

dxflib

Generated by Doxygen 1.8.5

Sun Apr 3 2022 12:48:18

Contents

1	Todo List	1
2	Hierarchical Index	3
2.1	Class Hierarchy	3
3	Class Index	5
3.1	Class List	5
4	Class Documentation	9
4.1	DL_ArcAlignedTextData Struct Reference	9
4.1.1	Detailed Description	9
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	10
4.1.2.6	cy	10
4.1.2.7	cz	10
4.1.2.8	direction	10
4.1.2.9	endAngle	10
4.1.2.10	font	11
4.1.2.11	height	11
4.1.2.12	italic	11
4.1.2.13	leftOffset	11
4.1.2.14	offset	11
4.1.2.15	pitch	11
4.1.2.16	radius	11
4.1.2.17	reversedCharacterOrder	11
4.1.2.18	rightOffset	11
4.1.2.19	shxFont	12
4.1.2.20	side	12

4.1.2.21	spacing	12
4.1.2.22	startAngle	12
4.1.2.23	style	12
4.1.2.24	text	12
4.1.2.25	underline	12
4.1.2.26	wizard	12
4.1.2.27	xScaleFactor	12
4.2	DL_ArcData Struct Reference	13
4.2.1	Detailed Description	13
4.2.2	Constructor & Destructor Documentation	13
4.2.2.1	DL_ArcData	13
4.2.3	Member Data Documentation	13
4.2.3.1	angle1	13
4.2.3.2	angle2	13
4.2.3.3	cx	13
4.2.3.4	cy	14
4.2.3.5	cz	14
4.2.3.6	radius	14
4.3	DL_AttributeData Struct Reference	14
4.3.1	Detailed Description	14
4.3.2	Constructor & Destructor Documentation	15
4.3.2.1	DL_AttributeData	15
4.3.3	Member Data Documentation	15
4.3.3.1	tag	15
4.4	DL_Attributes Class Reference	15
4.4.1	Detailed Description	16
4.4.2	Constructor & Destructor Documentation	16
4.4.2.1	DL_Attributes	16
4.4.2.2	DL_Attributes	16
4.4.3	Member Function Documentation	16
4.4.3.1	getColor	16
4.4.3.2	getColor24	17
4.4.3.3	getLayer	17
4.4.3.4	getLinetype	17
4.4.3.5	getWidth	17
4.4.3.6	setColor	17
4.4.3.7	setColor24	17
4.4.3.8	setLayer	18
4.4.3.9	setLinetype	18
4.5	DL_BlockData Struct Reference	18

4.5.1	Detailed Description	18
4.5.2	Constructor & Destructor Documentation	18
4.5.2.1	DL_BlockData	18
4.5.3	Member Data Documentation	19
4.5.3.1	bpx	19
4.5.3.2	bpy	19
4.5.3.3	bpz	19
4.5.3.4	flags	19
4.5.3.5	name	19
4.6	DL_CircleData Struct Reference	19
4.6.1	Detailed Description	20
4.6.2	Constructor & Destructor Documentation	20
4.6.2.1	DL_CircleData	20
4.6.3	Member Data Documentation	20
4.6.3.1	cx	20
4.6.3.2	cy	20
4.6.3.3	cz	20
4.6.3.4	radius	20
4.7	DL_Codes Class Reference	20
4.7.1	Detailed Description	21
4.8	DL_ControlPointData Struct Reference	21
4.8.1	Detailed Description	21
4.8.2	Constructor & Destructor Documentation	21
4.8.2.1	DL_ControlPointData	21
4.8.3	Member Data Documentation	21
4.8.3.1	w	21
4.8.3.2	x	22
4.8.3.3	y	22
4.8.3.4	z	22
4.9	DL_CreationAdapter Class Reference	22
4.9.1	Detailed Description	25
4.9.2	Member Function Documentation	25
4.9.2.1	addArcAlignedText	25
4.9.2.2	addAttribute	25
4.9.2.3	addBlock	25
4.9.2.4	addInsert	25
4.9.2.5	addMText	25
4.9.2.6	addMTextChunk	26
4.9.2.7	addText	26
4.9.2.8	processCodeValuePair	26

4.9.2.9	setVariableDouble	26
4.9.2.10	setVariableInt	26
4.9.2.11	setVariableString	26
4.9.2.12	setVariableVector	26
4.10	DL_CreationInterface Class Reference	27
4.10.1	Detailed Description	30
4.10.2	Member Function Documentation	30
4.10.2.1	addArcAlignedText	30
4.10.2.2	addAttribute	30
4.10.2.3	addBlock	30
4.10.2.4	addInsert	30
4.10.2.5	addMText	30
4.10.2.6	addMTextChunk	31
4.10.2.7	addText	31
4.10.2.8	getAttributes	31
4.10.2.9	getExtrusion	31
4.10.2.10	processCodeValuePair	31
4.10.2.11	setAttributes	31
4.10.2.12	setExtrusion	31
4.10.2.13	setVariableDouble	31
4.10.2.14	setVariableInt	32
4.10.2.15	setVariableString	32
4.10.2.16	setVariableVector	32
4.11	DL_DictionaryData Struct Reference	32
4.11.1	Detailed Description	33
4.12	DL_DictionaryEntryData Struct Reference	33
4.12.1	Detailed Description	33
4.13	DL_DimAlignedData Struct Reference	33
4.13.1	Detailed Description	34
4.13.2	Constructor & Destructor Documentation	34
4.13.2.1	DL_DimAlignedData	34
4.13.3	Member Data Documentation	34
4.13.3.1	epx1	34
4.13.3.2	epx2	34
4.13.3.3	epy1	34
4.13.3.4	epy2	34
4.13.3.5	epz1	34
4.13.3.6	epz2	34
4.14	DL_DimAngular2LData Struct Reference	34
4.14.1	Detailed Description	35

4.14.2	Constructor & Destructor Documentation	35
4.14.2.1	DL_DimAngular2LData	35
4.14.3	Member Data Documentation	35
4.14.3.1	dpx1	35
4.14.3.2	dpx2	35
4.14.3.3	dpx3	36
4.14.3.4	dpx4	36
4.14.3.5	dpy1	36
4.14.3.6	dpy2	36
4.14.3.7	dpy3	36
4.14.3.8	dpy4	36
4.14.3.9	dpz1	36
4.14.3.10	dpz2	36
4.14.3.11	dpz3	36
4.14.3.12	dpz4	36
4.15	DL_DimAngular3PData Struct Reference	37
4.15.1	Detailed Description	37
4.15.2	Constructor & Destructor Documentation	37
4.15.2.1	DL_DimAngular3PData	37
4.15.3	Member Data Documentation	37
4.15.3.1	dpx1	37
4.15.3.2	dpx2	37
4.15.3.3	dpx3	38
4.15.3.4	dpy1	38
4.15.3.5	dpy2	38
4.15.3.6	dpy3	38
4.15.3.7	dpz1	38
4.15.3.8	dpz2	38
4.15.3.9	dpz3	38
4.16	DL_DimDiametricData Struct Reference	38
4.16.1	Detailed Description	39
4.16.2	Constructor & Destructor Documentation	39
4.16.2.1	DL_DimDiametricData	39
4.16.3	Member Data Documentation	39
4.16.3.1	dpx	39
4.16.3.2	dpy	39
4.16.3.3	dpz	39
4.16.3.4	leader	39
4.17	DL_DimensionData Struct Reference	39
4.17.1	Detailed Description	40

4.17.2	Constructor & Destructor Documentation	40
4.17.2.1	DL_DimensionData	40
4.17.3	Member Data Documentation	41
4.17.3.1	attachmentPoint	41
4.17.3.2	dpx	41
4.17.3.3	dpy	41
4.17.3.4	dpz	41
4.17.3.5	lineSpacingFactor	41
4.17.3.6	lineSpacingStyle	41
4.17.3.7	mpx	41
4.17.3.8	mpy	42
4.17.3.9	mpz	42
4.17.3.10	style	42
4.17.3.11	text	42
4.17.3.12	type	42
4.18	DL_DimLinearData Struct Reference	42
4.18.1	Detailed Description	43
4.18.2	Constructor & Destructor Documentation	43
4.18.2.1	DL_DimLinearData	43
4.18.3	Member Data Documentation	43
4.18.3.1	angle	43
4.18.3.2	dpx1	43
4.18.3.3	dpx2	43
4.18.3.4	dpy1	43
4.18.3.5	dpy2	43
4.18.3.6	dpz1	44
4.18.3.7	dpz2	44
4.18.3.8	oblique	44
4.19	DL_DimOrdinateData Struct Reference	44
4.19.1	Detailed Description	44
4.19.2	Constructor & Destructor Documentation	44
4.19.2.1	DL_DimOrdinateData	44
4.19.3	Member Data Documentation	45
4.19.3.1	dpx1	45
4.19.3.2	dpx2	45
4.19.3.3	dpy1	45
4.19.3.4	dpy2	45
4.19.3.5	dpz1	45
4.19.3.6	dpz2	45
4.19.3.7	xtype	45

4.20 DL_DimRadialData Struct Reference	45
4.20.1 Detailed Description	46
4.20.2 Constructor & Destructor Documentation	46
4.20.2.1 DL_DimRadialData	46
4.20.3 Member Data Documentation	46
4.20.3.1 dpx	46
4.20.3.2 dpy	46
4.20.3.3 dpz	46
4.20.3.4 leader	46
4.21 DL_Dxf Class Reference	46
4.21.1 Detailed Description	52
4.21.2 Member Function Documentation	52
4.21.2.1 addAttribute	52
4.21.2.2 addSolid	52
4.21.2.3 addTrace	53
4.21.2.4 checkVariable	53
4.21.2.5 getDimData	53
4.21.2.6 getLibVersion	53
4.21.2.7 getStrippedLine	53
4.21.2.8 in	54
4.21.2.9 in	54
4.21.2.10 out	54
4.21.2.11 processDXFGroup	55
4.21.2.12 readDxfGroups	56
4.21.2.13 stripWhiteSpace	56
4.21.2.14 test	57
4.21.2.15 write3dFace	57
4.21.2.16 writeAppid	57
4.21.2.17 writeArc	57
4.21.2.18 writeBlockRecord	57
4.21.2.19 writeCircle	58
4.21.2.20 writeControlPoint	59
4.21.2.21 writeDimAligned	59
4.21.2.22 writeDimAngular2L	59
4.21.2.23 writeDimAngular3P	60
4.21.2.24 writeDimDiametric	61
4.21.2.25 writeDimLinear	61
4.21.2.26 writeDimOrdinate	61
4.21.2.27 writeDimRadial	62
4.21.2.28 writeDimStyle	62

4.21.2.29 writeEllipse	62
4.21.2.30 writeEndBlock	62
4.21.2.31 writeFitPoint	63
4.21.2.32 writeHatch1	63
4.21.2.33 writeHatch2	63
4.21.2.34 writeHatchEdge	63
4.21.2.35 writeHatchLoop1	64
4.21.2.36 writeHatchLoop2	64
4.21.2.37 writeImage	64
4.21.2.38 writeInsert	64
4.21.2.39 writeKnot	65
4.21.2.40 writeLayer	66
4.21.2.41 writeLeader	66
4.21.2.42 writeLeaderVertex	66
4.21.2.43 writeLine	66
4.21.2.44 writeLinetype	67
4.21.2.45 writeMText	67
4.21.2.46 writeObjects	67
4.21.2.47 writeObjectsEnd	67
4.21.2.48 writePoint	67
4.21.2.49 writePolyline	68
4.21.2.50 writePolylineEnd	68
4.21.2.51 writeRay	68
4.21.2.52 writeSolid	68
4.21.2.53 writeSpline	69
4.21.2.54 writeStyle	69
4.21.2.55 writeText	69
4.21.2.56 writeTrace	69
4.21.2.57 writeUcs	70
4.21.2.58 writeVertex	70
4.21.2.59 writeView	70
4.21.2.60 writeVPort	70
4.21.2.61 writeXLine	70
4.22 DL_EllipseData Struct Reference	71
4.22.1 Detailed Description	71
4.22.2 Constructor & Destructor Documentation	71
4.22.2.1 DL_EllipseData	71
4.22.3 Member Data Documentation	72
4.22.3.1 angle1	72
4.22.3.2 angle2	72

4.22.3.3	cx	72
4.22.3.4	cy	72
4.22.3.5	cz	72
4.22.3.6	mx	72
4.22.3.7	my	72
4.22.3.8	mz	72
4.22.3.9	ratio	72
4.23	DL_Exception Class Reference	73
4.23.1	Detailed Description	73
4.24	DL_Extrusion Class Reference	73
4.24.1	Detailed Description	74
4.24.2	Constructor & Destructor Documentation	74
4.24.2.1	DL_Extrusion	74
4.24.3	Member Function Documentation	74
4.24.3.1	getDirection	74
4.24.3.2	getDirection	74
4.24.3.3	getElevation	74
4.25	DL_FitPointData Struct Reference	74
4.25.1	Detailed Description	75
4.25.2	Constructor & Destructor Documentation	75
4.25.2.1	DL_FitPointData	75
4.25.3	Member Data Documentation	75
4.25.3.1	x	75
4.25.3.2	y	75
4.25.3.3	z	75
4.26	DL_GroupCodeExc Class Reference	75
4.26.1	Detailed Description	76
4.27	DL_HatchData Struct Reference	76
4.27.1	Detailed Description	76
4.27.2	Constructor & Destructor Documentation	76
4.27.2.1	DL_HatchData	76
4.27.3	Member Data Documentation	76
4.27.3.1	angle	76
4.27.3.2	numLoops	77
4.27.3.3	originX	77
4.27.3.4	pattern	77
4.27.3.5	scale	77
4.27.3.6	solid	77
4.28	DL_HatchEdgeData Struct Reference	77
4.28.1	Detailed Description	78

4.28.2	Constructor & Destructor Documentation	78
4.28.2.1	DL_HatchEdgeData	78
4.28.2.2	DL_HatchEdgeData	79
4.28.2.3	DL_HatchEdgeData	79
4.28.2.4	DL_HatchEdgeData	79
4.28.3	Member Data Documentation	79
4.28.3.1	angle1	79
4.28.3.2	angle2	79
4.28.3.3	ccw	79
4.28.3.4	cx	79
4.28.3.5	cy	79
4.28.3.6	degree	80
4.28.3.7	mx	80
4.28.3.8	my	80
4.28.3.9	nControl	80
4.28.3.10	nFit	80
4.28.3.11	nKnots	80
4.28.3.12	radius	80
4.28.3.13	ratio	80
4.28.3.14	type	80
4.28.3.15	x1	81
4.28.3.16	x2	81
4.28.3.17	y1	81
4.28.3.18	y2	81
4.29	DL_HatchLoopData Struct Reference	81
4.29.1	Detailed Description	81
4.29.2	Constructor & Destructor Documentation	82
4.29.2.1	DL_HatchLoopData	82
4.29.3	Member Data Documentation	82
4.29.3.1	numEdges	82
4.30	DL_ImageData Struct Reference	82
4.30.1	Detailed Description	82
4.30.2	Constructor & Destructor Documentation	83
4.30.2.1	DL_ImageData	83
4.30.3	Member Data Documentation	83
4.30.3.1	brightness	83
4.30.3.2	contrast	83
4.30.3.3	fade	83
4.30.3.4	height	83
4.30.3.5	ipx	83

4.30.3.6	ipy	83
4.30.3.7	ipz	83
4.30.3.8	ref	84
4.30.3.9	ux	84
4.30.3.10	uy	84
4.30.3.11	uz	84
4.30.3.12	vx	84
4.30.3.13	vy	84
4.30.3.14	vz	84
4.30.3.15	width	84
4.31	DL_ImageDefData Struct Reference	84
4.31.1	Detailed Description	85
4.31.2	Constructor & Destructor Documentation	85
4.31.2.1	DL_ImageDefData	85
4.31.3	Member Data Documentation	85
4.31.3.1	file	85
4.31.3.2	ref	85
4.32	DL_InsertData Struct Reference	85
4.32.1	Detailed Description	86
4.32.2	Constructor & Destructor Documentation	86
4.32.2.1	DL_InsertData	86
4.32.3	Member Data Documentation	86
4.32.3.1	angle	86
4.32.3.2	cols	86
4.32.3.3	colSp	86
4.32.3.4	ipx	86
4.32.3.5	ipy	87
4.32.3.6	ipz	87
4.32.3.7	name	87
4.32.3.8	rows	87
4.32.3.9	rowSp	87
4.32.3.10	sx	87
4.32.3.11	sy	87
4.32.3.12	sz	87
4.33	DL_KnotData Struct Reference	87
4.33.1	Detailed Description	88
4.33.2	Constructor & Destructor Documentation	88
4.33.2.1	DL_KnotData	88
4.33.3	Member Data Documentation	88
4.33.3.1	k	88

4.34 DL_LayerData Struct Reference	88
4.34.1 Detailed Description	89
4.34.2 Constructor & Destructor Documentation	89
4.34.2.1 DL_LayerData	89
4.34.3 Member Data Documentation	89
4.34.3.1 flags	89
4.34.3.2 name	89
4.35 DL_LeaderData Struct Reference	89
4.35.1 Detailed Description	90
4.35.2 Constructor & Destructor Documentation	90
4.35.2.1 DL_LeaderData	90
4.35.3 Member Data Documentation	90
4.35.3.1 arrowHeadFlag	90
4.35.3.2 dimScale	90
4.35.3.3 hooklineDirectionFlag	90
4.35.3.4 hooklineFlag	90
4.35.3.5 leaderCreationFlag	90
4.35.3.6 leaderPathType	90
4.35.3.7 number	90
4.35.3.8 textAnnotationHeight	91
4.35.3.9 textAnnotationWidth	91
4.36 DL_LeaderVertexData Struct Reference	91
4.36.1 Detailed Description	91
4.36.2 Constructor & Destructor Documentation	91
4.36.2.1 DL_LeaderVertexData	91
4.36.3 Member Data Documentation	91
4.36.3.1 x	91
4.36.3.2 y	92
4.36.3.3 z	92
4.37 DL_LineData Struct Reference	92
4.37.1 Detailed Description	92
4.37.2 Constructor & Destructor Documentation	92
4.37.2.1 DL_LineData	92
4.37.3 Member Data Documentation	92
4.37.3.1 x1	92
4.37.3.2 x2	93
4.37.3.3 y1	93
4.37.3.4 y2	93
4.37.3.5 z1	93
4.37.3.6 z2	93

4.38 DL_LinetypeData Struct Reference	93
4.38.1 Detailed Description	94
4.38.2 Constructor & Destructor Documentation	94
4.38.2.1 DL_LinetypeData	94
4.39 DL_MTextData Struct Reference	94
4.39.1 Detailed Description	95
4.39.2 Constructor & Destructor Documentation	95
4.39.2.1 DL_MTextData	95
4.39.3 Member Data Documentation	95
4.39.3.1 angle	95
4.39.3.2 attachmentPoint	95
4.39.3.3 dirx	95
4.39.3.4 diry	95
4.39.3.5 dirz	95
4.39.3.6 drawingDirection	96
4.39.3.7 height	96
4.39.3.8 ipx	96
4.39.3.9 ipy	96
4.39.3.10 ipz	96
4.39.3.11 lineSpacingFactor	96
4.39.3.12 lineSpacingStyle	96
4.39.3.13 style	96
4.39.3.14 text	96
4.39.3.15 width	97
4.40 DL_NullStrExc Class Reference	97
4.40.1 Detailed Description	97
4.41 DL_PointData Struct Reference	97
4.41.1 Detailed Description	97
4.41.2 Constructor & Destructor Documentation	98
4.41.2.1 DL_PointData	98
4.41.3 Member Data Documentation	98
4.41.3.1 x	98
4.41.3.2 y	98
4.41.3.3 z	98
4.42 DL_PolylineData Struct Reference	98
4.42.1 Detailed Description	98
4.42.2 Constructor & Destructor Documentation	99
4.42.2.1 DL_PolylineData	99
4.42.3 Member Data Documentation	99
4.42.3.1 elevation	99

4.42.3.2	flags	99
4.42.3.3	m	99
4.42.3.4	n	99
4.42.3.5	number	99
4.43	DL_RayData Struct Reference	99
4.43.1	Detailed Description	100
4.43.2	Constructor & Destructor Documentation	100
4.43.2.1	DL_RayData	100
4.43.3	Member Data Documentation	100
4.43.3.1	bx	100
4.43.3.2	by	100
4.43.3.3	bz	100
4.43.3.4	dx	100
4.43.3.5	dy	100
4.43.3.6	dz	101
4.44	DL_SplineData Struct Reference	101
4.44.1	Detailed Description	101
4.44.2	Constructor & Destructor Documentation	101
4.44.2.1	DL_SplineData	101
4.44.3	Member Data Documentation	101
4.44.3.1	degree	101
4.44.3.2	flags	102
4.44.3.3	nControl	102
4.44.3.4	nFit	102
4.44.3.5	nKnots	102
4.45	DL_StyleData Struct Reference	102
4.45.1	Detailed Description	103
4.45.2	Member Data Documentation	103
4.45.2.1	fixedTextHeight	103
4.46	DL_TextData Struct Reference	103
4.46.1	Detailed Description	104
4.46.2	Constructor & Destructor Documentation	104
4.46.2.1	DL_TextData	104
4.46.3	Member Data Documentation	104
4.46.3.1	angle	104
4.46.3.2	apx	104
4.46.3.3	apy	104
4.46.3.4	apz	104
4.46.3.5	height	105
4.46.3.6	hJustification	105

4.46.3.7	ipx	105
4.46.3.8	ipy	105
4.46.3.9	ipz	105
4.46.3.10	style	105
4.46.3.11	text	105
4.46.3.12	textGenerationFlags	105
4.46.3.13	vJustification	105
4.46.3.14	xScaleFactor	106
4.47	DL_TraceData Struct Reference	106
4.47.1	Detailed Description	106
4.47.2	Constructor & Destructor Documentation	106
4.47.2.1	DL_TraceData	106
4.47.3	Member Data Documentation	106
4.47.3.1	thickness	106
4.47.3.2	x	107
4.48	DL_VertexData Struct Reference	107
4.48.1	Detailed Description	107
4.48.2	Constructor & Destructor Documentation	107
4.48.2.1	DL_VertexData	107
4.48.3	Member Data Documentation	107
4.48.3.1	bulge	107
4.48.3.2	x	107
4.48.3.3	y	108
4.48.3.4	z	108
4.49	DL_Writer Class Reference	108
4.49.1	Detailed Description	110
4.49.2	Constructor & Destructor Documentation	110
4.49.2.1	DL_Writer	110
4.49.3	Member Function Documentation	110
4.49.3.1	comment	110
4.49.3.2	dxfBool	110
4.49.3.3	dxfEOF	110
4.49.3.4	dxfHex	111
4.49.3.5	dxflnt	112
4.49.3.6	dxfReal	112
4.49.3.7	dxfString	112
4.49.3.8	dxString	112
4.49.3.9	entity	112
4.49.3.10	entityAttributes	113
4.49.3.11	getNextHandle	113

4.49.3.12 section	113
4.49.3.13 sectionBlockEntry	113
4.49.3.14 sectionBlockEntryEnd	114
4.49.3.15 sectionBlocks	114
4.49.3.16 sectionClasses	114
4.49.3.17 sectionEnd	114
4.49.3.18 sectionEntities	114
4.49.3.19 sectionHeader	114
4.49.3.20 sectionObjects	115
4.49.3.21 sectionTables	115
4.49.3.22 table	115
4.49.3.23 tableAppid	115
4.49.3.24 tableAppidEntry	115
4.49.3.25 tableEnd	116
4.49.3.26 tableLayerEntry	116
4.49.3.27 tableLayers	116
4.49.3.28 tableLinetypeEntry	116
4.49.3.29 tableLinetypes	116
4.49.3.30 tableStyle	117
4.50 DL_WriterA Class Reference	118
4.50.1 Detailed Description	119
4.50.2 Member Function Documentation	119
4.50.2.1 dxflHex	119
4.50.2.2 dxflInt	119
4.50.2.3 dxflReal	119
4.50.2.4 dxflString	120
4.50.2.5 dxflString	120
4.50.2.6 openFailed	120
4.51 DL_XLineData Struct Reference	120
4.51.1 Detailed Description	121
4.51.2 Constructor & Destructor Documentation	121
4.51.2.1 DL_XLineData	121
4.51.3 Member Data Documentation	121
4.51.3.1 bx	121
4.51.3.2 by	121
4.51.3.3 bz	121
4.51.3.4 dx	121
4.51.3.5 dy	122
4.51.3.6 dz	122

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	13
DL_Attributes	15
DL_BlockData	18
DL_CircleData	19
DL_Codes	20
DL_ControlPointData	21
DL_CreationInterface	27
DL_CreationAdapter	22
DL_DictionaryData	32
DL_DictionaryEntryData	33
DL_DimAlignedData	33
DL_DimAngular2LData	34
DL_DimAngular3PData	37
DL_DimDiametricData	38
DL_DimensionData	39
DL_DimLinearData	42
DL_DimOrdinateData	44
DL_DimRadialData	45
DL_Dxf	46
DL_EllipseData	71
DL_Exception	73
DL_GroupCodeExc	75
DL_NullStrExc	97
DL_Extrusion	73
DL_FitPointData	74
DL_HatchData	76
DL_HatchEdgeData	77
DL_HatchLoopData	81
DL_ImageData	82
DL_ImageDefData	84
DL_InsertData	85
DL_KnotData	87
DL_LayerData	88
DL_LeaderData	89
DL_LeaderVertexData	91
DL_LineData	92

DL_LinetypeData	93
DL_MTextData	94
DL_PointData	97
DL_PolylineData	98
DL_RayData	99
DL_SplineData	101
DL_StyleData	102
DL_TextData	103
DL_AttributeData	14
DL_TraceData	106
DL_VertexData	107
DL_Writer	108
DL_WriterA	118
DL_XLineData	120

Chapter 3

Class Index

3.1 Class List

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

DL_ArcAlignedTextData	
Arc Aligned Text Data	9
DL_ArcData	
Arc Data	13
DL_AttributeData	
Block attribute data	14
DL_Attributes	
Storing and passing around attributes	15
DL_BlockData	
Block Data	18
DL_CircleData	
Circle Data	19
DL_Codes	
Codes for colors and DXF versions	20
DL_ControlPointData	
Spline control point data	21
DL_CreationAdapter	
An abstract adapter class for receiving DXF events when a DXF file is being read	22
DL_CreationInterface	
Abstract class (interface) for the creation of new entities	27
DL_DictionaryData	
Dictionary data	32
DL_DictionaryEntryData	
Dictionary entry data	33
DL_DimAlignedData	
Aligned Dimension Data	33
DL_DimAngular2LData	
Angular Dimension Data	34
DL_DimAngular3PData	
Angular Dimension Data (3 points version)	37
DL_DimDiametricData	
Diametric Dimension Data	38
DL_DimensionData	
Generic Dimension Data	39
DL_DimLinearData	
Linear (rotated) Dimension Data	42
DL_DimOrdinateData	
Ordinate Dimension Data	44

DL_DimRadialData	
Radial Dimension Data	45
DL_Dxf	
Reading and writing of DXF files	46
DL_EllipseData	
Ellipse Data	71
DL_Exception	
Used for exception handling	73
DL_Extrusion	
Extrusion direction	73
DL_FitPointData	
Spline fit point data	74
DL_GroupCodeExc	
Used for exception handling	75
DL_HatchData	
Hatch data	76
DL_HatchEdgeData	
Hatch edge data	77
DL_HatchLoopData	
Hatch boundary path (loop) data	81
DL_ImageData	
Image Data	82
DL_ImageDefData	
Image Definition Data	84
DL_InsertData	
Insert Data	85
DL_KnotData	
Spline knot data	87
DL_LayerData	
Layer Data	88
DL_LeaderData	
Leader (arrow)	89
DL_LeaderVertexData	
Leader Vertex Data	91
DL_LineData	
Line Data	92
DL_LinetypeData	
Line Type Data	93
DL_MTextData	
MText Data	94
DL_NullStrExc	
Used for exception handling	97
DL_PointData	
Point Data	97
DL_PolylineData	
Polyline Data	98
DL_RayData	
Ray Data	99
DL_SplineData	
Spline Data	101
DL_StyleData	
Text style data	102
DL_TextData	
Text Data	103
DL_TraceData	
Trace Data / solid data / 3d face data	106
DL_VertexData	
Vertex Data	107

DL_Writer	
Defines interface for writing low level DXF constructs to a file	108
DL_WriterA	
Implements functions defined in DL_Writer for writing low level DXF constructs to an ASCII format DXF file	118
DL_XLineData	
XLine Data	120

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 int DL_ArcAlignedTextData::alignment

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

Referenced by DL_Dxf::addArcAlignedText().

4.1.2.2 int DL_ArcAlignedTextData::arcHandle

Arc handle/ID

Referenced by DL_Dxf::addArcAlignedText().

4.1.2.3 bool DL_ArcAlignedTextData::bold

Bold flag

Referenced by DL_Dxf::addArcAlignedText().

4.1.2.4 int DL_ArcAlignedTextData::characterSet

Character set value. Windows character set identifier.

Referenced by DL_Dxf::addArcAlignedText().

4.1.2.5 double DL_ArcAlignedTextData::cx

X coordinate of arc center point.

Referenced by DL_Dxf::addArcAlignedText().

4.1.2.6 double DL_ArcAlignedTextData::cy

Y coordinate of arc center point.

Referenced by DL_Dxf::addArcAlignedText().

4.1.2.7 double DL_ArcAlignedTextData::cz

Z coordinate of arc center point.

Referenced by DL_Dxf::addArcAlignedText().

4.1.2.8 int DL_ArcAlignedTextData::direction

Direction 1: outward from center 2: inward from center

Referenced by DL_Dxf::addArcAlignedText().

4.1.2.9 double DL_ArcAlignedTextData::endAngle

End angle (radians)

Referenced by DL_Dxf::addArcAlignedText().

4.1.2.10 std::string DL_ArcAlignedTextData::font

Font name

Referenced by DL_Dxf::addArcAlignedText().

4.1.2.11 double DL_ArcAlignedTextData::height

Text height

Referenced by DL_Dxf::addArcAlignedText().

4.1.2.12 bool DL_ArcAlignedTextData::italic

Italic flag

Referenced by DL_Dxf::addArcAlignedText().

4.1.2.13 double DL_ArcAlignedTextData::leftOffset

Left offset

Referenced by DL_Dxf::addArcAlignedText().

4.1.2.14 double DL_ArcAlignedTextData::offset

Offset from arc

Referenced by DL_Dxf::addArcAlignedText().

4.1.2.15 int DL_ArcAlignedTextData::pitch

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

Referenced by DL_Dxf::addArcAlignedText().

4.1.2.16 double DL_ArcAlignedTextData::radius

Arc radius.

Referenced by DL_Dxf::addArcAlignedText().

4.1.2.17 bool DL_ArcAlignedTextData::reversedCharacterOrder

Reversed character order: false: normal true: reversed

Referenced by DL_Dxf::addArcAlignedText().

4.1.2.18 double DL_ArcAlignedTextData::rightOffset

Right offset

Referenced by DL_Dxf::addArcAlignedText().

4.1.2.19 `bool DL_ArcAlignedTextData::shxFont`

Font type: false: TTF true: SHX

Referenced by `DL_Dxf::addArcAlignedText()`.

4.1.2.20 `int DL_ArcAlignedTextData::side`

Side 1: convex 2: concave

Referenced by `DL_Dxf::addArcAlignedText()`.

4.1.2.21 `double DL_ArcAlignedTextData::spacing`

Character spacing

Referenced by `DL_Dxf::addArcAlignedText()`.

4.1.2.22 `double DL_ArcAlignedTextData::startAngle`

Start angle (radians)

Referenced by `DL_Dxf::addArcAlignedText()`.

4.1.2.23 `std::string DL_ArcAlignedTextData::style`

Style

Referenced by `DL_Dxf::addArcAlignedText()`.

4.1.2.24 `std::string DL_ArcAlignedTextData::text`

Text string

Referenced by `DL_Dxf::addArcAlignedText()`.

4.1.2.25 `bool DL_ArcAlignedTextData::underline`

Underline flag

Referenced by `DL_Dxf::addArcAlignedText()`.

4.1.2.26 `bool DL_ArcAlignedTextData::wizard`

Wizard flag

Referenced by `DL_Dxf::addArcAlignedText()`.

4.1.2.27 `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](#) (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 double DL_ArcData::angle1

Startangle of arc in degrees.

Referenced by [DL_Dxf::writeArc\(\)](#).

4.2.3.2 double DL_ArcData::angle2

Endangle of arc in degrees.

Referenced by [DL_Dxf::writeArc\(\)](#).

4.2.3.3 double DL_ArcData::cx

X Coordinate of center point.

Referenced by [DL_Dxf::writeArc\(\)](#).

4.2.3.4 double DL_ArcData::cy

Y Coordinate of center point.

Referenced by DL_Dxf::writeArc().

4.2.3.5 double DL_ArcData::cz

Z Coordinate of center point.

Referenced by DL_Dxf::writeArc().

4.2.3.6 double DL_ArcData::radius

Radius of arc.

Referenced by DL_Dxf::writeArc().

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

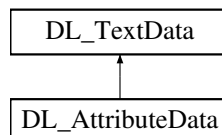
- src/dl_entities.h

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 (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 `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::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::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.
<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 int **DL_Attributes::getColor** () const `[inline]`

Returns

Color.

See Also

[DL_Codes](#), `dxflib::DxfColors`

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

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

Returns

Layer name.

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

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

Returns

Width.

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

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

Sets the color.

See Also

[DL_Codes](#), `dxfColors`

Referenced by `DL_Dxf::addLayer()`.

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

Sets the 24bit color.

See Also

[DL_Codes](#), `dxfColors`

4.4.3.8 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 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 (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 double DL_BlockData::bpx

X Coordinate of base point.

Referenced by DL_Dxf::writeBlock().

4.5.3.2 double DL_BlockData::bpy

Y Coordinate of base point.

Referenced by DL_Dxf::writeBlock().

4.5.3.3 double DL_BlockData::bpz

Z Coordinate of base point.

Referenced by DL_Dxf::writeBlock().

4.5.3.4 int DL_BlockData::flags

Block flags.

(not used currently)

4.5.3.5 std::string DL_BlockData::name

Block name.

Referenced by DL_Dxf::writeBlock().

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 (double *acx*, double *acy*, double *acz*, double *aRadius*) [inline]

Constructor.

Parameters: see member variables.

4.6.3 Member Data Documentation

4.6.3.1 double DL_CircleData::cx

X Coordinate of center point.

Referenced by DL_Dxf::writeCircle().

4.6.3.2 double DL_CircleData::cy

Y Coordinate of center point.

Referenced by DL_Dxf::writeCircle().

4.6.3.3 double DL_CircleData::cz

Z Coordinate of center point.

Referenced by DL_Dxf::writeCircle().

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

Constructor.

Parameters: see member variables.

4.8.3 Member Data Documentation

4.8.3.1 `double DL_ControlPointData::w`

Weight of control point.

4.8.3.2 double DL_ControlPointData::x

X coordinate of the control point.

Referenced by DL_Dxf::writeControlPoint().

4.8.3.3 double DL_ControlPointData::y

Y coordinate of the control point.

Referenced by DL_Dxf::writeControlPoint().

4.8.3.4 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:

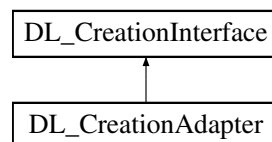
- src/dl_entities.h

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 virtual void [DL_CreationAdapter::addArcAlignedText](#) (const [DL_ArcAlignedTextData](#) & *data*) [inline],
[virtual]

Called for every arc aligned text entity.

Implements [DL_CreationInterface](#).

4.9.2.2 virtual void [DL_CreationAdapter::addAttribute](#) (const [DL_AttributeData](#) & *data*) [inline],[virtual]

Called for every block Attribute entity.

Implements [DL_CreationInterface](#).

4.9.2.3 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](#).

4.9.2.4 virtual void [DL_CreationAdapter::addInsert](#) (const [DL_InsertData](#) & *data*) [inline],[virtual]

Called for every insert.

Implements [DL_CreationInterface](#).

4.9.2.5 virtual void [DL_CreationAdapter::addMText](#) (const [DL_MTextData](#) & *data*) [inline],[virtual]

Called for every multi Text entity.

Implements [DL_CreationInterface](#).

4.9.2.6 `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](#).

4.9.2.7 `virtual void DL_CreationAdapter::addText (const DL_TextData & data) [inline],[virtual]`

Called for every text entity.

Implements [DL_CreationInterface](#).

4.9.2.8 `virtual void DL_CreationAdapter::processCodeValuePair (unsigned 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](#).

4.9.2.9 `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](#).

4.9.2.10 `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](#).

4.9.2.11 `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](#).

4.9.2.12 `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:

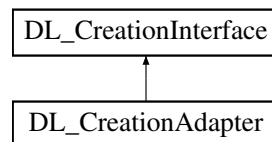
- src/dl_creationadapter.h

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 &appId)=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](#) ()
- 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 `virtual void DL_CreationInterface::addArcAlignedText (const DL_ArcAlignedTextData & data) [pure virtual]`

Called for every arc aligned text entity.

Implemented in [DL_CreationAdapter](#).

Referenced by `DL_Dxf::addArcAlignedText()`.

4.10.2.2 `virtual void DL_CreationInterface::addAttribute (const DL_AttributeData & data) [pure virtual]`

Called for every block Attribute entity.

Implemented in [DL_CreationAdapter](#).

Referenced by `DL_Dxf::addAttribute()`.

4.10.2.3 `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()`.

4.10.2.4 `virtual void DL_CreationInterface::addInsert (const DL_InsertData & data) [pure virtual]`

Called for every insert.

Implemented in [DL_CreationAdapter](#).

Referenced by `DL_Dxf::addInsert()`.

4.10.2.5 `virtual void DL_CreationInterface::addMText (const DL_MTextData & data) [pure virtual]`

Called for every multi Text entity.

Implemented in [DL_CreationAdapter](#).

Referenced by `DL_Dxf::addMText()`.

4.10.2.6 `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()`.

4.10.2.7 `virtual void DL_CreationInterface::addText (const DL_TextData & data) [pure virtual]`

Called for every text entity.

Implemented in [DL_CreationAdapter](#).

Referenced by `DL_Dxf::addText()`.

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

Returns

the current attributes used for new entities.

Referenced by `DL_Dxf::addLayer()`.

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

Returns

the current attributes used for new entities.

4.10.2.10 `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()`.

4.10.2.11 `void DL_CreationInterface::setAttributes (const DL_Attributes & attrib) [inline]`

Sets the current attributes for entities.

Referenced by `DL_Dxf::processDXFGroup()`.

4.10.2.12 `void DL_CreationInterface::setExtrusion (double dx, double dy, double dz, double elevation) [inline]`

Sets the current attributes for entities.

Referenced by `DL_Dxf::processDXFGroup()`.

4.10.2.13 `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()`.

4.10.2.14 `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()`.

4.10.2.15 `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()`.

4.10.2.16 `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 (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 `double DL_DimAlignedData::epx1`

X Coordinate of Extension point 1.

Referenced by `DL_Dxf::writeDimAligned()`.

4.13.3.2 `double DL_DimAlignedData::epx2`

X Coordinate of Extension point 2.

Referenced by `DL_Dxf::writeDimAligned()`.

4.13.3.3 `double DL_DimAlignedData::epy1`

Y Coordinate of Extension point 1.

Referenced by `DL_Dxf::writeDimAligned()`.

4.13.3.4 `double DL_DimAlignedData::epy2`

Y Coordinate of Extension point 2.

Referenced by `DL_Dxf::writeDimAligned()`.

4.13.3.5 `double DL_DimAlignedData::epz1`

Z Coordinate of Extension point 1.

4.13.3.6 `double DL_DimAlignedData::epz2`

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_x1, double ddp_y1, double ddp_z1, double ddp_x2, double ddp_y2, double ddp_z2, double ddp_x3, double ddp_y3, double ddp_z3, double ddp_x4, double ddp_y4, double ddp_z4)

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 (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 `double DL_DimAngular2LData::dpx1`

X Coordinate of definition point 1.

Referenced by `DL_Dxf::writeDimAngular2L()`.

4.14.3.2 `double DL_DimAngular2LData::dpx2`

X Coordinate of definition point 2.

Referenced by `DL_Dxf::writeDimAngular2L()`.

4.14.3.3 double DL_DimAngular2LData::dpx3

X Coordinate of definition point 3.

Referenced by DL_Dxf::writeDimAngular2L().

4.14.3.4 double DL_DimAngular2LData::dpx4

X Coordinate of definition point 4.

Referenced by DL_Dxf::writeDimAngular2L().

4.14.3.5 double DL_DimAngular2LData::dpy1

Y Coordinate of definition point 1.

Referenced by DL_Dxf::writeDimAngular2L().

4.14.3.6 double DL_DimAngular2LData::dpy2

Y Coordinate of definition point 2.

Referenced by DL_Dxf::writeDimAngular2L().

4.14.3.7 double DL_DimAngular2LData::dpy3

Y Coordinate of definition point 3.

Referenced by DL_Dxf::writeDimAngular2L().

4.14.3.8 double DL_DimAngular2LData::dpy4

Y Coordinate of definition point 4.

Referenced by DL_Dxf::writeDimAngular2L().

4.14.3.9 double DL_DimAngular2LData::dpz1

Z Coordinate of definition point 1.

4.14.3.10 double DL_DimAngular2LData::dpz2

Z Coordinate of definition point 2.

4.14.3.11 double DL_DimAngular2LData::dpz3

Z Coordinate of definition point 3.

4.14.3.12 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_x1, double ddp_y1, double ddp_z1, double ddp_x2, double ddp_y2, double ddp_z2, double ddp_x3, double ddp_y3, double ddp_z3)

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 (double ddpx1, double ddpy1, double ddpz1, double ddpx2, double ddpy2, double ddpz2, double ddpx3, double ddpy3, double ddpz3)` `[inline]`

Constructor.

Parameters: see member variables.

4.15.3 Member Data Documentation

4.15.3.1 `double DL_DimAngular3PData::dpx1`

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

Referenced by `DL_Dxf::writeDimAngular3P()`.

4.15.3.2 `double DL_DimAngular3PData::dpx2`

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

Referenced by `DL_Dxf::writeDimAngular3P()`.

4.15.3.3 double DL_DimAngular3PData::dpx3

X Coordinate of definition point 3 (center).

Referenced by DL_Dxf::writeDimAngular3P().

4.15.3.4 double DL_DimAngular3PData::dpy1

Y Coordinate of definition point 1.

Referenced by DL_Dxf::writeDimAngular3P().

4.15.3.5 double DL_DimAngular3PData::dpy2

Y Coordinate of definition point 2.

Referenced by DL_Dxf::writeDimAngular3P().

4.15.3.6 double DL_DimAngular3PData::dpy3

Y Coordinate of definition point 3.

Referenced by DL_Dxf::writeDimAngular3P().

4.15.3.7 double DL_DimAngular3PData::dpz1

Z Coordinate of definition point 1.

4.15.3.8 double DL_DimAngular3PData::dpz2

Z Coordinate of definition point 2.

4.15.3.9 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_x, double ddp_y, double ddp_z, 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 (double ddpx, double ddpy, double ddpz, double dleader)`
`[inline]`

Constructor.

Parameters: see member variables.

4.16.3 Member Data Documentation

4.16.3.1 `double DL_DimDiametricData::dpx`

X Coordinate of definition point (DXF 15).

Referenced by `DL_Dxf::writeDimDiametric()`.

4.16.3.2 `double DL_DimDiametricData::dpy`

Y Coordinate of definition point (DXF 25).

Referenced by `DL_Dxf::writeDimDiametric()`.

4.16.3.3 `double DL_DimDiametricData::dpz`

Z Coordinate of definition point (DXF 35).

4.16.3.4 `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 (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 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 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 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 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 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 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 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 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 double DL_DimensionData::mpz

Z Coordinate of middle point of the text.

4.17.3.10 std::string DL_DimensionData::style

Dimension style (font name).

4.17.3.11 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 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_{x1}, double ddp_{y1}, double ddp_{z1}, double ddp_{x2}, double ddp_{y2}, double ddp_{z2}, 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 (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 `double DL_DimLinearData::angle`

Rotation angle (angle of dimension line) in degrees.

Referenced by `DL_Dxf::writeDimLinear()`.

4.18.3.2 `double DL_DimLinearData::dpx1`

X Coordinate of Extension point 1.

Referenced by `DL_Dxf::writeDimLinear()`.

4.18.3.3 `double DL_DimLinearData::dpx2`

X Coordinate of Extension point 2.

Referenced by `DL_Dxf::writeDimLinear()`.

4.18.3.4 `double DL_DimLinearData::dpy1`

Y Coordinate of Extension point 1.

Referenced by `DL_Dxf::writeDimLinear()`.

4.18.3.5 `double DL_DimLinearData::dpy2`

Y Coordinate of Extension point 2.

Referenced by `DL_Dxf::writeDimLinear()`.

4.18.3.6 double DL_DimLinearData::dpz1

Z Coordinate of Extension point 1.

4.18.3.7 double DL_DimLinearData::dpz2

Z Coordinate of Extension point 2.

4.18.3.8 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_x1, double ddp_y1, double ddp_z1, double ddp_x2, double ddp_y2, double ddp_z2, bool dx_{type})

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 (double ddp_x1, double ddp_y1, double ddp_z1, double ddp_x2, double ddp_y2, double ddp_z2, bool dx_{type}) [inline]

Constructor.

Parameters: see member variables.

4.19.3 Member Data Documentation

4.19.3.1 double DL_DimOrdinateData::dpx1

X Coordinate of definition point 1.

Referenced by DL_Dxf::writeDimOrdinate().

4.19.3.2 double DL_DimOrdinateData::dpx2

X Coordinate of definition point 2.

Referenced by DL_Dxf::writeDimOrdinate().

4.19.3.3 double DL_DimOrdinateData::dpy1

Y Coordinate of definition point 1.

Referenced by DL_Dxf::writeDimOrdinate().

4.19.3.4 double DL_DimOrdinateData::dpy2

Y Coordinate of definition point 2.

Referenced by DL_Dxf::writeDimOrdinate().

4.19.3.5 double DL_DimOrdinateData::dpz1

Z Coordinate of definition point 1.

4.19.3.6 double DL_DimOrdinateData::dpz2

Z Coordinate of definition point 2.

4.19.3.7 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_x, double ddp_y, double ddp_z, 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 (double ddpx, double ddpy, double ddpz, double dleader)` `[inline]`

Constructor.

Parameters: see member variables.

4.20.3 Member Data Documentation

4.20.3.1 `double DL_DimRadialData::dpx`

X Coordinate of definition point.

Referenced by `DL_Dxf::writeDimRadial()`.

4.20.3.2 `double DL_DimRadialData::dpy`

Y Coordinate of definition point.

Referenced by `DL_Dxf::writeDimRadial()`.

4.20.3.3 `double DL_DimRadialData::dpz`

Z Coordinate of definition point.

4.20.3.4 `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)
- [void addPoint](#) ([DL_CreationInterface](#) *creationInterface)
Adds a point entity that was read from the file via the creation interface.
- [void addLine](#) ([DL_CreationInterface](#) *creationInterface)
Adds a line entity that was read from the file via the creation interface.
- [void addXLine](#) ([DL_CreationInterface](#) *creationInterface)
Adds an xline entity that was read from the file via the creation interface.
- [void addRay](#) ([DL_CreationInterface](#) *creationInterface)
Adds a ray entity that was read from the file via the creation interface.
- [void addPolyline](#) ([DL_CreationInterface](#) *creationInterface)
Adds a polyline entity that was read from the file via the creation interface.
- [void addVertex](#) ([DL_CreationInterface](#) *creationInterface)
Adds a polyline vertex entity that was read from the file via the creation interface.
- [void addSpline](#) ([DL_CreationInterface](#) *creationInterface)
Adds a spline entity that was read from the file via the creation interface.
- [void addArc](#) ([DL_CreationInterface](#) *creationInterface)
Adds an arc entity that was read from the file via the creation interface.
- [void addCircle](#) ([DL_CreationInterface](#) *creationInterface)
Adds a circle entity that was read from the file via the creation interface.
- [void addEllipse](#) ([DL_CreationInterface](#) *creationInterface)
Adds an ellipse entity that was read from the file via the creation interface.
- [void addInsert](#) ([DL_CreationInterface](#) *creationInterface)
Adds an insert entity that was read from the file via the creation interface.
- [void addTrace](#) ([DL_CreationInterface](#) *creationInterface)

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

 - void [add3dFace](#) ([DL_CreationInterface](#) *creationInterface)
- Adds a 3dface entity that was read from the file via the creation interface.*

 - void [addSolid](#) ([DL_CreationInterface](#) *creationInterface)
- Adds a solid entity (filled trace) that was read from the file via the creation interface.*

 - void [addMText](#) ([DL_CreationInterface](#) *creationInterface)
- Adds an MText entity that was read from the file via the creation interface.*

 - void [addText](#) ([DL_CreationInterface](#) *creationInterface)
- Adds an text entity that was read from the file via the creation interface.*

 - void [addArcAlignedText](#) ([DL_CreationInterface](#) *creationInterface)
- Adds an arc aligned text entity that was read from the file via the creation interface.*

 - void [addAttribute](#) ([DL_CreationInterface](#) *creationInterface)
- Adds an attrib entity that was read from the file via the creation interface.*

 - [DL_DimensionData](#) [getDimData](#) ()
- Adds a linear dimension entity that was read from the file via the creation interface.*

 - void [addDimLinear](#) ([DL_CreationInterface](#) *creationInterface)
- Adds an aligned dimension entity that was read from the file via the creation interface.*

 - void [addDimAligned](#) ([DL_CreationInterface](#) *creationInterface)
- Adds a radial dimension entity that was read from the file via the creation interface.*

 - void [addDimRadial](#) ([DL_CreationInterface](#) *creationInterface)
- Adds a diametric 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)
- Adds an angular dimension entity that was read from the file via the creation interface.*

 - void [addDimAngular3P](#) ([DL_CreationInterface](#) *creationInterface)
- Adds an ordinate dimension entity that was read from the file via the creation interface.*

 - void [addDimOrdinate](#) ([DL_CreationInterface](#) *creationInterface)
- Adds a leader entity that was read from the file via the creation interface.*

 - void [addLeader](#) ([DL_CreationInterface](#) *creationInterface)
- Adds a hatch 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](#) ()
- Adds a hatch entity that was read from the file via the creation interface.*

 - void [addHatchEdge](#) ()
- Adds a hatch entity that was read from the file via the creation interface.*

 - 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)
- Adds a dictionary entry that was read from the file via the creation interface.*

 - void [addDictionaryEntry](#) ([DL_CreationInterface](#) *creationInterface)
- Adds a dictionary entry that was read from the file via the creation interface.*

 - 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)
- void **[writeDimStyleOverrides](#)** ([DL_WriterA](#) &dw, const [DL_DimensionData](#) &data)
- void [writeDimAligned](#) ([DL_WriterA](#) &dw, const [DL_DimensionData](#) &data, const [DL_DimAlignedData](#) &edata, const [DL_Attributes](#) &attrib)
Writes an aligned dimension entity to the file.
- void [writeDimLinear](#) ([DL_WriterA](#) &dw, const [DL_DimensionData](#) &data, const [DL_DimLinearData](#) &edata, const [DL_Attributes](#) &attrib)
Writes a linear dimension entity to the file.
- void [writeDimRadial](#) ([DL_WriterA](#) &dw, const [DL_DimensionData](#) &data, const [DL_DimRadialData](#) &edata, const [DL_Attributes](#) &attrib)
Writes a radial dimension entity to the file.
- void [writeDimDiametric](#) ([DL_WriterA](#) &dw, const [DL_DimensionData](#) &data, const [DL_DimDiametricData](#) &edata, const [DL_Attributes](#) &attrib)
Writes a diametric 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)
- void [writeHatch1](#) ([DL_WriterA](#) &dw, const [DL_HatchData](#) &data, const [DL_Attributes](#) &attrib)
Writes the beginning of a hatch entity to the file.
- void [writeHatch2](#) ([DL_WriterA](#) &dw, const [DL_HatchData](#) &data, const [DL_Attributes](#) &attrib)
Writes the end of a hatch entity to the file.
- void [writeHatchLoop1](#) ([DL_WriterA](#) &dw, const [DL_HatchLoopData](#) &data)
Writes the beginning of a hatch loop to the file.
- void [writeHatchLoop2](#) ([DL_WriterA](#) &dw, const [DL_HatchLoopData](#) &data)
Writes the end of a hatch loop to the file.
- void [writeHatchEdge](#) ([DL_WriterA](#) &dw, const [DL_HatchEdgeData](#) &data)
Writes the beginning of a hatch entity to the file.
- unsigned long [writeImage](#) ([DL_WriterA](#) &dw, const [DL_ImageData](#) &data, const [DL_Attributes](#) &attrib)
Writes an image entity.
- void [writeImageDef](#) ([DL_WriterA](#) &dw, int handle, const [DL_ImageData](#) &data)
Writes an image definition entity.
- void [writeLayer](#) ([DL_WriterA](#) &dw, const [DL_LayerData](#) &data, const [DL_Attributes](#) &attrib)
Writes a layer to the file.
- void [writeLinetype](#) ([DL_WriterA](#) &dw, const [DL_LinetypeData](#) &data)
Writes a line type to the file.

- void [writeAppid](#) ([DL_WriterA](#) &dw, const std::string &name)
Writes the APPID section to the DXF file.
- void [writeBlock](#) ([DL_WriterA](#) &dw, const [DL_BlockData](#) &data)
Writes a block's definition (no entities) to the DXF file.
- void [writeEndBlock](#) ([DL_WriterA](#) &dw, const std::string &name)
Writes a block end.
- void [writeVPort](#) ([DL_WriterA](#) &dw)
Writes a viewport section.
- void [writeStyle](#) ([DL_WriterA](#) &dw, const [DL_StyleData](#) &style)
Writes a style section.
- void [writeView](#) ([DL_WriterA](#) &dw)
Writes a view section.
- void [writeUcs](#) ([DL_WriterA](#) &dw)
Writes a ucs section.
- void [writeDimStyle](#) ([DL_WriterA](#) &dw, double dimasz, double dimexe, double dimexo, double dimgap, double dimtxt)
Writes a dimstyle section.
- void [writeBlockRecord](#) ([DL_WriterA](#) &dw)
Writes a blockrecord section.
- void [writeBlockRecord](#) ([DL_WriterA](#) &dw, const std::string &name)
Writes a single block record with the given name.
- void [writeObjects](#) ([DL_WriterA](#) &dw, const std::string &appDictionaryName="")
Writes a objects 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 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 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 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 bool DL_Dxf::checkVariable (const char * *var*, DL_Codes::version *version*) [static]

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

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

References `readDxfGroups()`.

4.21.2.10 `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 `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(), handleLWPPolylineData(), handleMTextData(), handleSplineData(), handleXData(), handleXRecordData(), DL_CreationInterface::setAttributes(), DL_CreationInterface::setExtrusion(), and DL_Attributes::setLinetypeScale().

Referenced by readDxfGroups().

4.21.2.12 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 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 void DL_Dxf::test () [static]

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

Some test routines.

References `stripWhiteSpace()`.

4.21.2.15 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::dxfString()`, `DL_Writer::entity()`, `DL_Writer::entityAttributes()`, and `DL_TraceData::x`.

4.21.2.16 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::dxfInt()`, `DL_WriterA::dxfString()`, and `DL_Writer::tableAppidEntry()`.

4.21.2.17 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::dxfReal()`, `DL_WriterA::dxfString()`, `DL_Writer::entity()`, `DL_Writer::entityAttributes()`, and `DL_ArcData::radius`.

4.21.2.18 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 Aut*cad.

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

4.21.2.19 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 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 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>data</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::dxfInt(), DL_WriterA::dxfReal(), DL_WriterA::dxfString(), 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 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::dxfInt(), DL_WriterA::dxfReal(), DL_WriterA::dxfString(), 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 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 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 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 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 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::dxflnt(), DL_WriterA::dxflReal(), DL_WriterA::dxflString(), 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 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 Aut*cad.

References DL_WriterA::dxflHex(), DL_WriterA::dxflnt(), DL_WriterA::dxflReal(), and DL_WriterA::dxflString().

4.21.2.29 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::dxflReal(), DL_WriterA::dxflString(), DL_Writer::entity(), DL_Writer::entityAttributes(), DL_EllipseData::mx, DL_EllipseData::my, DL_EllipseData::mz, and DL_EllipseData::ratio.

4.21.2.30 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 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 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_Writer::entityAttributes(), DL_HatchData::numLoops, DL_HatchData::pattern, and DL_HatchData::solid.

4.21.2.33 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::dxfInt(), DL_WriterA::dxfReal(), DL_WriterA::dxfString(), DL_HatchData::originX, DL_HatchData::scale, and DL_HatchData::solid.

4.21.2.34 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::dxfBool(), DL_WriterA::dxfInt(), DL_WriterA::dxfReal(), DL_HatchEdgeData::mx, DL_HatchEdgeData::my, DL_HatchEdgeData::nControl, DL_HatchEdgeData::nFit, DL_HatchEdgeData::nKnots, DL_HatchEdgeData::radius, DL_HatchEdgeData::ratio, DL_Hatch-

EdgeData::type, DL_HatchEdgeData::x1, DL_HatchEdgeData::x2, DL_HatchEdgeData::y1, and DL_HatchEdgeData::y2.

4.21.2.35 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 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 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_Writer::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 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 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::dxReal()`, and `DL_KnotData::k`.

4.21.2.40 `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::dxHex()`, `DL_WriterA::dxflnt()`, `DL_WriterA::dxString()`, `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 `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::dxReal()`, `DL_WriterA::dxString()`, `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 `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::dxReal()`, `DL_LeaderVertexData::x`, and `DL_LeaderVertexData::y`.

4.21.2.43 `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 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::dxfInt(), DL_WriterA::dxfReal(), DL_WriterA::dxfString(), DL_LinetypeData::flags, DL_LinetypeData::name, DL_LinetypeData::numberOfDashes, DL_LinetypeData::pattern, DL_LinetypeData::patternLength, and DL_Writer::tableLinetypeEntry().

4.21.2.45 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::dxfInt(), DL_WriterA::dxfReal(), DL_WriterA::dxfString(), DL_Writer::entity(), DL_Writer::entityAttributes(), DL_MTextData::height, DL_MTextData::ipx, DL_MTextData::ipy, DL_MTextData::ipz, DL_MTextData::lineSpacingFactor, DL_MTextData::lineSpacingStyle, DL_MTextData::style, DL_MTextData::text, and DL_MTextData::width.

4.21.2.46 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::dxfHex(), DL_WriterA::dxfInt(), DL_WriterA::dxfReal(), DL_WriterA::dxfString(), DL_Writer::getNextHandle(), and DL_Writer::handle().

4.21.2.47 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 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::dxflnt(), DL_Writer::entity(), DL_Writer::entityAttributes(), DL_PointData::x, DL_PointData::y, and DL_PointData::z.

4.21.2.49 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::dxflnt(), DL_Writer::entity(), DL_Writer::entityAttributes(), DL_PolylineData::flags, DL_Attributes::getLayer(), and DL_PolylineData::number.

4.21.2.50 void DL_Dxf::writePolylineEnd (DL_WriterA & dw)

Writes the polyline end.

Only needed for DXF R12.

References DL_Writer::entity().

4.21.2.51 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::dxflnt(), DL_RayData::dy, DL_RayData::dz, DL_Writer::entity(), and DL_Writer::entityAttributes().

4.21.2.52 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 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::dxfInt(), 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 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::dxfInt(), 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 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::dxfInt(), 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 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::dxflReal(), DL_WriterA::dxflString(), DL_Writer::entity(), DL_Writer::entityAttributes(), DL_TraceData::thickness, and DL_TraceData::x.

4.21.2.57 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::dxflHex(), DL_WriterA::dxflInt(), and DL_WriterA::dxflString().

4.21.2.58 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::dxflReal(), DL_WriterA::dxflString(), DL_Writer::entity(), DL_VertexData::x, DL_VertexData::y, and DL_VertexData::z.

4.21.2.59 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::dxflHex(), DL_WriterA::dxflInt(), and DL_WriterA::dxflString().

4.21.2.60 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::dxflHex(), DL_WriterA::dxflInt(), DL_WriterA::dxflReal(), DL_WriterA::dxflString(), and DL_Writer::handle().

4.21.2.61 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::dxfString(), DL_XLineData::dy, DL_XLineData::dz, DL_Writer::entity(), and DL_Writer::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 (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 double DL_EllipseData::angle1

Startangle of ellipse in rad.

Referenced by DL_Dxf::writeEllipse().

4.22.3.2 double DL_EllipseData::angle2

Endangle of ellipse in rad.

Referenced by DL_Dxf::writeEllipse().

4.22.3.3 double DL_EllipseData::cx

X Coordinate of center point.

Referenced by DL_Dxf::writeEllipse().

4.22.3.4 double DL_EllipseData::cy

Y Coordinate of center point.

Referenced by DL_Dxf::writeEllipse().

4.22.3.5 double DL_EllipseData::cz

Z Coordinate of center point.

Referenced by DL_Dxf::writeEllipse().

4.22.3.6 double DL_EllipseData::mx

X coordinate of the endpoint of the major axis.

Referenced by DL_Dxf::writeEllipse().

4.22.3.7 double DL_EllipseData::my

Y coordinate of the endpoint of the major axis.

Referenced by DL_Dxf::writeEllipse().

4.22.3.8 double DL_EllipseData::mz

Z coordinate of the endpoint of the major axis.

Referenced by DL_Dxf::writeEllipse().

4.22.3.9 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:

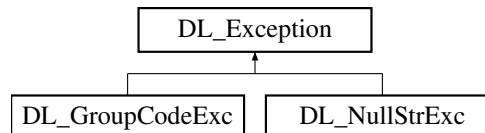
- src/dl_entities.h

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

Returns

direction vector.

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

Returns

direction vector.

4.24.3.3 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 (double x, double y, double z) `[inline]`

Constructor.

Parameters: see member variables.

4.25.3 Member Data Documentation

4.25.3.1 double DL_FitPointData::x

X coordinate of the fit point.

Referenced by DL_Dxf::writeFitPoint().

4.25.3.2 double DL_FitPointData::y

Y coordinate of the fit point.

Referenced by DL_Dxf::writeFitPoint().

4.25.3.3 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:

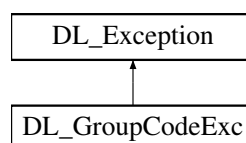
- `src/dl_entities.h`

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 (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 double DL_HatchData::angle

Pattern angle in degrees

Referenced by `DL_Dxf::writeHatch2()`.

4.27.3.2 int DL_HatchData::numLoops

Number of boundary paths (loops).

Referenced by DL_Dxf::writeHatch1().

4.27.3.3 double DL_HatchData::originX

Pattern origin

Referenced by DL_Dxf::writeHatch2().

4.27.3.4 std::string DL_HatchData::pattern

Pattern name.

Referenced by DL_Dxf::writeHatch1().

4.27.3.5 double DL_HatchData::scale

Pattern scale or spacing

Referenced by DL_Dxf::writeHatch2().

4.27.3.6 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::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::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::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::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 `double DL_HatchEdgeData::angle1`

Start angle of arc or ellipse arc.

Referenced by `DL_Dxf::handleHatchData()`, and `DL_Dxf::writeHatchEdge()`.

4.28.3.2 `double DL_HatchEdgeData::angle2`

End angle of arc or ellipse arc.

Referenced by `DL_Dxf::handleHatchData()`, and `DL_Dxf::writeHatchEdge()`.

4.28.3.3 `bool DL_HatchEdgeData::ccw`

Counterclockwise flag for arc or ellipse arc.

Referenced by `DL_Dxf::handleHatchData()`, and `DL_Dxf::writeHatchEdge()`.

4.28.3.4 `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 `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 unsigned int DL_HatchEdgeData::degree

Spline degree

Referenced by DL_Dxf::handleHatchData(), and DL_Dxf::writeHatchEdge().

4.28.3.7 double DL_HatchEdgeData::mx

Major axis end point (X).

Referenced by DL_Dxf::handleHatchData(), and DL_Dxf::writeHatchEdge().

4.28.3.8 double DL_HatchEdgeData::my

Major axis end point (Y).

Referenced by DL_Dxf::handleHatchData(), and DL_Dxf::writeHatchEdge().

4.28.3.9 unsigned int DL_HatchEdgeData::nControl

Number of control points.

Referenced by DL_Dxf::handleHatchData(), and DL_Dxf::writeHatchEdge().

4.28.3.10 unsigned int DL_HatchEdgeData::nFit

Number of fit points.

Referenced by DL_Dxf::handleHatchData(), and DL_Dxf::writeHatchEdge().

4.28.3.11 unsigned int DL_HatchEdgeData::nKnots

Number of knots.

Referenced by DL_Dxf::handleHatchData(), and DL_Dxf::writeHatchEdge().

4.28.3.12 double DL_HatchEdgeData::radius

Arc radius.

Referenced by DL_Dxf::handleHatchData(), and DL_Dxf::writeHatchEdge().

4.28.3.13 double DL_HatchEdgeData::ratio

Axis ratio

Referenced by DL_Dxf::handleHatchData(), and DL_Dxf::writeHatchEdge().

4.28.3.14 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 double DL_HatchEdgeData::x1

Start point (X).

Referenced by DL_Dxf::handleHatchData(), and DL_Dxf::writeHatchEdge().

4.28.3.16 double DL_HatchEdgeData::x2

End point (X).

Referenced by DL_Dxf::handleHatchData(), and DL_Dxf::writeHatchEdge().

4.28.3.17 double DL_HatchEdgeData::y1

Start point (Y).

Referenced by DL_Dxf::handleHatchData(), and DL_Dxf::writeHatchEdge().

4.28.3.18 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 (int *hNumEdges*) `[inline]`

Constructor.

Parameters: see member variables.

4.29.3 Member Data Documentation

4.29.3.1 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 &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)

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 (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 `int DL_ImageData::brightness`

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

Referenced by `DL_Dxf::writeImage()`.

4.30.3.2 `int DL_ImageData::contrast`

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

Referenced by `DL_Dxf::writeImage()`.

4.30.3.3 `int DL_ImageData::fade`

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

Referenced by `DL_Dxf::writeImage()`.

4.30.3.4 `int DL_ImageData::height`

Height of image in pixel.

Referenced by `DL_Dxf::writeImage()`, and `DL_Dxf::writeImageDef()`.

4.30.3.5 `double DL_ImageData::ipx`

X Coordinate of insertion point.

Referenced by `DL_Dxf::writeImage()`.

4.30.3.6 `double DL_ImageData::ipy`

Y Coordinate of insertion point.

Referenced by `DL_Dxf::writeImage()`.

4.30.3.7 `double DL_ImageData::ipz`

Z Coordinate of insertion point.

Referenced by `DL_Dxf::writeImage()`.

4.30.3.8 `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 `double DL_ImageData::ux`

X Coordinate of u vector along bottom of image.

Referenced by `DL_Dxf::writeImage()`.

4.30.3.10 `double DL_ImageData::uy`

Y Coordinate of u vector along bottom of image.

Referenced by `DL_Dxf::writeImage()`.

4.30.3.11 `double DL_ImageData::uz`

Z Coordinate of u vector along bottom of image.

Referenced by `DL_Dxf::writeImage()`.

4.30.3.12 `double DL_ImageData::vx`

X Coordinate of v vector along left side of image.

Referenced by `DL_Dxf::writeImage()`.

4.30.3.13 `double DL_ImageData::vy`

Y Coordinate of v vector along left side of image.

Referenced by `DL_Dxf::writeImage()`.

4.30.3.14 `double DL_ImageData::vz`

Z Coordinate of v vector along left side of image.

Referenced by `DL_Dxf::writeImage()`.

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

Constructor.

Parameters: see member variables.

4.31.3 Member Data Documentation

4.31.3.1 `std::string DL_ImageDefData::file`

Image file

4.31.3.2 `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 (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 double `DL_InsertData::angle`

Rotation angle in degrees.

Referenced by `DL_Dxf::writeInsert()`.

4.32.3.2 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 double `DL_InsertData::colSp`

Values for the spacing between cols.

Referenced by `DL_Dxf::writeInsert()`.

4.32.3.4 double `DL_InsertData::ipx`

X Coordinate of insertion point.

Referenced by `DL_Dxf::writeInsert()`.

4.32.3.5 double DL_InsertData::ipy

Y Coordinate of insertion point.

Referenced by DL_Dxf::writeInsert().

4.32.3.6 double DL_InsertData::ipz

Z Coordinate of insertion point.

Referenced by DL_Dxf::writeInsert().

4.32.3.7 std::string DL_InsertData::name

Name of the referred block.

Referenced by DL_Dxf::writeInsert().

4.32.3.8 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 double DL_InsertData::rowSp

Values for the spacing between rows.

Referenced by DL_Dxf::writeInsert().

4.32.3.10 double DL_InsertData::sx

X Scale factor.

Referenced by DL_Dxf::writeInsert().

4.32.3.11 double DL_InsertData::sy

Y Scale factor.

Referenced by DL_Dxf::writeInsert().

4.32.3.12 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 (double pk) [inline]`

Constructor.

Parameters: see member variables.

4.33.3 Member Data Documentation

4.33.3.1 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 (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 `int DL_LayerData::flags`

Layer flags.

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

Referenced by `DL_Dxf::writeLayer()`.

4.34.3.2 `std::string DL_LayerData::name`

Layer name.

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 (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 `int DL_LeaderData::arrowHeadFlag`

Arrow head flag (71).

Referenced by `DL_Dxf::writeLeader()`.

4.35.3.2 `double DL_LeaderData::dimScale`

Dimension scale (dimscale) style override.

4.35.3.3 `int DL_LeaderData::hooklineDirectionFlag`

Hookline direction flag (74).

Referenced by `DL_Dxf::writeLeader()`.

4.35.3.4 `int DL_LeaderData::hooklineFlag`

Hookline flag (75)

Referenced by `DL_Dxf::writeLeader()`.

4.35.3.5 `int DL_LeaderData::leaderCreationFlag`

Leader creation flag (73).

Referenced by `DL_Dxf::writeLeader()`.

4.35.3.6 `int DL_LeaderData::leaderPathType`

Leader path type (72).

Referenced by `DL_Dxf::writeLeader()`.

4.35.3.7 `int DL_LeaderData::number`

Number of vertices in leader (76).

Referenced by `DL_Dxf::writeLeader()`.

4.35.3.8 double DL_LeaderData::textAnnotationHeight

Text annotation height (40).

Referenced by DL_Dxf::writeLeader().

4.35.3.9 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 (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 double DL_LeaderVertexData::x

X Coordinate of the vertex.

Referenced by DL_Dxf::writeLeaderVertex().

4.36.3.2 double DL_LeaderVertexData::y

Y Coordinate of the vertex.

Referenced by DL_Dxf::writeLeaderVertex().

4.36.3.3 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 (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 double DL_LineData::x1

X Start coordinate of the point.

Referenced by DL_Dxf::writeLine().

4.37.3.2 double DL_LineData::x2

X End coordinate of the point.

Referenced by DL_Dxf::writeLine().

4.37.3.3 double DL_LineData::y1

Y Start coordinate of the point.

Referenced by DL_Dxf::writeLine().

4.37.3.4 double DL_LineData::y2

Y End coordinate of the point.

Referenced by DL_Dxf::writeLine().

4.37.3.5 double DL_LineData::z1

Z Start coordinate of the point.

Referenced by DL_Dxf::writeLine().

4.37.3.6 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 (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 (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 `double DL_MTextData::angle`

Rotation angle.

Referenced by `DL_Dxf::writeMText()`.

4.39.3.2 `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 `double DL_MTextData::dirx`

X Coordinate of X direction vector.

4.39.3.4 `double DL_MTextData::diry`

Y Coordinate of X direction vector.

4.39.3.5 `double DL_MTextData::dirz`

Z Coordinate of X direction vector.

4.39.3.6 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 double DL_MTextData::height

Text height

Referenced by DL_Dxf::writeMText().

4.39.3.8 double DL_MTextData::ipx

X Coordinate of insertion point.

Referenced by DL_Dxf::writeMText().

4.39.3.9 double DL_MTextData::ipy

Y Coordinate of insertion point.

Referenced by DL_Dxf::writeMText().

4.39.3.10 double DL_MTextData::ipz

Z Coordinate of insertion point.

Referenced by DL_Dxf::writeMText().

4.39.3.11 double DL_MTextData::lineSpacingFactor

Line spacing factor.

0.25 .. 4.0

Referenced by DL_Dxf::writeMText().

4.39.3.12 int DL_MTextData::lineSpacingStyle

Line spacing style.

1 = at least, 2 = exact

Referenced by DL_Dxf::writeMText().

4.39.3.13 std::string DL_MTextData::style

Style string.

Referenced by DL_Dxf::writeMText().

4.39.3.14 std::string DL_MTextData::text

Text string.

Referenced by DL_Dxf::writeMText().

4.39.3.15 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:

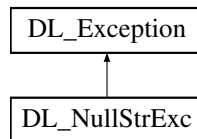
- src/dl_entities.h

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 (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 `double DL_PointData::x`

X Coordinate of the point.

Referenced by `DL_Dxf::writePoint()`.

4.41.3.2 `double DL_PointData::y`

Y Coordinate of the point.

Referenced by `DL_Dxf::writePoint()`.

4.41.3.3 `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 (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 `double DL_PolylineData::elevation`

elevation of the polyline.

4.42.3.2 `int DL_PolylineData::flags`

Flags

Referenced by `DL_Dxf::writePolyline()`.

4.42.3.3 `unsigned int DL_PolylineData::m`

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

4.42.3.4 `unsigned int DL_PolylineData::n`

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

4.42.3.5 `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 (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 `double DL_RayData::bx`

X base point.

Referenced by `DL_Dxf::writeRay()`.

4.43.3.2 `double DL_RayData::by`

Y base point.

Referenced by `DL_Dxf::writeRay()`.

4.43.3.3 `double DL_RayData::bz`

Z base point.

Referenced by `DL_Dxf::writeRay()`.

4.43.3.4 `double DL_RayData::dx`

X direction vector.

Referenced by `DL_Dxf::writeRay()`.

4.43.3.5 `double DL_RayData::dy`

Y direction vector.

Referenced by `DL_Dxf::writeRay()`.

4.43.3.6 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](#) (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 unsigned int [DL_SplineData::degree](#)

Degree of the spline curve.

Referenced by DL_Dxf::writeSpline().

4.44.3.2 int DL_SplineData::flags

Flags

Referenced by DL_Dxf::writeSpline().

4.44.3.3 unsigned int DL_SplineData::nControl

Number of control points.

Referenced by DL_Dxf::writeSpline().

4.44.3.4 unsigned int DL_SplineData::nFit

Number of fit points.

Referenced by DL_Dxf::writeSpline().

4.44.3.5 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.

4.45.2 Member Data Documentation

4.45.2.1 double DL_StyleData::fixedTextHeight

Fixed text height or 0 for not fixed.

Referenced by `DL_Dxf::writeStyle()`.

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

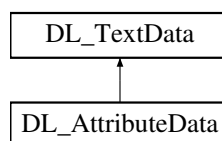
- `src/dl_entities.h`

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 (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 double `DL_TextData::angle`

Rotation angle of dimension text away from default orientation.

Referenced by `DL_Dxf::writeText()`.

4.46.3.2 double `DL_TextData::apx`

X Coordinate of alignment point.

Referenced by `DL_Dxf::writeText()`.

4.46.3.3 double `DL_TextData::apy`

Y Coordinate of alignment point.

Referenced by `DL_Dxf::writeText()`.

4.46.3.4 double `DL_TextData::apz`

Z Coordinate of alignment point.

Referenced by `DL_Dxf::writeText()`.

4.46.3.5 double DL_TextData::height

Text height

Referenced by DL_Dxf::writeText().

4.46.3.6 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 double DL_TextData::ipx

X Coordinate of insertion point.

Referenced by DL_Dxf::writeText().

4.46.3.8 double DL_TextData::ipy

Y Coordinate of insertion point.

Referenced by DL_Dxf::writeText().

4.46.3.9 double DL_TextData::ipz

Z Coordinate of insertion point.

Referenced by DL_Dxf::writeText().

4.46.3.10 std::string DL_TextData::style

Style (font).

Referenced by DL_Dxf::writeText().

4.46.3.11 std::string DL_TextData::text

Text string.

Referenced by DL_Dxf::writeText().

4.46.3.12 int DL_TextData::textGenerationFlags

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

Referenced by DL_Dxf::writeText().

4.46.3.13 int DL_TextData::vJustification

Vertical justification.

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

Referenced by DL_Dxf::writeText().

4.46.3.14 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 (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 double DL_TraceData::thickness

Thickness

Referenced by DL_Dxf::writeSolid(), and DL_Dxf::writeTrace().

4.47.3.2 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](#) (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 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 double DL_VertexData::x

X Coordinate of the vertex.

Referenced by DL_Dxf::writeVertex().

4.48.3.3 double DL_VertexData::y

Y Coordinate of the vertex.

Referenced by DL_Dxf::writeVertex().

4.48.3.4 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:

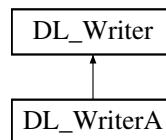
- src/dl_entities.h

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 [dxfReal](#) (int gc, double value) const =0
Must be overwritten by the implementing class to write a real value to the file.
- virtual void [dxfInt](#) (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 [dxfHex](#) (int gc, int value) const =0
Must be overwritten by the implementing class to write an int value (hex) to the file.
- virtual void [dxfString](#) (int gc, const char *value) const =0
Must be overwritten by the implementing class to write a string to the file.
- virtual void [dxfString](#) (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_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 `void DL_Writer::comment (const char * text) const` `[inline]`

Comment.

```
999
text
```

Referenced by DL_Dxf::writeHeader().

4.49.3.2 `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 `void DL_Writer::dxfEOF () const` `[inline]`

End of the DXF file.

```
0
EOF
```

4.49.3.4 `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 `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 `virtual void DL_Writer::dxflReal (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 `virtual void DL_Writer::dxflString (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 `virtual void DL_Writer::dxflString (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 `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 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 unsigned long DL_Writer::getNextHandle () const [inline]

Returns

Next handle that will be written.

Referenced by DL_Dxf::writeObjects().

4.49.3.12 void DL_Writer::section (const char * name) const [inline]

Generic section for section 'name'.

```

0
SECTION
2
name

```

4.49.3.13 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 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 void DL_Writer::sectionBlocks () const [inline]

Section BLOCKS.

```
0
SECTION
2
BLOCKS
```

4.49.3.16 void DL_Writer::sectionClasses () const [inline]

Section CLASSES.

```
0
SECTION
2
CLASSES
```

4.49.3.17 void DL_Writer::sectionEnd () const [inline]

End of a section.

```
0
ENDSEC
```

4.49.3.18 void DL_Writer::sectionEntities () const [inline]

Section ENTITIES.

```
0
SECTION
2
ENTITIES
```

4.49.3.19 void DL_Writer::sectionHeader () const [inline]

Section HEADER.

```
0
SECTION
2
HEADER
```

Referenced by DL_Dxf::writeHeader().

4.49.3.20 void DL_Writer::sectionObjects () const [inline]

Section OBJECTS.

```
0
SECTION
2
OBJECTS
```

4.49.3.21 void DL_Writer::sectionTables () const [inline]

Section TABLES.

```
0
SECTION
2
TABLES
```

4.49.3.22 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 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 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 void DL_Writer::tableEnd () const [inline]

End of a table.

```
0
ENDTAB
```

4.49.3.26 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 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 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 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 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:

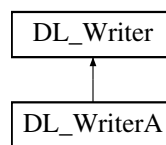
- src/dl_writer.h

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.

`fname` File name of the file to be created. `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 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 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 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 `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 `dxfHex()`, `dxfReal()`, `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 `void DL_WriterA::dxfString (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 `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 (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 `double DL_XLineData::bx`

X base point.

Referenced by `DL_Dxf::writeXLine()`.

4.51.3.2 `double DL_XLineData::by`

Y base point.

Referenced by `DL_Dxf::writeXLine()`.

4.51.3.3 `double DL_XLineData::bz`

Z base point.

Referenced by `DL_Dxf::writeXLine()`.

4.51.3.4 `double DL_XLineData::dx`

X direction vector.

Referenced by `DL_Dxf::writeXLine()`.

4.51.3.5 double DL_XLineData::dy

Y direction vector.

Referenced by DL_Dxf::writeXLine().

4.51.3.6 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

addArcAlignedText
 DL_CreationAdapter, [25](#)
 DL_CreationInterface, [30](#)
addAttribute
 DL_CreationAdapter, [25](#)
 DL_CreationInterface, [30](#)
 DL_Dxf, [52](#)
addBlock
 DL_CreationAdapter, [25](#)
 DL_CreationInterface, [30](#)
addInsert
 DL_CreationAdapter, [25](#)
 DL_CreationInterface, [30](#)
addMText
 DL_CreationAdapter, [25](#)
 DL_CreationInterface, [30](#)
addMTextChunk
 DL_CreationAdapter, [25](#)
 DL_CreationInterface, [30](#)
addSolid
 DL_Dxf, [52](#)
addText
 DL_CreationAdapter, [26](#)
 DL_CreationInterface, [31](#)
addTrace
 DL_Dxf, [52](#)
alignment
 DL_ArcAlignedTextData, [10](#)
angle
 DL_DimLinearData, [43](#)
 DL_HatchData, [76](#)
 DL_InsertData, [86](#)
 DL_MTextData, [95](#)
 DL_TextData, [104](#)
angle1
 DL_ArcData, [13](#)
 DL_EllipseData, [72](#)
 DL_HatchEdgeData, [79](#)
angle2
 DL_ArcData, [13](#)
 DL_EllipseData, [72](#)
 DL_HatchEdgeData, [79](#)
apx
 DL_TextData, [104](#)
apy
 DL_TextData, [104](#)
apz
 DL_TextData, [104](#)
arcHandle
 DL_ArcAlignedTextData, [10](#)
arrowHeadFlag
 DL_LeaderData, [90](#)
attachmentPoint
 DL_DimensionData, [41](#)
 DL_MTextData, [95](#)

bold
 DL_ArcAlignedTextData, [10](#)
bpx
 DL_BlockData, [19](#)
bpy
 DL_BlockData, [19](#)
bpz
 DL_BlockData, [19](#)
brightness
 DL_ImageData, [83](#)
bulge
 DL_VertexData, [107](#)
bx
 DL_RayData, [100](#)
 DL_XLineData, [121](#)
by
 DL_RayData, [100](#)
 DL_XLineData, [121](#)
bz
 DL_RayData, [100](#)
 DL_XLineData, [121](#)

ccw
 DL_HatchEdgeData, [79](#)
characerSet
 DL_ArcAlignedTextData, [10](#)
checkVariable
 DL_Dxf, [53](#)
colSp
 DL_InsertData, [86](#)
cols
 DL_InsertData, [86](#)
comment
 DL_Writer, [110](#)
contrast
 DL_ImageData, [83](#)
cx
 DL_ArcAlignedTextData, [10](#)
 DL_ArcData, [13](#)
 DL_CircleData, [20](#)
 DL_EllipseData, [72](#)
 DL_HatchEdgeData, [79](#)
cy

- DL_ArcAlignedTextData, 10
- DL_ArcData, 13
- DL_CircleData, 20
- DL_EllipseData, 72
- DL_HatchEdgeData, 79
- cz
 - DL_ArcAlignedTextData, 10
 - DL_ArcData, 14
 - DL_CircleData, 20
 - DL_EllipseData, 72
- DL_ArcAlignedTextData, 9
 - alignment, 10
 - arcHandle, 10
 - bold, 10
 - characerSet, 10
 - cx, 10
 - cy, 10
 - cz, 10
 - direction, 10
 - endAngle, 10
 - font, 10
 - height, 11
 - italic, 11
 - leftOffset, 11
 - offset, 11
 - pitch, 11
 - radius, 11
 - reversedCharacterOrder, 11
 - rightOffset, 11
 - shxFont, 11
 - side, 12
 - spacing, 12
 - startAngle, 12
 - style, 12
 - text, 12
 - underline, 12
 - wizard, 12
 - xScaleFactor, 12
- DL_ArcData, 13
 - angle1, 13
 - angle2, 13
 - cx, 13
 - cy, 13
 - cz, 14
 - DL_ArcData, 13
 - DL_ArcData, 13
 - radius, 14
- DL_AttributeData, 14
 - DL_AttributeData, 15
 - DL_AttributeData, 15
 - tag, 15
- DL_Attributes, 15
 - DL_Attributes, 16
 - DL_Attributes, 16
 - getColor, 16
 - getColor24, 16
 - getLayer, 17
 - getLinetype, 17
 - getWidth, 17
 - setColor, 17
 - setColor24, 17
 - setLayer, 17
 - setLinetype, 18
- DL_BlockData, 18
 - bpx, 19
 - bpy, 19
 - bpz, 19
 - DL_BlockData, 18
 - DL_BlockData, 18
 - flags, 19
 - name, 19
- DL_CircleData, 19
 - cx, 20
 - cy, 20
 - cz, 20
 - DL_CircleData, 20
 - DL_CircleData, 20
 - radius, 20
- DL_Codes, 20
- DL_ControlPointData, 21
 - DL_ControlPointData, 21
 - DL_ControlPointData, 21
 - w, 21
 - x, 21
 - y, 22
 - z, 22
- DL_CreationAdapter, 22
 - addArcAlignedText, 25
 - addAttribute, 25
 - addBlock, 25
 - addInsert, 25
 - addMText, 25
 - addMTextChunk, 25
 - addText, 26
 - processCodeValuePair, 26
 - setVariableDouble, 26
 - setVariableInt, 26
 - setVariableString, 26
 - setVariableVector, 26
- DL_CreationInterface, 27
 - addArcAlignedText, 30
 - addAttribute, 30
 - addBlock, 30
 - addInsert, 30
 - addMText, 30
 - addMTextChunk, 30
 - addText, 31
 - getAttributes, 31
 - getExtrusion, 31
 - processCodeValuePair, 31
 - setAttributes, 31
 - setExtrusion, 31
 - setVariableDouble, 31
 - setVariableInt, 32
 - setVariableString, 32
 - setVariableVector, 32

- DL_DictionaryData, [32](#)
- DL_DictionaryEntryData, [33](#)
- DL_DimAlignedData, [33](#)
 - DL_DimAlignedData, [34](#)
 - DL_DimAlignedData, [34](#)
 - epx1, [34](#)
 - epx2, [34](#)
 - epy1, [34](#)
 - epy2, [34](#)
 - epz1, [34](#)
 - epz2, [34](#)
- DL_DimAngular2LData, [34](#)
 - DL_DimAngular2LData, [35](#)
 - DL_DimAngular2LData, [35](#)
 - dpx1, [35](#)
 - dpx2, [35](#)
 - dpx3, [35](#)
 - dpx4, [36](#)
 - dpy1, [36](#)
 - dpy2, [36](#)
 - dpy3, [36](#)
 - dpy4, [36](#)
 - dpz1, [36](#)
 - dpz2, [36](#)
 - dpz3, [36](#)
 - dpz4, [36](#)
- DL_DimAngular3PData, [37](#)
 - DL_DimAngular3PData, [37](#)
 - DL_DimAngular3PData, [37](#)
 - dpx1, [37](#)
 - dpx2, [37](#)
 - dpx3, [37](#)
 - dpy1, [38](#)
 - dpy2, [38](#)
 - dpy3, [38](#)
 - dpz1, [38](#)
 - dpz2, [38](#)
 - dpz3, [38](#)
- DL_DimDiametricData, [38](#)
 - DL_DimDiametricData, [39](#)
 - DL_DimDiametricData, [39](#)
 - dpx, [39](#)
 - dpy, [39](#)
 - dpz, [39](#)
 - leader, [39](#)
- DL_DimLinearData, [42](#)
 - angle, [43](#)
 - DL_DimLinearData, [43](#)
 - DL_DimLinearData, [43](#)
 - dpx1, [43](#)
 - dpx2, [43](#)
 - dpy1, [43](#)
 - dpy2, [43](#)
 - dpz1, [43](#)
 - dpz2, [44](#)
 - oblique, [44](#)
- DL_DimOrdinateData, [44](#)
 - DL_DimOrdinateData, [44](#)
- DL_DimOrdinateData, [44](#)
 - DL_DimOrdinateData, [44](#)
- DL_DimRadialData, [45](#)
 - DL_DimRadialData, [46](#)
 - DL_DimRadialData, [46](#)
 - dpx, [46](#)
 - dpy, [46](#)
 - dpz, [46](#)
 - leader, [46](#)
- DL_DimensionData, [39](#)
 - attachmentPoint, [41](#)
 - DL_DimensionData, [40](#)
 - DL_DimensionData, [40](#)
 - dpx, [41](#)
 - dpy, [41](#)
 - dpz, [41](#)
 - lineSpacingFactor, [41](#)
 - lineSpacingStyle, [41](#)
 - mpx, [41](#)
 - mpy, [41](#)
 - mpz, [42](#)
 - style, [42](#)
 - text, [42](#)
 - type, [42](#)
- DL_Dxf, [46](#)
 - addAttribute, [52](#)
 - addSolid, [52](#)
 - addTrace, [52](#)
 - checkVariable, [53](#)
 - getDimData, [53](#)
 - getLibVersion, [53](#)
 - getStrippedLine, [53](#)
 - in, [54](#)
 - out, [54](#)
 - processDXFGroup, [54](#)
 - readDxfGroups, [56](#)
 - stripWhiteSpace, [56](#)
 - test, [57](#)
 - write3dFace, [57](#)
 - writeAppid, [57](#)
 - writeArc, [57](#)
 - writeBlockRecord, [57](#)
 - writeCircle, [57](#)
 - writeControlPoint, [59](#)
 - writeDimAligned, [59](#)
 - writeDimAngular2L, [59](#)
 - writeDimAngular3P, [59](#)
 - writeDimDiametric, [61](#)
 - writeDimLinear, [61](#)
 - writeDimOrdinate, [61](#)
 - writeDimRadial, [62](#)
 - writeDimStyle, [62](#)

- writeEllipse, 62
- writeEndBlock, 62
- writeFitPoint, 63
- writeHatch1, 63
- writeHatch2, 63
- writeHatchEdge, 63
- writeHatchLoop1, 64
- writeHatchLoop2, 64
- writelnImage, 64
- writeInsert, 64
- writeKnot, 64
- writeLayer, 66
- writeLeader, 66
- writeLeaderVertex, 66
- writeLine, 66
- writeLinetype, 67
- writeMText, 67
- writeObjects, 67
- writeObjectsEnd, 67
- writePoint, 67
- writePolyline, 68
- writePolylineEnd, 68
- writeRay, 68
- writeSolid, 68
- writeSpline, 69
- writeStyle, 69
- writeText, 69
- writeTrace, 69
- writeUcs, 70
- writeVPort, 70
- writeVertex, 70
- writeView, 70
- writeXLine, 70
- DL_EllipseData, 71
 - angle1, 72
 - angle2, 72
 - cx, 72
 - cy, 72
 - cz, 72
 - DL_EllipseData, 71
 - DL_EllipseData, 71
 - mx, 72
 - my, 72
 - mz, 72
 - ratio, 72
- DL_Exception, 73
- DL_Extrusion, 73
 - DL_Extrusion, 74
 - DL_Extrusion, 74
 - getDirection, 74
 - getElevation, 74
- DL_FitPointData, 74
 - DL_FitPointData, 75
 - DL_FitPointData, 75
 - x, 75
 - y, 75
 - z, 75
- DL_GroupCodeExc, 75
- DL_HatchData, 76
 - angle, 76
 - DL_HatchData, 76
 - DL_HatchData, 76
 - numLoops, 76
 - originX, 77
 - pattern, 77
 - scale, 77
 - solid, 77
- DL_HatchEdgeData, 77
 - angle1, 79
 - angle2, 79
 - ccw, 79
 - cx, 79
 - cy, 79
 - DL_HatchEdgeData, 78, 79
 - degree, 79
 - DL_HatchEdgeData, 78, 79
 - mx, 80
 - my, 80
 - nControl, 80
 - nFit, 80
 - nKnots, 80
 - radius, 80
 - ratio, 80
 - type, 80
 - x1, 80
 - x2, 81
 - y1, 81
 - y2, 81
- DL_HatchLoopData, 81
 - DL_HatchLoopData, 82
 - DL_HatchLoopData, 82
 - numEdges, 82
- DL_ImageData, 82
 - brightness, 83
 - contrast, 83
 - DL_ImageData, 83
 - DL_ImageData, 83
 - fade, 83
 - height, 83
 - ipx, 83
 - ipy, 83
 - ipz, 83
 - ref, 83
 - ux, 84
 - uy, 84
 - uz, 84
 - vx, 84
 - vy, 84
 - vz, 84
 - width, 84
- DL_ImageDefData, 84
 - DL_ImageDefData, 85
 - DL_ImageDefData, 85
 - file, 85
 - ref, 85
- DL_InsertData, 85

- angle, [86](#)
- colSp, [86](#)
- cols, [86](#)
- DL_InsertData, [86](#)
- DL_InsertData, [86](#)
- ipx, [86](#)
- ipy, [86](#)
- ipz, [87](#)
- name, [87](#)
- rowSp, [87](#)
- rows, [87](#)
- sx, [87](#)
- sy, [87](#)
- sz, [87](#)
- DL_KnotData, [87](#)
 - DL_KnotData, [88](#)
 - DL_KnotData, [88](#)
- k, [88](#)
- DL_LayerData, [88](#)
 - DL_LayerData, [89](#)
 - DL_LayerData, [89](#)
 - flags, [89](#)
 - name, [89](#)
- DL_LeaderData, [89](#)
 - arrowHeadFlag, [90](#)
 - DL_LeaderData, [90](#)
 - dimScale, [90](#)
 - DL_LeaderData, [90](#)
 - hooklineDirectionFlag, [90](#)
 - hooklineFlag, [90](#)
 - leaderCreationFlag, [90](#)
 - leaderPathType, [90](#)
 - number, [90](#)
 - textAnnotationHeight, [90](#)
 - textAnnotationWidth, [91](#)
- DL_LeaderVertexData, [91](#)
 - DL_LeaderVertexData, [91](#)
 - DL_LeaderVertexData, [91](#)
 - x, [91](#)
 - y, [91](#)
 - z, [92](#)
- DL_LineData, [92](#)
 - DL_LineData, [92](#)
 - DL_LineData, [92](#)
 - x1, [92](#)
 - x2, [92](#)
 - y1, [93](#)
 - y2, [93](#)
 - z1, [93](#)
 - z2, [93](#)
- DL_LinetypeData, [93](#)
 - DL_LinetypeData, [94](#)
 - DL_LinetypeData, [94](#)
- DL_MTextData, [94](#)
 - angle, [95](#)
 - attachmentPoint, [95](#)
 - DL_MTextData, [95](#)
 - dirx, [95](#)
 - diry, [95](#)
 - dirz, [95](#)
 - DL_MTextData, [95](#)
 - drawingDirection, [95](#)
 - height, [96](#)
 - ipx, [96](#)
 - ipy, [96](#)
 - ipz, [96](#)
 - lineSpacingFactor, [96](#)
 - lineSpacingStyle, [96](#)
 - style, [96](#)
 - text, [96](#)
 - width, [96](#)
- DL_NullStrExc, [97](#)
- DL_PointData, [97](#)
 - DL_PointData, [98](#)
 - DL_PointData, [98](#)
 - x, [98](#)
 - y, [98](#)
 - z, [98](#)
- DL_PolylineData, [98](#)
 - DL_PolylineData, [99](#)
 - DL_PolylineData, [99](#)
 - elevation, [99](#)
 - flags, [99](#)
 - m, [99](#)
 - n, [99](#)
 - number, [99](#)
- DL_RayData, [99](#)
 - bx, [100](#)
 - by, [100](#)
 - bz, [100](#)
 - DL_RayData, [100](#)
 - DL_RayData, [100](#)
 - dx, [100](#)
 - dy, [100](#)
 - dz, [100](#)
- DL_SplineData, [101](#)
 - DL_SplineData, [101](#)
 - degree, [101](#)
 - DL_SplineData, [101](#)
 - flags, [101](#)
 - nControl, [102](#)
 - nFit, [102](#)
 - nKnots, [102](#)
- DL_StyleData, [102](#)
 - fixedTextHeight, [103](#)
- DL_TextData, [103](#)
 - angle, [104](#)
 - apx, [104](#)
 - apy, [104](#)
 - apz, [104](#)
 - DL_TextData, [104](#)
 - DL_TextData, [104](#)
 - hJustification, [105](#)
 - height, [104](#)
 - ipx, [105](#)
 - ipy, [105](#)

- ipz, [105](#)
- style, [105](#)
- text, [105](#)
- textGenerationFlags, [105](#)
- vJustification, [105](#)
- xScaleFactor, [105](#)
- DL_TraceData, [106](#)
 - DL_TraceData, [106](#)
 - DL_TraceData, [106](#)
 - thickness, [106](#)
 - x, [106](#)
- DL_VertexData, [107](#)
 - bulge, [107](#)
 - DL_VertexData, [107](#)
 - DL_VertexData, [107](#)
 - x, [107](#)
 - y, [107](#)
 - z, [108](#)
- DL_Writer, [108](#)
 - comment, [110](#)
 - DL_Writer, [110](#)
 - DL_Writer, [110](#)
 - dxflBool, [110](#)
 - dxflEOF, [110](#)
 - dxflHex, [110](#)
 - dxflInt, [112](#)
 - dxflReal, [112](#)
 - dxflString, [112](#)
 - entity, [112](#)
 - entityAttributes, [113](#)
 - getNextHandle, [113](#)
 - section, [113](#)
 - sectionBlockEntry, [113](#)
 - sectionBlockEntryEnd, [113](#)
 - sectionBlocks, [114](#)
 - sectionClasses, [114](#)
 - sectionEnd, [114](#)
 - sectionEntities, [114](#)
 - sectionHeader, [114](#)
 - sectionObjects, [114](#)
 - sectionTables, [115](#)
 - table, [115](#)
 - tableAppid, [115](#)
 - tableAppidEntry, [115](#)
 - tableEnd, [115](#)
 - tableLayerEntry, [116](#)
 - tableLayers, [116](#)
 - tableLinetypeEntry, [116](#)
 - tableLinetypes, [116](#)
 - tableStyle, [116](#)
- DL_WriterA, [118](#)
 - dxflHex, [119](#)
 - dxflInt, [119](#)
 - dxflReal, [119](#)
 - dxflString, [120](#)
 - openFailed, [120](#)
- DL_XLineData, [120](#)
 - bx, [121](#)
 - by, [121](#)
 - bz, [121](#)
 - DL_XLineData, [121](#)
 - DL_XLineData, [121](#)
 - dx, [121](#)
 - dy, [121](#)
 - dz, [122](#)
- degree
 - DL_HatchEdgeData, [79](#)
 - DL_SplineData, [101](#)
- dimScale
 - DL_LeaderData, [90](#)
- direction
 - DL_ArcAlignedTextData, [10](#)
- dirx
 - DL_MTextData, [95](#)
- diry
 - DL_MTextData, [95](#)
- dirz
 - DL_MTextData, [95](#)
- dpx
 - DL_DimDiametricData, [39](#)
 - DL_DimensionData, [41](#)
 - DL_DimRadialData, [46](#)
- dpx1
 - DL_DimAngular2LData, [35](#)
 - DL_DimAngular3PData, [37](#)
 - DL_DimLinearData, [43](#)
 - DL_DimOrdinateData, [45](#)
- dpx2
 - DL_DimAngular2LData, [35](#)
 - DL_DimAngular3PData, [37](#)
 - DL_DimLinearData, [43](#)
 - DL_DimOrdinateData, [45](#)
- dpx3
 - DL_DimAngular2LData, [35](#)
 - DL_DimAngular3PData, [37](#)
- dpx4
 - DL_DimAngular2LData, [36](#)
- dpy
 - DL_DimDiametricData, [39](#)
 - DL_DimensionData, [41](#)
 - DL_DimRadialData, [46](#)
- dpy1
 - DL_DimAngular2LData, [36](#)
 - DL_DimAngular3PData, [38](#)
 - DL_DimLinearData, [43](#)
 - DL_DimOrdinateData, [45](#)
- dpy2
 - DL_DimAngular2LData, [36](#)
 - DL_DimAngular3PData, [38](#)
 - DL_DimLinearData, [43](#)
 - DL_DimOrdinateData, [45](#)
- dpy3
 - DL_DimAngular2LData, [36](#)
 - DL_DimAngular3PData, [38](#)
- dpy4
 - DL_DimAngular2LData, [36](#)

- dpz
 - DL_DimDiametricData, 39
 - DL_DimensionData, 41
 - DL_DimRadialData, 46
- dpz1
 - DL_DimAngular2LData, 36
 - DL_DimAngular3PData, 38
 - DL_DimLinearData, 43
 - DL_DimOrdinateData, 45
- dpz2
 - DL_DimAngular2LData, 36
 - DL_DimAngular3PData, 38
 - DL_DimLinearData, 44
 - DL_DimOrdinateData, 45
- dpz3
 - DL_DimAngular2LData, 36
 - DL_DimAngular3PData, 38
- dpz4
 - DL_DimAngular2LData, 36
- drawingDirection
 - DL_MTextData, 95
- dx
 - DL_RayData, 100
 - DL_XLineData, 121
- dxfBool
 - DL_Writer, 110
- dxfeof
 - DL_Writer, 110
- dxfHex
 - DL_Writer, 110
 - DL_WriterA, 119
- dxflnt
 - DL_Writer, 112
 - DL_WriterA, 119
- dxflReal
 - DL_Writer, 112
 - DL_WriterA, 119
- dxflString
 - DL_Writer, 112
 - DL_WriterA, 120
- dy
 - DL_RayData, 100
 - DL_XLineData, 121
- dz
 - DL_RayData, 100
 - DL_XLineData, 122
- elevation
 - DL_PolylineData, 99
- endAngle
 - DL_ArcAlignedTextData, 10
- entity
 - DL_Writer, 112
- entityAttributes
 - DL_Writer, 113
- epx1
 - DL_DimAlignedData, 34
- epx2
 - DL_DimAlignedData, 34
- epy1
 - DL_DimAlignedData, 34
- epy2
 - DL_DimAlignedData, 34
- epz1
 - DL_DimAlignedData, 34
- epz2
 - DL_DimAlignedData, 34
- fade
 - DL_ImageData, 83
- file
 - DL_ImageDefData, 85
- fixedTextHeight
 - DL_StyleData, 103
- flags
 - DL_BlockData, 19
 - DL_LayerData, 89
 - DL_PolylineData, 99
 - DL_SplineData, 101
- font
 - DL_ArcAlignedTextData, 10
- getAttributes
 - DL_CreationInterface, 31
- getColor
 - DL_Attributes, 16
- getColor24
 - DL_Attributes, 16
- getDimData
 - DL_Dxf, 53
- getDirection
 - DL_Extrusion, 74
- getElevation
 - DL_Extrusion, 74
- getExtrusion
 - DL_CreationInterface, 31
- getLayer
 - DL_Attributes, 17
- getLibVersion
 - DL_Dxf, 53
- getLinetype
 - DL_Attributes, 17
- getNextHandle
 - DL_Writer, 113
- getStrippedLine
 - DL_Dxf, 53
- getWidth
 - DL_Attributes, 17
- hJustification
 - DL_TextData, 105
- height
 - DL_ArcAlignedTextData, 11
 - DL_ImageData, 83
 - DL_MTextData, 96
 - DL_TextData, 104
- hooklineDirectionFlag
 - DL_LeaderData, 90

- hooklineFlag
 - DL_LeaderData, 90
- in
 - DL_Dxf, 54
- ipx
 - DL_ImageData, 83
 - DL_InsertData, 86
 - DL_MTextData, 96
 - DL_TextData, 105
- ipy
 - DL_ImageData, 83
 - DL_InsertData, 86
 - DL_MTextData, 96
 - DL_TextData, 105
- ipz
 - DL_ImageData, 83
 - DL_InsertData, 87
 - DL_MTextData, 96
 - DL_TextData, 105
- italic
 - DL_ArcAlignedTextData, 11
- k
 - DL_KnotData, 88
- leader
 - DL_DimDiametricData, 39
 - DL_DimRadialData, 46
- leaderCreationFlag
 - DL_LeaderData, 90
- leaderPathType
 - DL_LeaderData, 90
- leftOffset
 - DL_ArcAlignedTextData, 11
- lineSpacingFactor
 - DL_DimensionData, 41
 - DL_MTextData, 96
- lineSpacingStyle
 - DL_DimensionData, 41
 - DL_MTextData, 96
- m
 - DL_PolylineData, 99
- mpx
 - DL_DimensionData, 41
- mpy
 - DL_DimensionData, 41
- mpz
 - DL_DimensionData, 42
- mx
 - DL_EllipseData, 72
 - DL_HatchEdgeData, 80
- my
 - DL_EllipseData, 72
 - DL_HatchEdgeData, 80
- mz
 - DL_EllipseData, 72
- n
 - DL_PolylineData, 99
- nControl
 - DL_HatchEdgeData, 80
 - DL_SplineData, 102
- nFit
 - DL_HatchEdgeData, 80
 - DL_SplineData, 102
- nKnots
 - DL_HatchEdgeData, 80
 - DL_SplineData, 102
- name
 - DL_BlockData, 19
 - DL_InsertData, 87
 - DL_LayerData, 89
- numEdges
 - DL_HatchLoopData, 82
- numLoops
 - DL_HatchData, 76
- number
 - DL_LeaderData, 90
 - DL_PolylineData, 99
- oblique
 - DL_DimLinearData, 44
- offset
 - DL_ArcAlignedTextData, 11
- openFailed
 - DL_WriterA, 120
- originX
 - DL_HatchData, 77
- out
 - DL_Dxf, 54
- pattern
 - DL_HatchData, 77
- pitch
 - DL_ArcAlignedTextData, 11
- processCodeValuePair
 - DL_CreationAdapter, 26
 - DL_CreationInterface, 31
- processDXFGroup
 - DL_Dxf, 54
- radius
 - DL_ArcAlignedTextData, 11
 - DL_ArcData, 14
 - DL_CircleData, 20
 - DL_HatchEdgeData, 80
- ratio
 - DL_EllipseData, 72
 - DL_HatchEdgeData, 80
- readDxfGroups
 - DL_Dxf, 56
- ref
 - DL_ImageData, 83
 - DL_ImageDefData, 85
- reversedCharacterOrder
 - DL_ArcAlignedTextData, 11
- rightOffset

- DL_ArcAlignedTextData, 11
- rowSp
 - DL_InsertData, 87
- rows
 - DL_InsertData, 87
- scale
 - DL_HatchData, 77
- section
 - DL_Writer, 113
- sectionBlockEntry
 - DL_Writer, 113
- sectionBlockEntryEnd
 - DL_Writer, 113
- sectionBlocks
 - DL_Writer, 114
- sectionClasses
 - DL_Writer, 114
- sectionEnd
 - DL_Writer, 114
- sectionEntities
 - DL_Writer, 114
- sectionHeader
 - DL_Writer, 114
- sectionObjects
 - DL_Writer, 114
- sectionTables
 - DL_Writer, 115
- setAttributes
 - DL_CreationInterface, 31
- setColor
 - DL_Attributes, 17
- setColor24
 - DL_Attributes, 17
- setExtrusion
 - DL_CreationInterface, 31
- setLayer
 - DL_Attributes, 17
- setLinetype
 - DL_Attributes, 18
- setVariableDouble
 - DL_CreationAdapter, 26
 - DL_CreationInterface, 31
- setVariableInt
 - DL_CreationAdapter, 26
 - DL_CreationInterface, 32
- setVariableString
 - DL_CreationAdapter, 26
 - DL_CreationInterface, 32
- setVariableVector
 - DL_CreationAdapter, 26
 - DL_CreationInterface, 32
- shxFont
 - DL_ArcAlignedTextData, 11
- side
 - DL_ArcAlignedTextData, 12
- solid
 - DL_HatchData, 77
- spacing
 - DL_ArcAlignedTextData, 12
- startAngle
 - DL_ArcAlignedTextData, 12
- stripWhiteSpace
 - DL_Dxf, 56
- style
 - DL_ArcAlignedTextData, 12
 - DL_DimensionData, 42
 - DL_MTextData, 96
 - DL_TextData, 105
- sx
 - DL_InsertData, 87
- sy
 - DL_InsertData, 87
- sz
 - DL_InsertData, 87
- table
 - DL_Writer, 115
- tableAppid
 - DL_Writer, 115
- tableAppidEntry
 - DL_Writer, 115
- tableEnd
 - DL_Writer, 115
- tableLayerEntry
 - DL_Writer, 116
- tableLayers
 - DL_Writer, 116
- tableLinetypeEntry
 - DL_Writer, 116
- tableLinetypes
 - DL_Writer, 116
- tableStyle
 - DL_Writer, 116
- tag
 - DL_AttributeData, 15
- test
 - DL_Dxf, 57
- text
 - DL_ArcAlignedTextData, 12
 - DL_DimensionData, 42
 - DL_MTextData, 96
 - DL_TextData, 105
- textAnnotationHeight
 - DL_LeaderData, 90
- textAnnotationWidth
 - DL_LeaderData, 91
- textGenerationFlags
 - DL_TextData, 105
- thickness
 - DL_TraceData, 106
- type
 - DL_DimensionData, 42
 - DL_HatchEdgeData, 80
- underline
 - DL_ArcAlignedTextData, 12
- ux

- DL_ImageData, [84](#)
- uy
 - DL_ImageData, [84](#)
- uz
 - DL_ImageData, [84](#)
- vJustification
 - DL_TextData, [105](#)
- vx
 - DL_ImageData, [84](#)
- vy
 - DL_ImageData, [84](#)
- vz
 - DL_ImageData, [84](#)
- w
 - DL_ControlPointData, [21](#)
- width
 - DL_ImageData, [84](#)
 - DL_MTextData, [96](#)
- wizard
 - DL_ArcAlignedTextData, [12](#)
- write3dFace
 - DL_Dxf, [57](#)
- writeAppid
 - DL_Dxf, [57](#)
- writeArc
 - DL_Dxf, [57](#)
- writeBlockRecord
 - DL_Dxf, [57](#)
- writeCircle
 - DL_Dxf, [57](#)
- writeControlPoint
 - DL_Dxf, [59](#)
- writeDimAligned
 - DL_Dxf, [59](#)
- writeDimAngular2L
 - DL_Dxf, [59](#)
- writeDimAngular3P
 - DL_Dxf, [59](#)
- writeDimDiametric
 - DL_Dxf, [61](#)
- writeDimLinear
 - DL_Dxf, [61](#)
- writeDimOrdinate
 - DL_Dxf, [61](#)
- writeDimRadial
 - DL_Dxf, [62](#)
- writeDimStyle
 - DL_Dxf, [62](#)
- writeEllipse
 - DL_Dxf, [62](#)
- writeEndBlock
 - DL_Dxf, [62](#)
- writeFitPoint
 - DL_Dxf, [63](#)
- writeHatch1
 - DL_Dxf, [63](#)
- writeHatch2
 - DL_Dxf, [63](#)
- writeHatchEdge
 - DL_Dxf, [63](#)
- writeHatchLoop1
 - DL_Dxf, [64](#)
- writeHatchLoop2
 - DL_Dxf, [64](#)
- writeImage
 - DL_Dxf, [64](#)
- writeInsert
 - DL_Dxf, [64](#)
- writeKnot
 - DL_Dxf, [64](#)
- writeLayer
 - DL_Dxf, [66](#)
- writeLeader
 - DL_Dxf, [66](#)
- writeLeaderVertex
 - DL_Dxf, [66](#)
- writeLine
 - DL_Dxf, [66](#)
- writeLinetype
 - DL_Dxf, [67](#)
- writeMText
 - DL_Dxf, [67](#)
- writeObjects
 - DL_Dxf, [67](#)
- writeObjectsEnd
 - DL_Dxf, [67](#)
- writePoint
 - DL_Dxf, [67](#)
- writePolyline
 - DL_Dxf, [68](#)
- writePolylineEnd
 - DL_Dxf, [68](#)
- writeRay
 - DL_Dxf, [68](#)
- writeSolid
 - DL_Dxf, [68](#)
- writeSpline
 - DL_Dxf, [69](#)
- writeStyle
 - DL_Dxf, [69](#)
- writeText
 - DL_Dxf, [69](#)
- writeTrace
 - DL_Dxf, [69](#)
- writeUcs
 - DL_Dxf, [70](#)
- writeVPort
 - DL_Dxf, [70](#)
- writeVertex
 - DL_Dxf, [70](#)
- writeView
 - DL_Dxf, [70](#)
- writeXLine
 - DL_Dxf, [70](#)
- x

- DL_ControlPointData, [21](#)
- DL_FitPointData, [75](#)
- DL_LeaderVertexData, [91](#)
- DL_PointData, [98](#)
- DL_TraceData, [106](#)
- DL_VertexData, [107](#)
- x1
 - DL_HatchEdgeData, [80](#)
 - DL_LineData, [92](#)
- x2
 - DL_HatchEdgeData, [81](#)
 - DL_LineData, [92](#)
- xScaleFactor
 - DL_ArcAlignedTextData, [12](#)
 - DL_TextData, [105](#)
- xtype
 - DL_DimOrdinateData, [45](#)
- y
 - DL_ControlPointData, [22](#)
 - DL_FitPointData, [75](#)
 - DL_LeaderVertexData, [91](#)
 - DL_PointData, [98](#)
 - DL_VertexData, [107](#)
- y1
 - DL_HatchEdgeData, [81](#)
 - DL_LineData, [93](#)
- y2
 - DL_HatchEdgeData, [81](#)
 - DL_LineData, [93](#)
- z
 - DL_ControlPointData, [22](#)
 - DL_FitPointData, [75](#)
 - DL_LeaderVertexData, [92](#)
 - DL_PointData, [98](#)
 - DL_VertexData, [108](#)
- z1
 - DL_LineData, [93](#)
- z2
 - DL_LineData, [93](#)