documentation

#### 4.34.3.1 flags

```
int DL_LayerData::flags
```

Layer flags.

(1 = frozen, 2 = frozen by default, 4 = locked)

Referenced by DL\_Dxf::writeLayer().

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

- src/dl\_entities.h

## 4.35 DL\_LeaderData Struct Reference

Leader (arrow).

```
#include <dl_entities.h>
```

## Public Member Functions

- [DL\\_LeaderData](#) (int [arrowHeadFlag](#), int [leaderPathType](#), int [leaderCreationFlag](#), int [hooklineDirectionFlag](#), int [hooklineFlag](#), double [textAnnotationHeight](#), double [textAnnotationWidth](#), int [number](#), double [dimScale](#)=1.0)

*Constructor.*

## Public Attributes

- int [arrowHeadFlag](#)
- int [leaderPathType](#)
- int [leaderCreationFlag](#)
- int [hooklineDirectionFlag](#)
- int [hooklineFlag](#)
- double [textAnnotationHeight](#)
- double [textAnnotationWidth](#)
- int [number](#)
- double [dimScale](#)

### 4.35.1 Detailed Description

Leader (arrow).

### 4.35.2 Constructor & Destructor Documentation

#### 4.35.2.1 DL\_LeaderData()

```
DL_LeaderData::DL_LeaderData (
    int arrowHeadFlag,
    int leaderPathType,
    int leaderCreationFlag,
    int hooklineDirectionFlag,
    int hooklineFlag,
    double textAnnotationHeight,
    double textAnnotationWidth,
    int number,
    double dimScale = 1.0 ) [inline]
```

Constructor.

Parameters: see member variables.

### 4.35.3 Member Data Documentation

#### 4.35.3.1 arrowHeadFlag

```
int DL_LeaderData::arrowHeadFlag
```

Arrow head flag (71).

Referenced by DL\_Dxf::writeLeader().

#### 4.35.3.2 dimScale

```
double DL_LeaderData::dimScale
```

Dimension scale (dimscale) style override.

#### 4.35.3.3 hooklineDirectionFlag

```
int DL_LeaderData::hooklineDirectionFlag
```

Hookline direction flag (74).

Referenced by DL\_Dxf::writeLeader().

#### 4.35.3.4 hooklineFlag

```
int DL_LeaderData::hooklineFlag
```

Hookline flag (75)

Referenced by DL\_Dxf::writeLeader().

#### 4.35.3.5 leaderCreationFlag

```
int DL_LeaderData::leaderCreationFlag
```

Leader creation flag (73).

Referenced by DL\_Dxf::writeLeader().

#### 4.35.3.6 leaderPathType

```
int DL_LeaderData::leaderPathType
```

Leader path type (72).

Referenced by DL\_Dxf::writeLeader().

#### 4.35.3.7 number

```
int DL_LeaderData::number
```

Number of vertices in leader (76).

Referenced by DL\_Dxf::writeLeader().

#### 4.35.3.8 textAnnotationHeight

```
double DL_LeaderData::textAnnotationHeight
```

Text annotation height (40).

Referenced by DL\_Dxf::writeLeader().

#### 4.35.3.9 textAnnotationWidth

```
double DL_LeaderData::textAnnotationWidth
```

Text annotation width (41)

Referenced by DL\_Dxf::writeLeader().

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

- src/dl\_entities.h

## 4.36 DL\_LeaderVertexData Struct Reference

Leader Vertex Data.

```
#include <dl_entities.h>
```

## Public Member Functions

- [DL\\_LeaderVertexData](#) (double px=0.0, double py=0.0, double pz=0.0)  
*Constructor.*

## Public Attributes

- double [x](#)
- double [y](#)
- double [z](#)

### 4.36.1 Detailed Description

Leader Vertex Data.

### 4.36.2 Constructor & Destructor Documentation

#### 4.36.2.1 DL\_LeaderVertexData()

```
DL_LeaderVertexData::DL_LeaderVertexData (
    double px = 0.0,
    double py = 0.0,
    double pz = 0.0 ) [inline]
```

Constructor.

Parameters: see member variables.

### 4.36.3 Member Data Documentation

#### 4.36.3.1 x

```
double DL_LeaderVertexData::x
```

X Coordinate of the vertex.

Referenced by `DL_Dxf::writeLeaderVertex()`.

#### 4.36.3.2 y

```
double DL_LeaderVertexData::y
```

Y Coordinate of the vertex.

Referenced by DL\_Dxf::writeLeaderVertex().

#### 4.36.3.3 z

```
double DL_LeaderVertexData::z
```

Z Coordinate of the vertex.

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

- src/dl\_entities.h

## 4.37 DL\_LineData Struct Reference

Line Data.

```
#include <dl_entities.h>
```

### Public Member Functions

- [DL\\_LineData](#) (double lx1, double ly1, double lz1, double lx2, double ly2, double lz2)  
*Constructor.*

### Public Attributes

- double [x1](#)
- double [y1](#)
- double [z1](#)
- double [x2](#)
- double [y2](#)
- double [z2](#)

#### 4.37.1 Detailed Description

Line Data.

#### 4.37.2 Constructor & Destructor Documentation



### 4.37.2.1 DL\_LineData()

```
DL_LineData::DL_LineData (
    double lx1,
    double ly1,
    double lz1,
    double lx2,
    double ly2,
    double lz2 ) [inline]
```

Constructor.

Parameters: see member variables.

## 4.37.3 Member Data Documentation

### 4.37.3.1 x1

```
double DL_LineData::x1
```

X Start coordinate of the point.

Referenced by DL\_Dxf::writeLine().

### 4.37.3.2 x2

```
double DL_LineData::x2
```

X End coordinate of the point.

Referenced by DL\_Dxf::writeLine().

### 4.37.3.3 y1

```
double DL_LineData::y1
```

Y Start coordinate of the point.

Referenced by DL\_Dxf::writeLine().

#### 4.37.3.4 y2

```
double DL_LineData::y2
```

Y End coordinate of the point.

Referenced by DL\_Dxf::writeLine().

#### 4.37.3.5 z1

```
double DL_LineData::z1
```

Z Start coordinate of the point.

Referenced by DL\_Dxf::writeLine().

#### 4.37.3.6 z2

```
double DL_LineData::z2
```

Z End coordinate of the point.

Referenced by DL\_Dxf::writeLine().

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

- src/dl\_entities.h

## 4.38 DL\_LinetypeData Struct Reference

Line Type Data.

```
#include <dl_entities.h>
```

### Public Member Functions

- [DL\\_LinetypeData](#) (const std::string &[name](#), const std::string &[description](#), int [flags](#), int [numberOfDashes](#), double [patternLength](#), double \*[pattern](#)=NULL)

*Constructor.*

## Public Attributes

- std::string [name](#)  
*Linetype name.*
- std::string [description](#)  
*Linetype description.*
- int [flags](#)  
*Linetype flags.*
- int [numberOfDashes](#)  
*Number of dashes.*
- double [patternLength](#)  
*Pattern length.*
- double \* [pattern](#)  
*Pattern.*

### 4.38.1 Detailed Description

Line Type Data.

### 4.38.2 Constructor & Destructor Documentation

#### 4.38.2.1 DL\_LinetypeData()

```
DL_LinetypeData::DL_LinetypeData (  
    const std::string & name,  
    const std::string & description,  
    int flags,  
    int numberOfDashes,  
    double patternLength,  
    double * pattern = NULL ) [inline]
```

Constructor.

Parameters: see member variables.

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

- src/dl\_entities.h

## 4.39 DL\_MTextData Struct Reference

MText Data.

```
#include <dl_entities.h>
```

## Public Member Functions

- [DL\\_MTextData](#) (double [ipx](#), double [ipy](#), double [ipz](#), double [dirx](#), double [diry](#), double [dirz](#), double [height](#), double [width](#), int [attachmentPoint](#), int [drawingDirection](#), int [lineSpacingStyle](#), double [lineSpacingFactor](#), const std::string &[text](#), const std::string &[style](#), double [angle](#))

*Constructor.*

## Public Attributes

- double [ipx](#)
- double [ipy](#)
- double [ipz](#)
- double [dirx](#)
- double [diry](#)
- double [dirz](#)
- double [height](#)
- double [width](#)
- int [attachmentPoint](#)  
*Attachment point.*
- int [drawingDirection](#)  
*Drawing direction.*
- int [lineSpacingStyle](#)  
*Line spacing style.*
- double [lineSpacingFactor](#)  
*Line spacing factor.*
- std::string [text](#)
- std::string [style](#)
- double [angle](#)

### 4.39.1 Detailed Description

MText Data.

### 4.39.2 Constructor & Destructor Documentation

#### 4.39.2.1 DL\_MTextData()

```
DL_MTextData::DL_MTextData (
    double ipx,
    double ipy,
    double ipz,
    double dirx,
    double diry,
    double dirz,
    double height,
    double width,
    int attachmentPoint,
```

```
int drawingDirection,  
int lineSpacingStyle,  
double lineSpacingFactor,  
const std::string & text,  
const std::string & style,  
double angle ) [inline]
```

Constructor.

Parameters: see member variables.

### 4.39.3 Member Data Documentation

#### 4.39.3.1 angle

```
double DL_MTextData::angle
```

Rotation angle.

Referenced by DL\_Dxf::writeMText().

#### 4.39.3.2 attachmentPoint

```
int DL_MTextData::attachmentPoint
```

Attachment point.

1 = Top left, 2 = Top center, 3 = Top right, 4 = Middle left, 5 = Middle center, 6 = Middle right, 7 = Bottom left, 8 = Bottom center, 9 = Bottom right

Referenced by DL\_Dxf::writeMText().

#### 4.39.3.3 dirx

```
double DL_MTextData::dirx
```

X Coordinate of X direction vector.

#### 4.39.3.4 diry

```
double DL_MTextData::diry
```

Y Coordinate of X direction vector.

#### 4.39.3.5 dirz

```
double DL_MTextData::dirz
```

Z Coordinate of X direction vector.

#### 4.39.3.6 drawingDirection

```
int DL_MTextData::drawingDirection
```

Drawing direction.

1 = left to right, 3 = top to bottom, 5 = by style

Referenced by DL\_Dxf::writeMText().

#### 4.39.3.7 height

```
double DL_MTextData::height
```

Text height

Referenced by DL\_Dxf::writeMText().

#### 4.39.3.8 ipx

```
double DL_MTextData::ipx
```

X Coordinate of insertion point.

Referenced by DL\_Dxf::writeMText().

#### 4.39.3.9 ipy

```
double DL_MTextData::ipy
```

Y Coordinate of insertion point.

Referenced by DL\_Dxf::writeMText().

#### 4.39.3.10 ipz

```
double DL_MTextData::ipz
```

Z Coordinate of insertion point.

Referenced by DL\_Dxf::writeMText().

#### 4.39.3.11 lineSpacingFactor

```
double DL_MTextData::lineSpacingFactor
```

Line spacing factor.

0.25 .. 4.0

Referenced by DL\_Dxf::writeMText().

#### 4.39.3.12 lineSpacingStyle

```
int DL_MTextData::lineSpacingStyle
```

Line spacing style.

1 = at least, 2 = exact

Referenced by DL\_Dxf::writeMText().

#### 4.39.3.13 style

```
std::string DL_MTextData::style
```

Style string.

Referenced by DL\_Dxf::writeMText().

#### 4.39.3.14 text

```
std::string DL_MTextData::text
```

Text string.

Referenced by DL\_Dxf::writeMText().

#### 4.39.3.15 width

```
double DL_MTextData::width
```

Width of the text box.

Referenced by DL\_Dxf::writeMText().

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

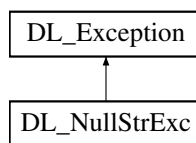
- src/dl\_entities.h

### 4.40 DL\_NullStrExc Class Reference

Used for exception handling.

```
#include <dl_exception.h>
```

Inheritance diagram for DL\_NullStrExc:



#### 4.40.1 Detailed Description

Used for exception handling.

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

- src/dl\_exception.h

### 4.41 DL\_PointData Struct Reference

Point Data.

```
#include <dl_entities.h>
```

#### Public Member Functions

- [DL\\_PointData](#) (double px=0.0, double py=0.0, double pz=0.0)  
*Constructor.*



## Public Attributes

- double [x](#)
- double [y](#)
- double [z](#)

### 4.41.1 Detailed Description

Point Data.

### 4.41.2 Constructor & Destructor Documentation

#### 4.41.2.1 DL\_PointData()

```
DL_PointData::DL_PointData (  
    double px = 0.0,  
    double py = 0.0,  
    double pz = 0.0 ) [inline]
```

Constructor.

Parameters: see member variables.

### 4.41.3 Member Data Documentation

#### 4.41.3.1 [x](#)

```
double DL_PointData::x
```

X Coordinate of the point.

Referenced by `DL_Dxf::writePoint()`.

#### 4.41.3.2 [y](#)

```
double DL_PointData::y
```

Y Coordinate of the point.

Referenced by `DL_Dxf::writePoint()`.

#### 4.41.3.3 z

```
double DL_PointData::z
```

Z Coordinate of the point.

Referenced by DL\_Dxf::writePoint().

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

- src/dl\_entities.h

## 4.42 DL\_PolylineData Struct Reference

Polyline Data.

```
#include <dl_entities.h>
```

### Public Member Functions

- [DL\\_PolylineData](#) (int pNumber, int pMVerteces, int pNVerteces, int pFlags, double pElevation=0.0)  
*Constructor.*

### Public Attributes

- unsigned int [number](#)
- unsigned int [m](#)
- unsigned int [n](#)
- double [elevation](#)
- int [flags](#)

#### 4.42.1 Detailed Description

Polyline Data.

#### 4.42.2 Constructor & Destructor Documentation

##### 4.42.2.1 DL\_PolylineData()

```
DL_PolylineData::DL_PolylineData (  
    int pNumber,  
    int pMVerteces,  
    int pNVerteces,  
    int pFlags,  
    double pElevation = 0.0 ) [inline]
```

Constructor.

Parameters: see member variables.

### 4.42.3 Member Data Documentation

#### 4.42.3.1 elevation

```
double DL_PolylineData::elevation
```

elevation of the polyline.

#### 4.42.3.2 flags

```
int DL_PolylineData::flags
```

Flags

Referenced by DL\_Dxf::writePolyline().

#### 4.42.3.3 m

```
unsigned int DL_PolylineData::m
```

Number of vertices in m direction if polyline is a polygon mesh.

#### 4.42.3.4 n

```
unsigned int DL_PolylineData::n
```

Number of vertices in n direction if polyline is a polygon mesh.

#### 4.42.3.5 number

```
unsigned int DL_PolylineData::number
```

Number of vertices in this polyline.

Referenced by DL\_Dxf::writePolyline().

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

- src/dl\_entities.h

## 4.43 DL\_RayData Struct Reference

Ray Data.

```
#include <dl_entities.h>
```

### Public Member Functions

- [DL\\_RayData](#) (double [bx](#), double [by](#), double [bz](#), double [dx](#), double [dy](#), double [dz](#))  
*Constructor.*

### Public Attributes

- double [bx](#)
- double [by](#)
- double [bz](#)
- double [dx](#)
- double [dy](#)
- double [dz](#)

### 4.43.1 Detailed Description

Ray Data.

### 4.43.2 Constructor & Destructor Documentation

#### 4.43.2.1 DL\_RayData()

```
DL_RayData::DL_RayData (  
    double bx,  
    double by,  
    double bz,  
    double dx,  
    double dy,  
    double dz ) [inline]
```

Constructor.

Parameters: see member variables.

### 4.43.3 Member Data Documentation

#### 4.43.3.1 bx

```
double DL_RayData::bx
```

X base point.

Referenced by DL\_Dxf::writeRay().

#### 4.43.3.2 by

```
double DL_RayData::by
```

Y base point.

Referenced by DL\_Dxf::writeRay().

#### 4.43.3.3 bz

```
double DL_RayData::bz
```

Z base point.

Referenced by DL\_Dxf::writeRay().

#### 4.43.3.4 dx

```
double DL_RayData::dx
```

X direction vector.

Referenced by DL\_Dxf::writeRay().

#### 4.43.3.5 dy

```
double DL_RayData::dy
```

Y direction vector.

Referenced by DL\_Dxf::writeRay().

#### 4.43.3.6 dz

```
double DL_RayData::dz
```

Z direction vector.

Referenced by DL\_Dxf::writeRay().

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

- src/dl\_entities.h

### 4.44 DL\_SplineData Struct Reference

Spline Data.

```
#include <dl_entities.h>
```

#### Public Member Functions

- [DL\\_SplineData](#) (int [degree](#), int [nKnots](#), int [nControl](#), int [nFit](#), int [flags](#))  
*Constructor.*

#### Public Attributes

- unsigned int [degree](#)
- unsigned int [nKnots](#)
- unsigned int [nControl](#)
- unsigned int [nFit](#)
- int [flags](#)
- double [tangentStartX](#)
- double [tangentStartY](#)
- double [tangentStartZ](#)
- double [tangentEndX](#)
- double [tangentEndY](#)
- double [tangentEndZ](#)

#### 4.44.1 Detailed Description

Spline Data.

#### 4.44.2 Constructor & Destructor Documentation

#### 4.44.2.1 DL\_SplineData()

```
DL_SplineData::DL_SplineData (
    int degree,
    int nKnots,
    int nControl,
    int nFit,
    int flags ) [inline]
```

Constructor.

Parameters: see member variables.

### 4.44.3 Member Data Documentation

#### 4.44.3.1 degree

```
unsigned int DL_SplineData::degree
```

Degree of the spline curve.

Referenced by DL\_Dxf::writeSpline().

#### 4.44.3.2 flags

```
int DL_SplineData::flags
```

Flags

Referenced by DL\_Dxf::writeSpline().

#### 4.44.3.3 nControl

```
unsigned int DL_SplineData::nControl
```

Number of control points.

Referenced by DL\_Dxf::writeSpline().

#### 4.44.3.4 nFit

```
unsigned int DL_SplineData::nFit
```

Number of fit points.

Referenced by DL\_Dxf::writeSpline().

#### 4.44.3.5 nKnots

```
unsigned int DL_SplineData::nKnots
```

Number of knots.

Referenced by DL\_Dxf::writeSpline().

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

- src/dl\_entities.h

## 4.45 DL\_StyleData Struct Reference

Text style data.

```
#include <dl_entities.h>
```

### Public Member Functions

- [DL\\_StyleData](#) (const std::string &name, int flags, double fixedTextHeight, double widthFactor, double obliqueAngle, int textGenerationFlags, double lastHeightUsed, const std::string &primaryFontFile, const std::string &bigFontFile)  
*Constructor Parameters: see member variables.*
- bool **operator==** (const [DL\\_StyleData](#) &other)

### Public Attributes

- std::string name  
*Style name.*
- int flags  
*Style flags.*
- double fixedTextHeight  
*Fixed text height or 0 for not fixed.*
- double widthFactor  
*Width factor.*
- double obliqueAngle  
*Oblique angle.*
- int textGenerationFlags  
*Text generation flags.*
- double lastHeightUsed  
*Last height used.*
- std::string primaryFontFile  
*Primary font file name.*
- std::string bigFontFile  
*Big font file name.*
- bool **bold**
- bool **italic**



### 4.45.1 Detailed Description

Text style data.

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

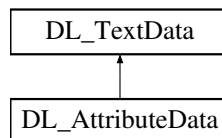
- src/dl\_entities.h

## 4.46 DL\_TextData Struct Reference

Text Data.

```
#include <dl_entities.h>
```

Inheritance diagram for DL\_TextData:



### Public Member Functions

- **DL\_TextData** (double [ipx](#), double [ipy](#), double [ipz](#), double [apx](#), double [apy](#), double [apz](#), double [height](#), double [xScaleFactor](#), int [textGenerationFlags](#), int [hJustification](#), int [vJustification](#), const std::string &[text](#), const std::string &[style](#), double [angle](#))

*Constructor.*

### Public Attributes

- double [ipx](#)
  - double [ipy](#)
  - double [ipz](#)
  - double [apx](#)
  - double [apy](#)
  - double [apz](#)
  - double [height](#)
  - double [xScaleFactor](#)
  - int [textGenerationFlags](#)
  - int [hJustification](#)
- Horizontal justification.*
- int [vJustification](#)
- Vertical justification.*
- std::string [text](#)
  - std::string [style](#)
  - double [angle](#)

### 4.46.1 Detailed Description

Text Data.

### 4.46.2 Constructor & Destructor Documentation

#### 4.46.2.1 DL\_TextData()

```
DL_TextData::DL_TextData (
    double ipx,
    double ipy,
    double ipz,
    double apx,
    double apy,
    double apz,
    double height,
    double xScaleFactor,
    int textGenerationFlags,
    int hJustification,
    int vJustification,
    const std::string & text,
    const std::string & style,
    double angle ) [inline]
```

Constructor.

Parameters: see member variables.

### 4.46.3 Member Data Documentation

#### 4.46.3.1 angle

```
double DL_TextData::angle
```

Rotation angle of dimension text away from default orientation.

Referenced by DL\_Dxf::writeText().

#### 4.46.3.2 apx

```
double DL_TextData::apx
```

X Coordinate of alignment point.

Referenced by DL\_Dxf::writeText().

#### 4.46.3.3 apy

```
double DL_TextData::apy
```

Y Coordinate of alignment point.

Referenced by DL\_Dxf::writeText().

#### 4.46.3.4 apz

```
double DL_TextData::apz
```

Z Coordinate of alignment point.

Referenced by DL\_Dxf::writeText().

#### 4.46.3.5 height

```
double DL_TextData::height
```

Text height

Referenced by DL\_Dxf::writeText().

#### 4.46.3.6 hJustification

```
int DL_TextData::hJustification
```

Horizontal justification.

0 = Left (default), 1 = Center, 2 = Right, 3 = Aligned, 4 = Middle, 5 = Fit For 3, 4, 5 the vertical alignment has to be 0.

Referenced by DL\_Dxf::writeText().

#### 4.46.3.7 ipx

```
double DL_TextData::ipx
```

X Coordinate of insertion point.

Referenced by DL\_Dxf::writeText().

#### 4.46.3.8 ipy

```
double DL_TextData::ipy
```

Y Coordinate of insertion point.

Referenced by DL\_Dxf::writeText().

#### 4.46.3.9 ipz

```
double DL_TextData::ipz
```

Z Coordinate of insertion point.

Referenced by DL\_Dxf::writeText().

#### 4.46.3.10 style

```
std::string DL_TextData::style
```

Style (font).

Referenced by DL\_Dxf::writeText().

#### 4.46.3.11 text

```
std::string DL_TextData::text
```

Text string.

Referenced by DL\_Dxf::writeText().

#### 4.46.3.12 textGenerationFlags

```
int DL_TextData::textGenerationFlags
```

0 = default, 2 = Backwards, 4 = Upside down

Referenced by DL\_Dxf::writeText().

#### 4.46.3.13 vJustification

```
int DL_TextData::vJustification
```

Vertical justification.

0 = Baseline (default), 1 = Bottom, 2 = Middle, 3= Top

Referenced by DL\_Dxf::writeText().

#### 4.46.3.14 xScaleFactor

```
double DL_TextData::xScaleFactor
```

Relative X scale factor.

Referenced by DL\_Dxf::writeText().

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

- src/dl\_entities.h

## 4.47 DL\_TraceData Struct Reference

Trace Data / solid data / 3d face data.

```
#include <dl_entities.h>
```

### Public Member Functions

- [DL\\_TraceData](#) (double sx1, double sy1, double sz1, double sx2, double sy2, double sz2, double sx3, double sy3, double sz3, double sx4, double sy4, double sz4, double sthickness=0.0)

*Constructor.*

### Public Attributes

- double [thickness](#)
- double [x](#) [4]
- double [y](#) [4]
- double [z](#) [4]

#### 4.47.1 Detailed Description

Trace Data / solid data / 3d face data.

## 4.47.2 Constructor & Destructor Documentation

### 4.47.2.1 DL\_TraceData()

```
DL_TraceData::DL_TraceData (
    double sx1,
    double sy1,
    double sz1,
    double sx2,
    double sy2,
    double sz2,
    double sx3,
    double sy3,
    double sz3,
    double sx4,
    double sy4,
    double sz4,
    double sthickness = 0.0 ) [inline]
```

Constructor.

Parameters: see member variables.

## 4.47.3 Member Data Documentation

### 4.47.3.1 thickness

```
double DL_TraceData::thickness
```

Thickness

Referenced by DL\_Dxf::writeSolid(), and DL\_Dxf::writeTrace().

### 4.47.3.2 x

```
double DL_TraceData::x[4]
```

Points

Referenced by DL\_Dxf::add3dFace(), DL\_Dxf::addSolid(), DL\_Dxf::addTrace(), DL\_Dxf::write3dFace(), DL\_Dxf::writeSolid(), and DL\_Dxf::writeTrace().

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

- src/dl\_entities.h

## 4.48 DL\_VertexData Struct Reference

Vertex Data.

```
#include <dl_entities.h>
```

### Public Member Functions

- [DL\\_VertexData](#) (double px=0.0, double py=0.0, double pz=0.0, double pBulge=0.0)  
*Constructor.*

### Public Attributes

- double [x](#)
- double [y](#)
- double [z](#)
- double [bulge](#)

#### 4.48.1 Detailed Description

Vertex Data.

#### 4.48.2 Constructor & Destructor Documentation

##### 4.48.2.1 DL\_VertexData()

```
DL_VertexData::DL_VertexData (
    double px = 0.0,
    double py = 0.0,
    double pz = 0.0,
    double pBulge = 0.0 ) [inline]
```

Constructor.

Parameters: see member variables.

#### 4.48.3 Member Data Documentation

#### 4.48.3.1 bulge

```
double DL_VertexData::bulge
```

Bulge of vertex. (The tangent of 1/4 of the arc angle or 0 for lines)

Referenced by DL\_Dxf::writeVertex().

#### 4.48.3.2 x

```
double DL_VertexData::x
```

X Coordinate of the vertex.

Referenced by DL\_Dxf::writeVertex().

#### 4.48.3.3 y

```
double DL_VertexData::y
```

Y Coordinate of the vertex.

Referenced by DL\_Dxf::writeVertex().

#### 4.48.3.4 z

```
double DL_VertexData::z
```

Z Coordinate of the vertex.

Referenced by DL\_Dxf::writeVertex().

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

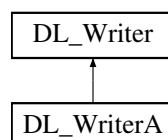
- src/dl\_entities.h

## 4.49 DL\_Writer Class Reference

Defines interface for writing low level DXF constructs to a file.

```
#include <dl_writer.h>
```

Inheritance diagram for DL\_Writer:





## Public Member Functions

- [DL\\_Writer](#) ([DL\\_Codes::version version](#))
- void [section](#) (const char \*name) const  
*Generic section for section 'name'.*
- void [sectionHeader](#) () const  
*Section HEADER.*
- void [sectionTables](#) () const  
*Section TABLES.*
- void [sectionBlocks](#) () const  
*Section BLOCKS.*
- void [sectionEntities](#) () const  
*Section ENTITIES.*
- void [sectionClasses](#) () const  
*Section CLASSES.*
- void [sectionObjects](#) () const  
*Section OBJECTS.*
- void [sectionEnd](#) () const  
*End of a section.*
- void [table](#) (const char \*name, int num, int h=0) const  
*Generic table for table 'name' with 'num' entries:*
- void [tableLayers](#) (int num) const  
*Table for layers.*
- void [tableLinetypes](#) (int num) const  
*Table for line types.*
- void [tableAppid](#) (int num) const  
*Table for application id.*
- void [tableStyle](#) (int num) const  
*Table for text style.*
- void [tableEnd](#) () const  
*End of a table.*
- void [dxfEOF](#) () const  
*End of the DXF file.*
- void [comment](#) (const char \*text) const  
*Comment.*
- void [entity](#) (const char \*entTypeName) const  
*Entity.*
- void [entityAttributes](#) (const [DL\\_Attributes](#) &attrib) const  
*Attributes of an entity.*
- void [subClass](#) (const char \*sub) const  
*Subclass.*
- void [tableLayerEntry](#) (unsigned long int h=0) const  
*Layer (must be in the TABLES section LAYER).*
- void [tableLinetypeEntry](#) (unsigned long int h=0) const  
*Line type (must be in the TABLES section LTYPE).*
- void [tableAppidEntry](#) (unsigned long int h=0) const  
*Appid (must be in the TABLES section APPID).*
- void [sectionBlockEntry](#) (unsigned long int h=0) const  
*Block (must be in the section BLOCKS).*
- void [sectionBlockEntryEnd](#) (unsigned long int h=0) const  
*End of Block (must be in the section BLOCKS).*

- void **color** (int col=256) const
- void **linetype** (const char \*lt) const
- void **linetypeScale** (double scale) const
- void **lineWeight** (int lw) const
- void **coord** (int gc, double x, double y, double z=0) const
- void **coordTriplet** (int gc, const double \*value) const
- void **resetHandle** () const
- unsigned long **handle** (int gc=5) const  
*Writes a unique handle and returns it.*
- unsigned long **getNextHandle** () const
- virtual void **dxReal** (int gc, double value) const =0  
*Must be overwritten by the implementing class to write a real value to the file.*
- virtual void **dxflnt** (int gc, int value) const =0  
*Must be overwritten by the implementing class to write an int value to the file.*
- virtual void **dxfBool** (int gc, bool value) const  
*Can be overwritten by the implementing class to write a bool value to the file.*
- virtual void **dxHex** (int gc, int value) const =0  
*Must be overwritten by the implementing class to write an int value (hex) to the file.*
- virtual void **dxString** (int gc, const char \*value) const =0  
*Must be overwritten by the implementing class to write a string to the file.*
- virtual void **dxString** (int gc, const std::string &value) const =0  
*Must be overwritten by the implementing class to write a string to the file.*

## Protected Attributes

- unsigned long **m\_handle**
- unsigned long **modelSpaceHandle**
- unsigned long **paperSpaceHandle**
- unsigned long **paperSpace0Handle**
- **DL\_Codes::version** version  
*DXF version to be created.*

### 4.49.1 Detailed Description

Defines interface for writing low level DXF constructs to a file.

Implementation is defined in derived classes that write to binary or ASCII files.

Implements functions that write higher level constructs in terms of the low level ones.

**Todo** Add error checking for string/entry length.

### 4.49.2 Constructor & Destructor Documentation

#### 4.49.2.1 DL\_Writer()

```
DL_Writer::DL_Writer (
    DL_Codes::version version ) [inline]
```

## Parameters

<i>version</i>	DXF version. Defaults to DL_VERSION_2002.
----------------	---

### 4.49.3 Member Function Documentation

#### 4.49.3.1 comment()

```
void DL_Writer::comment (
    const char * text ) const [inline]
```

Comment.

```
999
text
```

Referenced by DL\_Dxf::writeHeader().

#### 4.49.3.2 dxfBool()

```
virtual void DL_Writer::dxfBool (
    int gc,
    bool value ) const [inline], [virtual]
```

Can be overwritten by the implementing class to write a bool value to the file.

## Parameters

<i>gc</i>	Group code.
<i>value</i>	The bool value.

Referenced by DL\_Dxf::writeHatchEdge().

#### 4.49.3.3 dxfeof()

```
void DL_Writer::dxfeof ( ) const [inline]
```

End of the DXF file.

```
0
EOF
```

#### 4.49.3.4 dxfHex()

```
virtual void DL_Writer::dxfHex (
    int gc,
    int value ) const [pure virtual]
```

Must be overwritten by the implementing class to write an int value (hex) to the file.

##### Parameters

<i>gc</i>	Group code.
<i>value</i>	The int value.

Implemented in [DL\\_WriterA](#).

#### 4.49.3.5 dxflnt()

```
virtual void DL_Writer::dxflnt (
    int gc,
    int value ) const [pure virtual]
```

Must be overwritten by the implementing class to write an int value to the file.

##### Parameters

<i>gc</i>	Group code.
<i>value</i>	The int value.

Implemented in [DL\\_WriterA](#).

#### 4.49.3.6 dxfReal()

```
virtual void DL_Writer::dxfReal (
    int gc,
    double value ) const [pure virtual]
```

Must be overwritten by the implementing class to write a real value to the file.

##### Parameters

<i>gc</i>	Group code.
<i>value</i>	The real value.

Implemented in [DL\\_WriterA](#).

**4.49.3.7 dxfString()** [1/2]

```
virtual void DL_Writer::dxfString (
    int gc,
    const char * value ) const [pure virtual]
```

Must be overwritten by the implementing class to write a string to the file.

**Parameters**

<i>gc</i>	Group code.
<i>value</i>	The string.

Implemented in [DL\\_WriterA](#).

**4.49.3.8 dxfString()** [2/2]

```
virtual void DL_Writer::dxfString (
    int gc,
    const std::string & value ) const [pure virtual]
```

Must be overwritten by the implementing class to write a string to the file.

**Parameters**

<i>gc</i>	Group code.
<i>value</i>	The string.

Implemented in [DL\\_WriterA](#).

**4.49.3.9 entity()**

```
void DL_Writer::entity (
    const char * entTypeName ) const [inline]
```

Entity.

```
0
entTypeName
```

**Returns**

Unique handle or 0.

Referenced by [DL\\_Dxf::write3dFace\(\)](#), [DL\\_Dxf::writeArc\(\)](#), [DL\\_Dxf::writeCircle\(\)](#), [DL\\_Dxf::writeDimAligned\(\)](#), [DL\\_Dxf::writeDimAngular2L\(\)](#), [DL\\_Dxf::writeDimAngular3P\(\)](#), [DL\\_Dxf::writeDimDiametric\(\)](#), [DL\\_Dxf::writeDimLinear\(\)](#), [DL\\_Dxf::writeDimOrdinate\(\)](#), [DL\\_Dxf::writeDimRadial\(\)](#), [DL\\_Dxf::writeEllipse\(\)](#), [DL\\_Dxf::writeHatch1\(\)](#), [DL\\_Dxf::writeImage\(\)](#), [DL\\_Dxf::writeInsert\(\)](#), [DL\\_Dxf::writeLeader\(\)](#), [DL\\_Dxf::writeLine\(\)](#), [DL\\_Dxf::writeMText\(\)](#), [DL\\_Dxf::writePoint\(\)](#), [DL\\_Dxf::writePolyline\(\)](#), [DL\\_Dxf::writePolylineEnd\(\)](#), [DL\\_Dxf::writeRay\(\)](#), [DL\\_Dxf::writeSolid\(\)](#), [DL\\_Dxf::writeSpline\(\)](#), [DL\\_Dxf::writeText\(\)](#), [DL\\_Dxf::writeTrace\(\)](#), [DL\\_Dxf::writeVertex\(\)](#), and [DL\\_Dxf::writeXLine\(\)](#).

#### 4.49.3.10 entityAttributes()

```
void DL_Writer::entityAttributes (
    const DL_Attributes & attrib ) const [inline]
```

Attributes of an entity.

```
8
layer
62
color
39
width
6
linetype
```

References DL\_Attributes::getColor(), DL\_Attributes::getColor24(), DL\_Attributes::getLayer(), DL\_Attributes::getLinetype(), and DL\_Attributes::getWidth().

Referenced by DL\_Dxf::write3dFace(), DL\_Dxf::writeArc(), DL\_Dxf::writeCircle(), DL\_Dxf::writeDimAligned(), DL\_Dxf::writeDimAngular2L(), DL\_Dxf::writeDimAngular3P(), DL\_Dxf::writeDimDiametric(), DL\_Dxf::writeDimLinear(), DL\_Dxf::writeDimOrdinate(), DL\_Dxf::writeDimRadial(), DL\_Dxf::writeEllipse(), DL\_Dxf::writeHatch1(), DL\_Dxf::writeImage(), DL\_Dxf::writeInsert(), DL\_Dxf::writeLeader(), DL\_Dxf::writeLine(), DL\_Dxf::writeMText(), DL\_Dxf::writePoint(), DL\_Dxf::writePolyline(), DL\_Dxf::writeRay(), DL\_Dxf::writeSolid(), DL\_Dxf::writeSpline(), DL\_Dxf::writeText(), DL\_Dxf::writeTrace(), and DL\_Dxf::writeXLine().

#### 4.49.3.11 getNextHandle()

```
unsigned long DL_Writer::getNextHandle ( ) const [inline]
```

##### Returns

Next handle that will be written.

Referenced by DL\_Dxf::writeObjects().

#### 4.49.3.12 section()

```
void DL_Writer::section (
    const char * name ) const [inline]
```

Generic section for section 'name'.

```
0
SECTION
2
name
```

#### 4.49.3.13 sectionBlockEntry()

```
void DL_Writer::sectionBlockEntry (
    unsigned long int h = 0 ) const [inline]
```

Block (must be in the section BLOCKS).

```
0
BLOCK
```

Referenced by DL\_Dxf::writeBlock().

#### 4.49.3.14 sectionBlockEntryEnd()

```
void DL_Writer::sectionBlockEntryEnd (
    unsigned long int h = 0 ) const [inline]
```

End of Block (must be in the section BLOCKS).

```
0
ENDBLK
```

Referenced by DL\_Dxf::writeEndBlock().

#### 4.49.3.15 sectionBlocks()

```
void DL_Writer::sectionBlocks ( ) const [inline]
```

Section BLOCKS.

```
0
SECTION
2
BLOCKS
```

#### 4.49.3.16 sectionClasses()

```
void DL_Writer::sectionClasses ( ) const [inline]
```

Section CLASSES.

```
0
SECTION
2
CLASSES
```

#### 4.49.3.17 sectionEnd()

```
void DL_Writer::sectionEnd ( ) const [inline]
```

End of a section.

```
0  
ENDSEC
```

#### 4.49.3.18 sectionEntities()

```
void DL_Writer::sectionEntities ( ) const [inline]
```

Section ENTITIES.

```
0  
SECTION  
2  
ENTITIES
```

#### 4.49.3.19 sectionHeader()

```
void DL_Writer::sectionHeader ( ) const [inline]
```

Section HEADER.

```
0  
SECTION  
2  
HEADER
```

Referenced by DL\_Dxf::writeHeader().

#### 4.49.3.20 sectionObjects()

```
void DL_Writer::sectionObjects ( ) const [inline]
```

Section OBJECTS.

```
0  
SECTION  
2  
OBJECTS
```



#### 4.49.3.21 sectionTables()

```
void DL_Writer::sectionTables ( ) const [inline]
```

Section TABLES.

```
0
SECTION
2
TABLES
```

#### 4.49.3.22 table()

```
void DL_Writer::table (
    const char * name,
    int num,
    int h = 0 ) const [inline]
```

Generic table for table 'name' with 'num' entries:

```
0
TABLE
2
name
70
num
```

#### 4.49.3.23 tableAppid()

```
void DL_Writer::tableAppid (
    int num ) const [inline]
```

Table for application id.

##### Parameters

<i>num</i>	Number of registered applications in total.
------------	---

```
0
TABLE
2
APPID
70
num
```

#### 4.49.3.24 tableAppidEntry()

```
void DL_Writer::tableAppidEntry (
    unsigned long int h = 0 ) const [inline]
```

Appid (must be in the TABLES section APPID).

```
0
APPID
```

Referenced by DL\_Dxf::writeAppid().

#### 4.49.3.25 tableEnd()

```
void DL_Writer::tableEnd ( ) const [inline]
```

End of a table.

```
0
ENDTAB
```

#### 4.49.3.26 tableLayerEntry()

```
void DL_Writer::tableLayerEntry (
    unsigned long int h = 0 ) const [inline]
```

Layer (must be in the TABLES section LAYER).

```
0
LAYER
```

Referenced by DL\_Dxf::writeLayer().

#### 4.49.3.27 tableLayers()

```
void DL_Writer::tableLayers (
    int num ) const [inline]
```

Table for layers.

## Parameters

<i>num</i>	Number of layers in total.
------------	----------------------------

```
0
TABLE
2
LAYER
70
    num
```

**4.49.3.28 tableLinetypeEntry()**

```
void DL_Writer::tableLinetypeEntry (
    unsigned long int h = 0 ) const [inline]
```

Line type (must be in the TABLES section LTYPE).

```
0
LTYPE
```

Referenced by DL\_Dxf::writeLinetype().

**4.49.3.29 tableLinetypes()**

```
void DL_Writer::tableLinetypes (
    int num ) const [inline]
```

Table for line types.

## Parameters

<i>num</i>	Number of line types in total.
------------	--------------------------------

```
0
TABLE
2
LTYPE
70
    num
```

#### 4.49.3.30 tableStyle()

```
void DL_Writer::tableStyle (
    int num ) const [inline]
```

Table for text style.

##### Parameters

<i>num</i>	Number of text styles.
------------	------------------------

```
0
TABLE
2
STYLE
70
    num
```

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

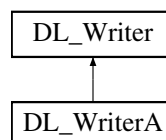
- src/dl\_writer.h

## 4.50 DL\_WriterA Class Reference

Implements functions defined in [DL\\_Writer](#) for writing low level DXF constructs to an ASCII format DXF file.

```
#include <dl_writer_ascii.h>
```

Inheritance diagram for DL\_WriterA:



### Public Member Functions

- **DL\_WriterA** (const char \*fname, [DL\\_Codes::version version](#)=DL\_VERSION\_2000)
- bool [openFailed](#) () const
- void [close](#) () const  
*Closes the output file.*
- void [dxfReal](#) (int gc, double value) const  
*Writes a real (double) variable to the DXF file.*
- void [dxfInt](#) (int gc, int value) const  
*Writes an int variable to the DXF file.*
- void [dxfHex](#) (int gc, int value) const  
*Writes a hex int variable to the DXF file.*
- void [dxfString](#) (int gc, const char \*value) const  
*Writes a string variable to the DXF file.*
- void [dxfString](#) (int gc, const std::string &value) const  
*Must be overwritten by the implementing class to write a string to the file.*

## Static Public Member Functions

- static void [strReplace](#) (char \*str, char src, char dest)  
*Replaces every occurrence of src with dest in the null terminated str.*

## Additional Inherited Members

### 4.50.1 Detailed Description

Implements functions defined in [DL\\_Writer](#) for writing low level DXF constructs to an ASCII format DXF file.

@para fname File name of the file to be created. @para version DXF version. Defaults to DL\_VERSION\_2002.

**Todo** What if fname is NULL? Or fname can't be opened for another reason?

### 4.50.2 Member Function Documentation

#### 4.50.2.1 dxfHex()

```
void DL_WriterA::dxfHex (
    int gc,
    int value ) const [virtual]
```

Writes a hex int variable to the DXF file.

#### Parameters

<i>gc</i>	Group code.
<i>value</i>	Int value

Implements [DL\\_Writer](#).

References [dxfString\(\)](#).

Referenced by [DL\\_Dxf::writeBlockRecord\(\)](#), [DL\\_Dxf::writeDimStyle\(\)](#), [DL\\_Dxf::writeHeader\(\)](#), [DL\\_Dxf::writeImageDef\(\)](#), [DL\\_Dxf::writeLayer\(\)](#), [DL\\_Dxf::writeObjects\(\)](#), [DL\\_Dxf::writeUcs\(\)](#), [DL\\_Dxf::writeView\(\)](#), and [DL\\_Dxf::writeVPort\(\)](#).

#### 4.50.2.2 dxfInt()

```
void DL_WriterA::dxfInt (
    int gc,
    int value ) const [virtual]
```

Writes an int variable to the DXF file.

## Parameters

<i>gc</i>	Group code.
<i>value</i>	Int value

Implements [DL\\_Writer](#).

Referenced by [DL\\_Dxf::writeAppid\(\)](#), [DL\\_Dxf::writeBlock\(\)](#), [DL\\_Dxf::writeBlockRecord\(\)](#), [DL\\_Dxf::writeDimAligned\(\)](#), [DL\\_Dxf::writeDimAngular2L\(\)](#), [DL\\_Dxf::writeDimAngular3P\(\)](#), [DL\\_Dxf::writeDimDiametric\(\)](#), [DL\\_Dxf::writeDimLinear\(\)](#), [DL\\_Dxf::writeDimOrdinate\(\)](#), [DL\\_Dxf::writeDimRadial\(\)](#), [DL\\_Dxf::writeDimStyle\(\)](#), [DL\\_Dxf::writeHatch1\(\)](#), [DL\\_Dxf::writeHatch2\(\)](#), [DL\\_Dxf::writeHatchEdge\(\)](#), [DL\\_Dxf::writeHatchLoop1\(\)](#), [DL\\_Dxf::writeHatchLoop2\(\)](#), [DL\\_Dxf::writeImage\(\)](#), [DL\\_Dxf::writeImageDef\(\)](#), [DL\\_Dxf::writeInsert\(\)](#), [DL\\_Dxf::writeLayer\(\)](#), [DL\\_Dxf::writeLeader\(\)](#), [DL\\_Dxf::writeLinetype\(\)](#), [DL\\_Dxf::writeMText\(\)](#), [DL\\_Dxf::writeObjects\(\)](#), [DL\\_Dxf::writePolyline\(\)](#), [DL\\_Dxf::writeSpline\(\)](#), [DL\\_Dxf::writeStyle\(\)](#), [DL\\_Dxf::writeText\(\)](#), [DL\\_Dxf::writeUcs\(\)](#), [DL\\_Dxf::writeView\(\)](#), and [DL\\_Dxf::writeVPort\(\)](#).

#### 4.50.2.3 dxfReal()

```
void DL_WriterA::dxfReal (
    int gc,
    double value ) const [virtual]
```

Writes a real (double) variable to the DXF file.

## Parameters

<i>gc</i>	Group code.
<i>value</i>	Double value

Implements [DL\\_Writer](#).

References [dxfString\(\)](#), [strReplace\(\)](#), and [DL\\_Writer::version](#).

Referenced by [DL\\_Dxf::writeArc\(\)](#), [DL\\_Dxf::writeCircle\(\)](#), [DL\\_Dxf::writeControlPoint\(\)](#), [DL\\_Dxf::writeDimAligned\(\)](#), [DL\\_Dxf::writeDimAngular2L\(\)](#), [DL\\_Dxf::writeDimAngular3P\(\)](#), [DL\\_Dxf::writeDimDiametric\(\)](#), [DL\\_Dxf::writeDimLinear\(\)](#), [DL\\_Dxf::writeDimOrdinate\(\)](#), [DL\\_Dxf::writeDimRadial\(\)](#), [DL\\_Dxf::writeDimStyle\(\)](#), [DL\\_Dxf::writeEllipse\(\)](#), [DL\\_Dxf::writeFitPoint\(\)](#), [DL\\_Dxf::writeHatch1\(\)](#), [DL\\_Dxf::writeHatch2\(\)](#), [DL\\_Dxf::writeHatchEdge\(\)](#), [DL\\_Dxf::writeImage\(\)](#), [DL\\_Dxf::writeImageDef\(\)](#), [DL\\_Dxf::writeInsert\(\)](#), [DL\\_Dxf::writeKnot\(\)](#), [DL\\_Dxf::writeLeader\(\)](#), [DL\\_Dxf::writeLeaderVertex\(\)](#), [DL\\_Dxf::writeLinetype\(\)](#), [DL\\_Dxf::writeMText\(\)](#), [DL\\_Dxf::writeObjects\(\)](#), [DL\\_Dxf::writeSolid\(\)](#), [DL\\_Dxf::writeStyle\(\)](#), [DL\\_Dxf::writeText\(\)](#), [DL\\_Dxf::writeTrace\(\)](#), [DL\\_Dxf::writeVertex\(\)](#), and [DL\\_Dxf::writeVPort\(\)](#).

#### 4.50.2.4 dxfString() [1/2]

```
void DL_WriterA::dxfString (
    int gc,
    const char * value ) const [virtual]
```

Writes a string variable to the DXF file.

## Parameters

<i>gc</i>	Group code.
<i>value</i>	String

Implements [DL\\_Writer](#).

Referenced by `dxHex()`, `dxReal()`, `DL_Dxf::write3dFace()`, `DL_Dxf::writeAppid()`, `DL_Dxf::writeArc()`, `DL_Dxf::writeBlock()`, `DL_Dxf::writeBlockRecord()`, `DL_Dxf::writeCircle()`, `DL_Dxf::writeComment()`, `DL_Dxf::writeDimAligned()`, `DL_Dxf::writeDimAngular2L()`, `DL_Dxf::writeDimAngular3P()`, `DL_Dxf::writeDimDiametric()`, `DL_Dxf::writeDimLinear()`, `DL_Dxf::writeDimOrdinate()`, `DL_Dxf::writeDimRadial()`, `DL_Dxf::writeDimStyle()`, `DL_Dxf::writeEllipse()`, `DL_Dxf::writeHatch1()`, `DL_Dxf::writeHatch2()`, `DL_Dxf::writeHeader()`, `DL_Dxf::writeImage()`, `DL_Dxf::writeImageDef()`, `DL_Dxf::writeInsert()`, `DL_Dxf::writeLayer()`, `DL_Dxf::writeLeader()`, `DL_Dxf::writeLine()`, `DL_Dxf::writeLinetype()`, `DL_Dxf::writeMText()`, `DL_Dxf::writeObjects()`, `DL_Dxf::writeObjectsEnd()`, `DL_Dxf::writePoint()`, `DL_Dxf::writePolyline()`, `DL_Dxf::writeRay()`, `DL_Dxf::writeSolid()`, `DL_Dxf::writeSpline()`, `DL_Dxf::writeStyle()`, `DL_Dxf::writeText()`, `DL_Dxf::writeTrace()`, `DL_Dxf::writeUcs()`, `DL_Dxf::writeVertex()`, `DL_Dxf::writeView()`, `DL_Dxf::writeVPort()`, and `DL_Dxf::writeXLine()`.

4.50.2.5 `dxString()` [2/2]

```
void DL_WriterA::dxString (
    int gc,
    const std::string & value ) const [virtual]
```

Must be overwritten by the implementing class to write a string to the file.

## Parameters

<i>gc</i>	Group code.
<i>value</i>	The string.

Implements [DL\\_Writer](#).

4.50.2.6 `openFailed()`

```
bool DL_WriterA::openFailed ( ) const
```

## Return values

<i>true</i>	Opening file has failed.
<i>false</i>	Otherwise.

Referenced by `DL_Dxf::out()`.

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

- src/dl\_writer\_ascii.h
- src/dl\_writer\_ascii.cpp

## 4.51 DL\_XLineData Struct Reference

XLine Data.

```
#include <dl_entities.h>
```

### Public Member Functions

- [DL\\_XLineData](#) (double [bx](#), double [by](#), double [bz](#), double [dx](#), double [dy](#), double [dz](#))  
*Constructor.*

### Public Attributes

- double [bx](#)
- double [by](#)
- double [bz](#)
- double [dx](#)
- double [dy](#)
- double [dz](#)

### 4.51.1 Detailed Description

XLine Data.

### 4.51.2 Constructor & Destructor Documentation

#### 4.51.2.1 DL\_XLineData()

```
DL_XLineData::DL_XLineData (  
    double bx,  
    double by,  
    double bz,  
    double dx,  
    double dy,  
    double dz ) [inline]
```

Constructor.

Parameters: see member variables.



### 4.51.3 Member Data Documentation

#### 4.51.3.1 bx

```
double DL_XLineData::bx
```

X base point.

Referenced by DL\_Dxf::writeXLine().

#### 4.51.3.2 by

```
double DL_XLineData::by
```

Y base point.

Referenced by DL\_Dxf::writeXLine().

#### 4.51.3.3 bz

```
double DL_XLineData::bz
```

Z base point.

Referenced by DL\_Dxf::writeXLine().

#### 4.51.3.4 dx

```
double DL_XLineData::dx
```

X direction vector.

Referenced by DL\_Dxf::writeXLine().

#### 4.51.3.5 dy

```
double DL_XLineData::dy
```

Y direction vector.

Referenced by DL\_Dxf::writeXLine().

#### 4.51.3.6 dz

```
double DL_XLineData::dz
```

Z direction vector.

Referenced by DL\_Dxf::writeXLine().

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

- src/dl\_entities.h



# Index

- addAttribute
  - DL\_Dxf, [68](#)
- addBlock
  - DL\_CreationAdapter, [32](#)
  - DL\_CreationInterface, [37](#)
- addMTextChunk
  - DL\_CreationAdapter, [32](#)
  - DL\_CreationInterface, [37](#)
- addSolid
  - DL\_Dxf, [68](#)
- addTrace
  - DL\_Dxf, [68](#)
- alignment
  - DL\_ArcAlignedTextData, [10](#)
- angle
  - DL\_DimLinearData, [57](#)
  - DL\_HatchData, [101](#)
  - DL\_InsertData, [117](#)
  - DL\_MTextData, [131](#)
  - DL\_TextData, [144](#)
- angle1
  - DL\_ArcData, [16](#)
  - DL\_EllipseData, [94](#)
  - DL\_HatchEdgeData, [105](#)
- angle2
  - DL\_ArcData, [16](#)
  - DL\_EllipseData, [95](#)
  - DL\_HatchEdgeData, [105](#)
- apx
  - DL\_TextData, [144](#)
- apy
  - DL\_TextData, [144](#)
- apz
  - DL\_TextData, [145](#)
- arcHandle
  - DL\_ArcAlignedTextData, [10](#)
- arrowHeadFlag
  - DL\_LeaderData, [122](#)
- attachmentPoint
  - DL\_DimensionData, [53](#)
  - DL\_MTextData, [131](#)
- bold
  - DL\_ArcAlignedTextData, [10](#)
- brightness
  - DL\_ImageData, [111](#)
- bulge
  - DL\_VertexData, [149](#)
- bx
  - DL\_RayData, [138](#)
  - DL\_XLineData, [167](#)
- by
  - DL\_RayData, [139](#)
  - DL\_XLineData, [167](#)
- bz
  - DL\_RayData, [139](#)
  - DL\_XLineData, [167](#)
- ccw
  - DL\_HatchEdgeData, [106](#)
- characerSet
  - DL\_ArcAlignedTextData, [10](#)
- checkVariable
  - DL\_Dxf, [68](#)
- cols
  - DL\_InsertData, [117](#)
- colSp
  - DL\_InsertData, [117](#)
- comment
  - DL\_Writer, [153](#)
- contrast
  - DL\_ImageData, [111](#)
- cx
  - DL\_ArcAlignedTextData, [10](#)
  - DL\_ArcData, [16](#)
  - DL\_CircleData, [25](#)
  - DL\_EllipseData, [95](#)
  - DL\_HatchEdgeData, [106](#)
- cy
  - DL\_ArcAlignedTextData, [11](#)
  - DL\_ArcData, [17](#)
  - DL\_CircleData, [26](#)
  - DL\_EllipseData, [95](#)
  - DL\_HatchEdgeData, [106](#)
- cz
  - DL\_ArcAlignedTextData, [11](#)
  - DL\_ArcData, [17](#)
  - DL\_CircleData, [26](#)
  - DL\_EllipseData, [95](#)
- degree
  - DL\_HatchEdgeData, [106](#)
  - DL\_SplineData, [141](#)
- dimScale
  - DL\_LeaderData, [123](#)
- direction
  - DL\_ArcAlignedTextData, [11](#)
- dirx
  - DL\_MTextData, [131](#)
- diry

- DL\_MTextData, 131
- dirz
  - DL\_MTextData, 131
- DL\_ArcAlignedTextData, 9
  - alignment, 10
  - arcHandle, 10
  - bold, 10
  - characerSet, 10
  - cx, 10
  - cy, 11
  - cz, 11
  - direction, 11
  - endAngle, 11
  - font, 11
  - height, 12
  - italic, 12
  - leftOffset, 12
  - offset, 12
  - pitch, 12
  - radius, 13
  - reversedCharacterOrder, 13
  - rightOffset, 13
  - shxFont, 13
  - side, 13
  - spacing, 14
  - startAngle, 14
  - style, 14
  - text, 14
  - underline, 14
  - wizard, 15
  - xScaleFactor, 15
- DL\_ArcData, 15
  - angle1, 16
  - angle2, 16
  - cx, 16
  - cy, 17
  - cz, 17
  - DL\_ArcData, 16
  - radius, 17
- DL\_AttributeData, 17
  - DL\_AttributeData, 18
  - tag, 18
- DL\_Attributes, 19
  - DL\_Attributes, 20
  - getColor, 21
  - getColor24, 21
  - getLayer, 21
  - getLinetype, 22
  - getWidth, 22
  - setColor, 22
  - setColor24, 22
  - setLayer, 23
  - setLinetype, 23
- DL\_BlockData, 23
  - DL\_BlockData, 24
  - flags, 24
- DL\_CircleData, 25
  - cx, 25
  - cy, 26
  - cz, 26
  - DL\_CircleData, 25
  - radius, 26
- DL\_Codes, 26
- DL\_ControlPointData, 27
  - DL\_ControlPointData, 27
  - w, 28
  - x, 28
  - y, 28
  - z, 28
- DL\_CreationAdapter, 29
  - addBlock, 32
  - addMTextChunk, 32
  - endEntity, 32
  - processCodeValuePair, 32
  - setVariableDouble, 33
  - setVariableInt, 33
  - setVariableString, 33
  - setVariableVector, 33
- DL\_CreationInterface, 34
  - addBlock, 37
  - addMTextChunk, 37
  - endEntity, 38
  - getAttributes, 38
  - getExtrusion, 38
  - processCodeValuePair, 38
  - setVariableDouble, 39
  - setVariableInt, 39
  - setVariableString, 39
  - setVariableVector, 40
- DL\_DictionaryData, 40
- DL\_DictionaryEntryData, 41
- DL\_DimAlignedData, 41
  - DL\_DimAlignedData, 42
  - epx1, 42
  - epx2, 42
  - epy1, 43
  - epy2, 43
  - epz1, 43
  - epz2, 43
- DL\_DimAngular2LData, 44
  - DL\_DimAngular2LData, 44
  - dpx1, 45
  - dpx2, 45
  - dpx3, 45
  - dpx4, 45
  - dpy1, 45
  - dpy2, 45
  - dpy3, 46
  - dpy4, 46
  - dpz1, 46
  - dpz2, 46
  - dpz3, 46
  - dpz4, 46
- DL\_DimAngular3PData, 47
  - DL\_DimAngular3PData, 47
  - dpx1, 48

- dpx2, 48
- dpx3, 48
- dpy1, 48
- dpy2, 49
- dpy3, 49
- dpz1, 49
- dpz2, 49
- dpz3, 49
- DL\_DimDiametricData, 50
  - DL\_DimDiametricData, 50
  - dpx, 50
  - dpy, 51
  - dpz, 51
  - leader, 51
- DL\_DimensionData, 51
  - attachmentPoint, 53
  - DL\_DimensionData, 52
  - dpx, 53
  - dpy, 53
  - dpz, 53
  - lineSpacingFactor, 54
  - lineSpacingStyle, 54
  - mpx, 54
  - mpy, 54
  - mpz, 55
  - style, 55
  - text, 55
  - type, 55
- DL\_DimLinearData, 56
  - angle, 57
  - DL\_DimLinearData, 56
  - dpx1, 57
  - dpx2, 57
  - dpy1, 57
  - dpy2, 57
  - dpz1, 57
  - dpz2, 58
  - oblique, 58
- DL\_DimOrdinateData, 58
  - DL\_DimOrdinateData, 59
  - dpx1, 59
  - dpx2, 59
  - dpy1, 59
  - dpy2, 59
  - dpz1, 60
  - dpz2, 60
  - xtype, 60
- DL\_DimRadialData, 60
  - DL\_DimRadialData, 61
  - dpx, 61
  - dpy, 61
  - dpz, 61
  - leader, 62
- DL\_Dxf, 62
  - addAttribute, 68
  - addSolid, 68
  - addTrace, 68
  - checkVariable, 68
  - getDimData, 69
  - getLibVersion, 69
  - getStrippedLine, 69
  - in, 70
  - out, 71
  - processDXFGroup, 71
  - readDxfGroups, 72
  - stripWhiteSpace, 73
  - test, 73
  - write3dFace, 73
  - writeAppid, 74
  - writeArc, 74
  - writeBlockRecord, 74
  - writeCircle, 74
  - writeControlPoint, 75
  - writeDimAligned, 75
  - writeDimAngular2L, 76
  - writeDimAngular3P, 76
  - writeDimDiametric, 77
  - writeDimLinear, 77
  - writeDimOrdinate, 78
  - writeDimRadial, 78
  - writeDimStyle, 79
  - writeEllipse, 79
  - writeEndBlock, 79
  - writeFitPoint, 80
  - writeHatch1, 80
  - writeHatch2, 80
  - writeHatchEdge, 81
  - writeHatchLoop1, 81
  - writeHatchLoop2, 82
  - writeImage, 82
  - writeInsert, 82
  - writeKnot, 84
  - writeLayer, 84
  - writeLeader, 85
  - writeLeaderVertex, 85
  - writeLine, 85
  - writeLinetype, 86
  - writeMText, 86
  - writeObjects, 87
  - writeObjectsEnd, 87
  - writePoint, 87
  - writePolyline, 88
  - writePolylineEnd, 88
  - writeRay, 88
  - writeSolid, 89
  - writeSpline, 89
  - writeStyle, 90
  - writeText, 90
  - writeTrace, 90
  - writeUcs, 92
  - writeVertex, 92
  - writeView, 92
  - writeVPort, 93
  - writeXLine, 93
- DL\_EllipseData, 93
  - angle1, 94

- angle2, [95](#)
- cx, [95](#)
- cy, [95](#)
- cz, [95](#)
- DL\_EllipseData, [94](#)
- mx, [95](#)
- my, [96](#)
- mz, [96](#)
- ratio, [96](#)
- DL\_Exception, [96](#)
- DL\_Extrusion, [97](#)
  - DL\_Extrusion, [97](#)
  - getDirection, [98](#)
  - getElevation, [98](#)
- DL\_FitPointData, [99](#)
  - DL\_FitPointData, [99](#)
  - x, [99](#)
  - y, [99](#)
  - z, [100](#)
- DL\_GroupCodeExc, [100](#)
- DL\_HatchData, [100](#)
  - angle, [101](#)
  - DL\_HatchData, [101](#)
  - numLoops, [102](#)
  - originX, [102](#)
  - pattern, [102](#)
  - scale, [102](#)
  - solid, [102](#)
- DL\_HatchEdgeData, [103](#)
  - angle1, [105](#)
  - angle2, [105](#)
  - ccw, [106](#)
  - cx, [106](#)
  - cy, [106](#)
  - degree, [106](#)
  - DL\_HatchEdgeData, [104](#), [105](#)
  - mx, [106](#)
  - my, [107](#)
  - nControl, [107](#)
  - nFit, [107](#)
  - nKnots, [107](#)
  - radius, [107](#)
  - ratio, [108](#)
  - type, [108](#)
  - x1, [108](#)
  - x2, [108](#)
  - y1, [108](#)
  - y2, [109](#)
- DL\_HatchLoopData, [109](#)
  - DL\_HatchLoopData, [109](#)
  - numEdges, [110](#)
- DL\_ImageData, [110](#)
  - brightness, [111](#)
  - contrast, [111](#)
  - DL\_ImageData, [111](#)
  - fade, [112](#)
  - height, [112](#)
  - ipx, [112](#)
  - ipy, [112](#)
  - ipz, [112](#)
  - ref, [113](#)
  - ux, [113](#)
  - uy, [113](#)
  - uz, [113](#)
  - vx, [113](#)
  - vy, [114](#)
  - vz, [114](#)
  - width, [114](#)
- DL\_ImageDefData, [114](#)
  - DL\_ImageDefData, [115](#)
  - file, [115](#)
  - ref, [115](#)
- DL\_InsertData, [116](#)
  - angle, [117](#)
  - cols, [117](#)
  - colSp, [117](#)
  - DL\_InsertData, [116](#)
  - ipx, [117](#)
  - ipy, [117](#)
  - ipz, [117](#)
  - name, [118](#)
  - rows, [118](#)
  - rowSp, [118](#)
  - sx, [118](#)
  - sy, [118](#)
  - sz, [119](#)
- DL\_KnotData, [119](#)
  - DL\_KnotData, [119](#)
  - k, [120](#)
- DL\_LayerData, [120](#)
  - DL\_LayerData, [121](#)
  - flags, [121](#)
- DL\_LeaderData, [121](#)
  - arrowHeadFlag, [122](#)
  - dimScale, [123](#)
  - DL\_LeaderData, [122](#)
  - hooklineDirectionFlag, [123](#)
  - hooklineFlag, [123](#)
  - leaderCreationFlag, [123](#)
  - leaderPathType, [123](#)
  - number, [124](#)
  - textAnnotationHeight, [124](#)
  - textAnnotationWidth, [124](#)
- DL\_LeaderVertexData, [124](#)
  - DL\_LeaderVertexData, [125](#)
  - x, [125](#)
  - y, [125](#)
  - z, [126](#)
- DL\_LineData, [126](#)
  - DL\_LineData, [126](#)
  - x1, [127](#)
  - x2, [127](#)
  - y1, [127](#)
  - y2, [127](#)
  - z1, [128](#)
  - z2, [128](#)

- DL\_LinetypeData, 128
  - DL\_LinetypeData, 129
- DL\_MTextData, 129
  - angle, 131
  - attachmentPoint, 131
  - dirx, 131
  - diry, 131
  - dirz, 131
  - DL\_MTextData, 130
  - drawingDirection, 132
  - height, 132
  - ipx, 132
  - ipy, 132
  - ipz, 132
  - lineSpacingFactor, 133
  - lineSpacingStyle, 133
  - style, 133
  - text, 133
  - width, 133
- DL\_NullStrExc, 134
- DL\_PointData, 134
  - DL\_PointData, 135
  - x, 135
  - y, 135
  - z, 135
- DL\_PolylineData, 136
  - DL\_PolylineData, 136
  - elevation, 137
  - flags, 137
  - m, 137
  - n, 137
  - number, 137
- DL\_RayData, 138
  - bx, 138
  - by, 139
  - bz, 139
  - DL\_RayData, 138
  - dx, 139
  - dy, 139
  - dz, 139
- DL\_SplineData, 140
  - degree, 141
  - DL\_SplineData, 140
  - flags, 141
  - nControl, 141
  - nFit, 141
  - nKnots, 142
- DL\_StyleData, 142
- DL\_TextData, 143
  - angle, 144
  - apx, 144
  - apy, 144
  - apz, 145
  - DL\_TextData, 144
  - height, 145
  - hJustification, 145
  - ipx, 145
  - ipy, 145
  - ipz, 146
  - style, 146
  - text, 146
  - textGenerationFlags, 146
  - vJustification, 146
  - xScaleFactor, 147
- DL\_TraceData, 147
  - DL\_TraceData, 148
  - thickness, 148
  - x, 148
- DL\_VertexData, 149
  - bulge, 149
  - DL\_VertexData, 149
  - x, 150
  - y, 150
  - z, 150
- DL\_Writer, 150
  - comment, 153
  - DL\_Writer, 152
  - dxfBool, 153
  - dxfEOF, 153
  - dxfHex, 153
  - dxflnt, 154
  - dxflReal, 154
  - dxflString, 154, 155
  - entity, 155
  - entityAttributes, 155
  - getNextHandle, 156
  - section, 156
  - sectionBlockEntry, 156
  - sectionBlockEntryEnd, 157
  - sectionBlocks, 157
  - sectionClasses, 157
  - sectionEnd, 157
  - sectionEntities, 158
  - sectionHeader, 158
  - sectionObjects, 158
  - sectionTables, 158
  - table, 159
  - tableAppid, 159
  - tableAppidEntry, 159
  - tableEnd, 160
  - tableLayerEntry, 160
  - tableLayers, 160
  - tableLinetypeEntry, 161
  - tableLinetypes, 161
  - tableStyle, 161
- DL\_WriterA, 162
  - dxflHex, 163
  - dxflnt, 163
  - dxflReal, 164
  - dxflString, 164, 165
  - openFailed, 165
- DL\_XLineData, 166
  - bx, 167
  - by, 167
  - bz, 167
  - DL\_XLineData, 166

- dx, [167](#)
- dy, [167](#)
- dz, [167](#)
- dpx
  - DL\_DimDiametricData, [50](#)
  - DL\_DimensionData, [53](#)
  - DL\_DimRadialData, [61](#)
- dpx1
  - DL\_DimAngular2LData, [45](#)
  - DL\_DimAngular3PData, [48](#)
  - DL\_DimLinearData, [57](#)
  - DL\_DimOrdinateData, [59](#)
- dpx2
  - DL\_DimAngular2LData, [45](#)
  - DL\_DimAngular3PData, [48](#)
  - DL\_DimLinearData, [57](#)
  - DL\_DimOrdinateData, [59](#)
- dpx3
  - DL\_DimAngular2LData, [45](#)
  - DL\_DimAngular3PData, [48](#)
- dpx4
  - DL\_DimAngular2LData, [45](#)
- dpy
  - DL\_DimDiametricData, [51](#)
  - DL\_DimensionData, [53](#)
  - DL\_DimRadialData, [61](#)
- dpy1
  - DL\_DimAngular2LData, [45](#)
  - DL\_DimAngular3PData, [48](#)
  - DL\_DimLinearData, [57](#)
  - DL\_DimOrdinateData, [59](#)
- dpy2
  - DL\_DimAngular2LData, [45](#)
  - DL\_DimAngular3PData, [49](#)
  - DL\_DimLinearData, [57](#)
  - DL\_DimOrdinateData, [59](#)
- dpy3
  - DL\_DimAngular2LData, [46](#)
  - DL\_DimAngular3PData, [49](#)
- dpy4
  - DL\_DimAngular2LData, [46](#)
- dpz
  - DL\_DimDiametricData, [51](#)
  - DL\_DimensionData, [53](#)
  - DL\_DimRadialData, [61](#)
- dpz1
  - DL\_DimAngular2LData, [46](#)
  - DL\_DimAngular3PData, [49](#)
  - DL\_DimLinearData, [57](#)
  - DL\_DimOrdinateData, [60](#)
- dpz2
  - DL\_DimAngular2LData, [46](#)
  - DL\_DimAngular3PData, [49](#)
  - DL\_DimLinearData, [58](#)
  - DL\_DimOrdinateData, [60](#)
- dpz3
  - DL\_DimAngular2LData, [46](#)
  - DL\_DimAngular3PData, [49](#)
- dpz4
  - DL\_DimAngular2LData, [46](#)
- drawingDirection
  - DL\_MTextData, [132](#)
- dx
  - DL\_RayData, [139](#)
  - DL\_XLineData, [167](#)
- dxflBool
  - DL\_Writer, [153](#)
- dxflEOF
  - DL\_Writer, [153](#)
- dxflHex
  - DL\_Writer, [153](#)
  - DL\_WriterA, [163](#)
- dxflInt
  - DL\_Writer, [154](#)
  - DL\_WriterA, [163](#)
- dxflReal
  - DL\_Writer, [154](#)
  - DL\_WriterA, [164](#)
- dxflString
  - DL\_Writer, [154](#), [155](#)
  - DL\_WriterA, [164](#), [165](#)
- dy
  - DL\_RayData, [139](#)
  - DL\_XLineData, [167](#)
- dz
  - DL\_RayData, [139](#)
  - DL\_XLineData, [167](#)
- elevation
  - DL\_PolylineData, [137](#)
- endAngle
  - DL\_ArcAlignedTextData, [11](#)
- endEntity
  - DL\_CreationAdapter, [32](#)
  - DL\_CreationInterface, [38](#)
- entity
  - DL\_Writer, [155](#)
- entityAttributes
  - DL\_Writer, [155](#)
- epx1
  - DL\_DimAlignedData, [42](#)
- epx2
  - DL\_DimAlignedData, [42](#)
- epy1
  - DL\_DimAlignedData, [43](#)
- epy2
  - DL\_DimAlignedData, [43](#)
- epz1
  - DL\_DimAlignedData, [43](#)
- epz2
  - DL\_DimAlignedData, [43](#)
- fade
  - DL\_ImageData, [112](#)
- file
  - DL\_ImageDefData, [115](#)
- flags



- DL\_BlockData, [24](#)
- DL\_LayerData, [121](#)
- DL\_PolylineData, [137](#)
- DL\_SplineData, [141](#)
- font
  - DL\_ArcAlignedTextData, [11](#)
- getAttributes
  - DL\_CreationInterface, [38](#)
- getColor
  - DL\_Attributes, [21](#)
- getColor24
  - DL\_Attributes, [21](#)
- getDimData
  - DL\_Dxf, [69](#)
- getDirection
  - DL\_Extrusion, [98](#)
- getElevation
  - DL\_Extrusion, [98](#)
- getExtrusion
  - DL\_CreationInterface, [38](#)
- getLayer
  - DL\_Attributes, [21](#)
- getLibVersion
  - DL\_Dxf, [69](#)
- getLinetype
  - DL\_Attributes, [22](#)
- getNextHandle
  - DL\_Writer, [156](#)
- getStrippedLine
  - DL\_Dxf, [69](#)
- getWidth
  - DL\_Attributes, [22](#)
- height
  - DL\_ArcAlignedTextData, [12](#)
  - DL\_ImageData, [112](#)
  - DL\_MTextData, [132](#)
  - DL\_TextData, [145](#)
- hJustification
  - DL\_TextData, [145](#)
- hooklineDirectionFlag
  - DL\_LeaderData, [123](#)
- hooklineFlag
  - DL\_LeaderData, [123](#)
- in
  - DL\_Dxf, [70](#)
- ipx
  - DL\_ImageData, [112](#)
  - DL\_InsertData, [117](#)
  - DL\_MTextData, [132](#)
  - DL\_TextData, [145](#)
- ipy
  - DL\_ImageData, [112](#)
  - DL\_InsertData, [117](#)
  - DL\_MTextData, [132](#)
  - DL\_TextData, [145](#)
- ipz
  - DL\_ImageData, [112](#)
  - DL\_InsertData, [117](#)
  - DL\_MTextData, [132](#)
  - DL\_TextData, [146](#)
- italic
  - DL\_ArcAlignedTextData, [12](#)
- k
  - DL\_KnotData, [120](#)
- leader
  - DL\_DimDiametricData, [51](#)
  - DL\_DimRadialData, [62](#)
- leaderCreationFlag
  - DL\_LeaderData, [123](#)
- leaderPathType
  - DL\_LeaderData, [123](#)
- leftOffset
  - DL\_ArcAlignedTextData, [12](#)
- lineSpacingFactor
  - DL\_DimensionData, [54](#)
  - DL\_MTextData, [133](#)
- lineSpacingStyle
  - DL\_DimensionData, [54](#)
  - DL\_MTextData, [133](#)
- m
  - DL\_PolylineData, [137](#)
- mpx
  - DL\_DimensionData, [54](#)
- mpy
  - DL\_DimensionData, [54](#)
- mpz
  - DL\_DimensionData, [55](#)
- mx
  - DL\_EllipseData, [95](#)
  - DL\_HatchEdgeData, [106](#)
- my
  - DL\_EllipseData, [96](#)
  - DL\_HatchEdgeData, [107](#)
- mz
  - DL\_EllipseData, [96](#)
- n
  - DL\_PolylineData, [137](#)
- name
  - DL\_InsertData, [118](#)
- nControl
  - DL\_HatchEdgeData, [107](#)
  - DL\_SplineData, [141](#)
- nFit
  - DL\_HatchEdgeData, [107](#)
  - DL\_SplineData, [141](#)
- nKnots
  - DL\_HatchEdgeData, [107](#)
  - DL\_SplineData, [142](#)
- number
  - DL\_LeaderData, [124](#)
  - DL\_PolylineData, [137](#)

- numEdges
  - DL\_HatchLoopData, 110
- numLoops
  - DL\_HatchData, 102
- oblique
  - DL\_DimLinearData, 58
- offset
  - DL\_ArcAlignedTextData, 12
- openFailed
  - DL\_WriterA, 165
- originX
  - DL\_HatchData, 102
- out
  - DL\_Dxf, 71
- pattern
  - DL\_HatchData, 102
- pitch
  - DL\_ArcAlignedTextData, 12
- processCodeValuePair
  - DL\_CreationAdapter, 32
  - DL\_CreationInterface, 38
- processDXFGroup
  - DL\_Dxf, 71
- radius
  - DL\_ArcAlignedTextData, 13
  - DL\_ArcData, 17
  - DL\_CircleData, 26
  - DL\_HatchEdgeData, 107
- ratio
  - DL\_EllipseData, 96
  - DL\_HatchEdgeData, 108
- readDxfGroups
  - DL\_Dxf, 72
- ref
  - DL\_ImageData, 113
  - DL\_ImageDefData, 115
- reversedCharacterOrder
  - DL\_ArcAlignedTextData, 13
- rightOffset
  - DL\_ArcAlignedTextData, 13
- rows
  - DL\_InsertData, 118
- rowSp
  - DL\_InsertData, 118
- scale
  - DL\_HatchData, 102
- section
  - DL\_Writer, 156
- sectionBlockEntry
  - DL\_Writer, 156
- sectionBlockEntryEnd
  - DL\_Writer, 157
- sectionBlocks
  - DL\_Writer, 157
- sectionClasses
  - DL\_Writer, 157
- sectionEnd
  - DL\_Writer, 157
- sectionEntities
  - DL\_Writer, 158
- sectionHeader
  - DL\_Writer, 158
- sectionObjects
  - DL\_Writer, 158
- sectionTables
  - DL\_Writer, 158
- setColor
  - DL\_Attributes, 22
- setColor24
  - DL\_Attributes, 22
- setLayer
  - DL\_Attributes, 23
- setLinetype
  - DL\_Attributes, 23
- setVariableDouble
  - DL\_CreationAdapter, 33
  - DL\_CreationInterface, 39
- setVariableInt
  - DL\_CreationAdapter, 33
  - DL\_CreationInterface, 39
- setVariableString
  - DL\_CreationAdapter, 33
  - DL\_CreationInterface, 39
- setVariableVector
  - DL\_CreationAdapter, 33
  - DL\_CreationInterface, 40
- shxFont
  - DL\_ArcAlignedTextData, 13
- side
  - DL\_ArcAlignedTextData, 13
- solid
  - DL\_HatchData, 102
- spacing
  - DL\_ArcAlignedTextData, 14
- startAngle
  - DL\_ArcAlignedTextData, 14
- stripWhiteSpace
  - DL\_Dxf, 73
- style
  - DL\_ArcAlignedTextData, 14
  - DL\_DimensionData, 55
  - DL\_MTextData, 133
  - DL\_TextData, 146
- sx
  - DL\_InsertData, 118
- sy
  - DL\_InsertData, 118
- sz
  - DL\_InsertData, 119
- table
  - DL\_Writer, 159
- tableAppid
  - DL\_Writer, 159

- tableAppidEntry
  - DL\_Writer, 159
- tableEnd
  - DL\_Writer, 160
- tableLayerEntry
  - DL\_Writer, 160
- tableLayers
  - DL\_Writer, 160
- tableLinetypeEntry
  - DL\_Writer, 161
- tableLinetypes
  - DL\_Writer, 161
- tableStyle
  - DL\_Writer, 161
- tag
  - DL\_AttributeData, 18
- test
  - DL\_Dxf, 73
- text
  - DL\_ArcAlignedTextData, 14
  - DL\_DimensionData, 55
  - DL\_MTextData, 133
  - DL\_TextData, 146
- textAnnotationHeight
  - DL\_LeaderData, 124
- textAnnotationWidth
  - DL\_LeaderData, 124
- textGenerationFlags
  - DL\_TextData, 146
- thickness
  - DL\_TraceData, 148
- type
  - DL\_DimensionData, 55
  - DL\_HatchEdgeData, 108
- underline
  - DL\_ArcAlignedTextData, 14
- ux
  - DL\_ImageData, 113
- uy
  - DL\_ImageData, 113
- uz
  - DL\_ImageData, 113
- vJustification
  - DL\_TextData, 146
- vx
  - DL\_ImageData, 113
- vy
  - DL\_ImageData, 114
- vz
  - DL\_ImageData, 114
- w
  - DL\_ControlPointData, 28
- width
  - DL\_ImageData, 114
  - DL\_MTextData, 133
- wizard
  - DL\_ArcAlignedTextData, 15
- write3dFace
  - DL\_Dxf, 73
- writeAppid
  - DL\_Dxf, 74
- writeArc
  - DL\_Dxf, 74
- writeBlockRecord
  - DL\_Dxf, 74
- writeCircle
  - DL\_Dxf, 74
- writeControlPoint
  - DL\_Dxf, 75
- writeDimAligned
  - DL\_Dxf, 75
- writeDimAngular2L
  - DL\_Dxf, 76
- writeDimAngular3P
  - DL\_Dxf, 76
- writeDimDiametric
  - DL\_Dxf, 77
- writeDimLinear
  - DL\_Dxf, 77
- writeDimOrdinate
  - DL\_Dxf, 78
- writeDimRadial
  - DL\_Dxf, 78
- writeDimStyle
  - DL\_Dxf, 79
- writeEllipse
  - DL\_Dxf, 79
- writeEndBlock
  - DL\_Dxf, 79
- writeFitPoint
  - DL\_Dxf, 80
- writeHatch1
  - DL\_Dxf, 80
- writeHatch2
  - DL\_Dxf, 80
- writeHatchEdge
  - DL\_Dxf, 81
- writeHatchLoop1
  - DL\_Dxf, 81
- writeHatchLoop2
  - DL\_Dxf, 82
- writeImage
  - DL\_Dxf, 82
- writeInsert
  - DL\_Dxf, 82
- writeKnot
  - DL\_Dxf, 84
- writeLayer
  - DL\_Dxf, 84
- writeLeader
  - DL\_Dxf, 85
- writeLeaderVertex
  - DL\_Dxf, 85
- writeLine

- DL\_Dxf, [85](#)
- writeLinetype
  - DL\_Dxf, [86](#)
- writeMText
  - DL\_Dxf, [86](#)
- writeObjects
  - DL\_Dxf, [87](#)
- writeObjectsEnd
  - DL\_Dxf, [87](#)
- writePoint
  - DL\_Dxf, [87](#)
- writePolyline
  - DL\_Dxf, [88](#)
- writePolylineEnd
  - DL\_Dxf, [88](#)
- writeRay
  - DL\_Dxf, [88](#)
- writeSolid
  - DL\_Dxf, [89](#)
- writeSpline
  - DL\_Dxf, [89](#)
- writeStyle
  - DL\_Dxf, [90](#)
- writeText
  - DL\_Dxf, [90](#)
- writeTrace
  - DL\_Dxf, [90](#)
- writeUcs
  - DL\_Dxf, [92](#)
- writeVertex
  - DL\_Dxf, [92](#)
- writeView
  - DL\_Dxf, [92](#)
- writeVPort
  - DL\_Dxf, [93](#)
- writeXLine
  - DL\_Dxf, [93](#)
- x
  - DL\_ControlPointData, [28](#)
  - DL\_FitPointData, [99](#)
  - DL\_LeaderVertexData, [125](#)
  - DL\_PointData, [135](#)
  - DL\_TraceData, [148](#)
  - DL\_VertexData, [150](#)
- x1
  - DL\_HatchEdgeData, [108](#)
  - DL\_LineData, [127](#)
- x2
  - DL\_HatchEdgeData, [108](#)
  - DL\_LineData, [127](#)
- xScaleFactor
  - DL\_ArcAlignedTextData, [15](#)
  - DL\_TextData, [147](#)
- xtype
  - DL\_DimOrdinateData, [60](#)
- y
  - DL\_ControlPointData, [28](#)
  - DL\_FitPointData, [99](#)
  - DL\_LeaderVertexData, [125](#)
  - DL\_PointData, [135](#)
  - DL\_VertexData, [150](#)
- y1
  - DL\_HatchEdgeData, [108](#)
  - DL\_LineData, [127](#)
- y2
  - DL\_HatchEdgeData, [109](#)
  - DL\_LineData, [127](#)
- z
  - DL\_ControlPointData, [28](#)
  - DL\_FitPointData, [100](#)
  - DL\_LeaderVertexData, [126](#)
  - DL\_PointData, [135](#)
  - DL\_VertexData, [150](#)
- z1
  - DL\_LineData, [128](#)
- z2
  - DL\_LineData, [128](#)