



Delta3D Version 2.4.0

# **dtUtil::**

## **Reference Manual**



# Contents

---

<b>1</b>	<b>Main Page</b>	<b>1</b>
<b>2</b>	<b>Todo List</b>	<b>3</b>
<b>3</b>	<b>Deprecated List</b>	<b>5</b>
<b>4</b>	<b>Directory Documentation</b>	<b>7</b>
4.1	src/dtUtil/ Directory Reference . . . . .	7
4.2	inc/dtUtil/ Directory Reference . . . . .	8
4.3	inc/ Directory Reference . . . . .	10
4.4	src/ Directory Reference . . . . .	11
<b>5</b>	<b>Namespace Documentation</b>	<b>13</b>
5.1	dtUtil Namespace Reference . . . . .	13
5.1.1	Detailed Description . . . . .	22
5.1.2	Typedef Documentation . . . . .	23
5.1.2.1	DirectoryContents . . . . .	23
5.1.2.2	FileExtensionList . . . . .	23
5.1.2.3	Fractal2d . . . . .	23
5.1.2.4	Fractal2f . . . . .	23
5.1.2.5	Fractal3d . . . . .	23
5.1.2.6	Fractal3f . . . . .	23
5.1.2.7	Noise1d . . . . .	23
5.1.2.8	Noise1f . . . . .	23
5.1.2.9	Noise2d . . . . .	23
5.1.2.10	Noise2f . . . . .	23
5.1.2.11	Noise3d . . . . .	23
5.1.2.12	Noise3f . . . . .	23
5.1.2.13	SeamlessFractal . . . . .	23
5.1.3	Enumeration Type Documentation . . . . .	23
5.1.3.1	FileType . . . . .	23
5.1.4	Function Documentation . . . . .	23
5.1.4.1	_S_accumulate_node_distance . . . . .	23
5.1.4.2	_S_node_compare . . . . .	23
5.1.4.3	_S_node_descend . . . . .	23
5.1.4.4	_S_node_distance . . . . .	23
5.1.4.5	_S_node_nearest . . . . .	24
5.1.4.6	Abs . . . . .	25

---

5.1.4.7	Bind	25
5.1.4.8	Bind	25
5.1.4.9	Bind	25
5.1.4.10	Bind	25
5.1.4.11	Bind	25
5.1.4.12	Bind	25
5.1.4.13	Bind	25
5.1.4.14	Bind	25
5.1.4.15	CalculateBarycentricCenter	25
5.1.4.16	CalculateNormal	25
5.1.4.17	Clamp	26
5.1.4.18	ClampMax	26
5.1.4.19	ClampMin	26
5.1.4.20	construct	26
5.1.4.21	Equivalent	26
5.1.4.22	Equivalent	26
5.1.4.23	Equivalent	26
5.1.4.24	Equivalent	26
5.1.4.25	Equivalent	27
5.1.4.26	FindAttributeValueFor	27
5.1.4.27	getFileExtensionIncludingDot	27
5.1.4.28	GetH	27
5.1.4.29	GetH	27
5.1.4.30	iMakeDirectory	27
5.1.4.31	IMPLEMENT_ENUM	27
5.1.4.32	IMPLEMENT_ENUM	27
5.1.4.33	IMPLEMENT_ENUM	27
5.1.4.34	IMPLEMENT_ENUM	27
5.1.4.35	IsFinite	27
5.1.4.36	IsFiniteVec	27
5.1.4.37	IsNaN	27
5.1.4.38	Lerp	27
5.1.4.39	MakeFunctor	28
5.1.4.40	MakeFunctor	28
5.1.4.41	MakeFunctor	28
5.1.4.42	MakeIndexString	28
5.1.4.43	manager	28
5.1.4.44	MapRangeValue	28
5.1.4.45	Match	28
5.1.4.46	Max	28
5.1.4.47	MergeParms	28

---

5.1.4.48	Min	28
5.1.4.49	operator!=	28
5.1.4.50	operator!=	28
5.1.4.51	operator!=	28
5.1.4.52	operator!=	28
5.1.4.53	operator+	28
5.1.4.54	operator<<	28
5.1.4.55	operator<<	28
5.1.4.56	operator<<	28
5.1.4.57	operator==	29
5.1.4.58	operator==	29
5.1.4.59	operator==	29
5.1.4.60	operator==	29
5.1.4.61	ParseVec	29
5.1.4.62	RandFloat	29
5.1.4.63	RandPercent	29
5.1.4.64	RandRange	29
5.1.4.65	safeASIN	29
5.1.4.66	Scan	29
5.1.4.67	sTitle	29
5.1.4.68	ToDouble	29
5.1.4.69	ToFloat	29
5.1.4.70	ToString	29
5.1.4.71	ToType	30
5.1.4.72	ToType< bool >	30
5.1.4.73	ToUnsignedInt	30
5.1.4.74	trim	30
5.1.4.75	Trim	30
5.1.4.76	WildMatch	30
5.1.4.77	WithinRange	30
5.1.5	Variable Documentation	30
5.1.5.1	AD_C	30
5.1.5.2	CentralMeridianScale	30
5.1.5.3	COS_67P5	30
5.1.5.4	flatteningReciprocal	30
5.1.5.5	Geocent_a	31
5.1.5.6	Geocent_e2	31
5.1.5.7	Geocent_e2_2	31
5.1.5.8	Geocent_e2_3	31
5.1.5.9	Geocent_ef	31
5.1.5.10	Geocent_ef_3	31

5.1.5.11	Geocent_ef_4 . . . . .	31
5.1.5.12	Geocent_ep2 . . . . .	31
5.1.5.13	Geocent_f . . . . .	31
5.1.5.14	gStringSetMutex . . . . .	31
5.1.5.15	LOGNAME . . . . .	31
5.1.5.16	MagneticNorthLatitude . . . . .	31
5.1.5.17	MagneticNorthLongitude . . . . .	31
5.1.5.18	MAX_DELTA_LONG . . . . .	31
5.1.5.19	MAX_EASTING . . . . .	31
5.1.5.20	MAX_LAT . . . . .	31
5.1.5.21	MAX_NORTHING . . . . .	31
5.1.5.22	MAX_SCALE_FACTOR . . . . .	31
5.1.5.23	METERS_PER_DEGREE . . . . .	31
5.1.5.24	MIN_EASTING . . . . .	31
5.1.5.25	MIN_LAT . . . . .	31
5.1.5.26	MIN_NORTHING . . . . .	31
5.1.5.27	MIN_SCALE_FACTOR . . . . .	31
5.1.5.28	p . . . . .	31
5.1.5.29	permutation . . . . .	31
5.1.5.30	semiMajorAxis . . . . .	31
5.1.5.31	sLogFileName . . . . .	31
5.1.5.32	StringCount . . . . .	31
5.1.5.33	StringSet . . . . .	31
5.2	dtUtil::Bits Namespace Reference . . . . .	32
5.2.1	Detailed Description . . . . .	32
5.2.2	Function Documentation . . . . .	32
5.2.2.1	Add . . . . .	32
5.2.2.2	Has . . . . .	32
5.2.2.3	Remove . . . . .	32
5.2.2.4	Toggle . . . . .	32
5.3	dtUtil::CollectorUtil Namespace Reference . . . . .	33
5.3.1	Function Documentation . . . . .	33
5.3.1.1	AddNode . . . . .	33
5.3.1.2	FindNodePointer . . . . .	33
5.3.1.3	FindNodePointer . . . . .	34
5.3.1.4	RemoveNode . . . . .	34
5.4	dtUtil::details Namespace Reference . . . . .	35
<b>6</b>	<b>Class Documentation</b>	<b>37</b>
6.1	_Alloc_base<_Tp, _Alloc > Class Template Reference . . . . .	37
6.1.1	Member Typedef Documentation . . . . .	37

6.1.1.1	_Base_ptr . . . . .	37
6.1.1.2	_Node_ . . . . .	37
6.1.1.3	allocator_type . . . . .	37
6.1.2	Constructor & Destructor Documentation . . . . .	38
6.1.2.1	_Alloc_base . . . . .	38
6.1.3	Member Function Documentation . . . . .	38
6.1.3.1	_M_allocate_node . . . . .	38
6.1.3.2	_M_construct_node . . . . .	38
6.1.3.3	_M_deallocate_node . . . . .	38
6.1.3.4	_M_destroy_node . . . . .	38
6.1.3.5	get_allocator . . . . .	38
6.1.4	Member Data Documentation . . . . .	38
6.1.4.1	_M_node_allocator . . . . .	38
6.2	_Base_iterator Class Reference . . . . .	39
6.2.1	Member Typedef Documentation . . . . .	39
6.2.1.1	_Base_const_ptr . . . . .	39
6.2.2	Constructor & Destructor Documentation . . . . .	39
6.2.2.1	_Base_iterator . . . . .	39
6.2.2.2	_Base_iterator . . . . .	39
6.2.3	Member Function Documentation . . . . .	39
6.2.3.1	_M_decrement . . . . .	39
6.2.3.2	_M_increment . . . . .	39
6.2.4	Friends And Related Function Documentation . . . . .	39
6.2.4.1	KDTree . . . . .	39
6.2.5	Member Data Documentation . . . . .	39
6.2.5.1	_M_node . . . . .	39
6.3	_Bracket_accessor<_Val> Struct Template Reference . . . . .	40
6.3.1	Member Typedef Documentation . . . . .	40
6.3.1.1	result_type . . . . .	40
6.3.2	Member Function Documentation . . . . .	40
6.3.2.1	operator() . . . . .	40
6.4	_Iterator<_Val, _Ref, _Ptr> Class Template Reference . . . . .	41
6.4.1	Member Typedef Documentation . . . . .	42
6.4.1.1	_Link_const_type . . . . .	42
6.4.1.2	_Self . . . . .	42
6.4.1.3	const_iterator . . . . .	42
6.4.1.4	difference_type . . . . .	42
6.4.1.5	iterator . . . . .	42
6.4.1.6	iterator_category . . . . .	42
6.4.1.7	pointer . . . . .	42
6.4.1.8	reference . . . . .	42

6.4.1.9	value_type	42
6.4.2	Constructor & Destructor Documentation	42
6.4.2.1	_Iterator	42
6.4.2.2	_Iterator	42
6.4.2.3	_Iterator	42
6.4.3	Member Function Documentation	42
6.4.3.1	get_raw_node	42
6.4.3.2	operator*	42
6.4.3.3	operator++	42
6.4.3.4	operator++	42
6.4.3.5	operator--	42
6.4.3.6	operator--	42
6.4.3.7	operator->	42
6.4.4	Friends And Related Function Documentation	42
6.4.4.1	operator!=	42
6.4.4.2	operator!=	42
6.4.4.3	operator!=	42
6.4.4.4	operator==	42
6.4.4.5	operator==	42
6.4.4.6	operator==	42
6.5	_Node<_Val> Struct Template Reference	43
6.5.1	Member Typedef Documentation	43
6.5.1.1	_Base_ptr	43
6.5.1.2	_Link_type	43
6.5.2	Constructor & Destructor Documentation	43
6.5.2.1	_Node	43
6.5.3	Member Data Documentation	43
6.5.3.1	_M_value	43
6.6	_Node_base Struct Reference	44
6.6.1	Member Typedef Documentation	44
6.6.1.1	_Base_const_ptr	44
6.6.1.2	_Base_ptr	44
6.6.2	Constructor & Destructor Documentation	44
6.6.2.1	_Node_base	44
6.6.3	Member Function Documentation	44
6.6.3.1	_S_maximum	44
6.6.3.2	_S_minimum	44
6.6.4	Member Data Documentation	44
6.6.4.1	_M_left	44
6.6.4.2	_M_parent	44
6.6.4.3	_M_right	44

6.7	<code>_Node_compare&lt;_Val, _Acc, _Cmp &gt;</code> Class Template Reference . . . . .	45
6.7.1	Constructor & Destructor Documentation . . . . .	45
6.7.1.1	<code>_Node_compare</code> . . . . .	45
6.7.2	Member Function Documentation . . . . .	45
6.7.2.1	<code>operator()</code> . . . . .	45
6.8	<code>_Region&lt;_K, _Val, _SubVal, _Acc, _Cmp &gt;</code> Struct Template Reference . . . . .	46
6.8.1	Member Typedef Documentation . . . . .	47
6.8.1.1	<code>_CenterPt</code> . . . . .	47
6.8.1.2	<code>subvalue_type</code> . . . . .	47
6.8.1.3	<code>value_type</code> . . . . .	47
6.8.2	Constructor & Destructor Documentation . . . . .	47
6.8.2.1	<code>_Region</code> . . . . .	47
6.8.2.2	<code>_Region</code> . . . . .	47
6.8.2.3	<code>_Region</code> . . . . .	47
6.8.3	Member Function Documentation . . . . .	47
6.8.3.1	<code>encloses</code> . . . . .	47
6.8.3.2	<code>intersects_with</code> . . . . .	47
6.8.3.3	<code>intersects_with</code> . . . . .	47
6.8.3.4	<code>set_high_bound</code> . . . . .	47
6.8.3.5	<code>set_low_bound</code> . . . . .	47
6.8.4	Member Data Documentation . . . . .	47
6.8.4.1	<code>_M_acc</code> . . . . .	47
6.8.4.2	<code>_M_cmp</code> . . . . .	47
6.8.4.3	<code>_M_high_bounds</code> . . . . .	47
6.8.4.4	<code>_M_low_bounds</code> . . . . .	47
6.9	<code>always_true&lt;_Tp &gt;</code> Struct Template Reference . . . . .	48
6.9.1	Member Function Documentation . . . . .	48
6.9.1.1	<code>operator()</code> . . . . .	48
6.10	<code>AppendTL&lt; TList, T &gt;</code> Struct Template Reference . . . . .	49
6.10.1	Member Typedef Documentation . . . . .	49
6.10.1.1	<code>Type</code> . . . . .	49
6.11	<code>AppendTL&lt; NullType, T &gt;</code> Struct Template Reference . . . . .	50
6.11.1	Member Typedef Documentation . . . . .	50
6.11.1.1	<code>Type</code> . . . . .	50
6.12	<code>array_remove&lt;_Container &gt;</code> Class Template Reference . . . . .	51
6.12.1	Member Typedef Documentation . . . . .	51
6.12.1.1	<code>container_type</code> . . . . .	51
6.12.1.2	<code>reference</code> . . . . .	51
6.12.2	Constructor & Destructor Documentation . . . . .	51
6.12.2.1	<code>array_remove</code> . . . . .	51
6.12.3	Member Function Documentation . . . . .	51

6.12.3.1	operator()	51
6.12.4	Member Data Documentation	51
6.12.4.1	container	51
6.13	AttributeSearch Class Reference	52
6.13.1	Detailed Description	52
6.13.2	Member Typedef Documentation	52
6.13.2.1	ResultMap	52
6.13.3	Constructor & Destructor Documentation	52
6.13.3.1	AttributeSearch	52
6.13.3.2	~AttributeSearch	52
6.13.4	Member Function Documentation	52
6.13.4.1	operator()	52
6.14	BarycentricSpace< VecT > Class Template Reference	53
6.14.1	Detailed Description	53
6.14.2	Constructor & Destructor Documentation	53
6.14.2.1	BarycentricSpace	53
6.14.3	Member Function Documentation	53
6.14.3.1	Transform	53
6.15	BaseExceptionType Class Reference	54
6.15.1	Constructor & Destructor Documentation	54
6.15.1.1	BaseExceptionType	54
6.15.2	Member Data Documentation	54
6.15.2.1	GENERAL_EXCEPTION	54
6.16	Binder< Incoming, BoundIdsTL > Class Template Reference	55
6.16.1	Member Typedef Documentation	56
6.16.1.1	BoundParamsTL	56
6.16.1.2	BoundPTL	56
6.16.1.3	Outgoing	56
6.16.1.4	Parm1	56
6.16.1.5	Parm2	56
6.16.1.6	Parm3	56
6.16.1.7	Parm4	56
6.16.1.8	Parm5	56
6.16.1.9	ResultType	56
6.16.1.10	TypeListType	56
6.16.1.11	UnboundParamsTL	56
6.16.1.12	UnboundPTL	56
6.16.2	Constructor & Destructor Documentation	56
6.16.2.1	Binder	56
6.16.2.2	Binder	56
6.16.3	Member Function Documentation	56

6.16.3.1	operator()	56
6.16.3.2	operator()	56
6.16.3.3	operator()	56
6.16.3.4	operator()	56
6.16.3.5	operator()	56
6.16.3.6	operator()	56
6.17	Bound Struct Reference	57
6.17.1	Member Function Documentation	57
6.17.1.1	MergeParms	57
6.18	BoundHelper< Incoming, IdsTL > Struct Template Reference	58
6.18.1	Member Typedef Documentation	58
6.18.1.1	BoundTL	58
6.18.1.2	IncomingTL	58
6.18.1.3	Parm1	58
6.18.1.4	Parm2	58
6.18.1.5	Parm3	58
6.18.1.6	Parm4	58
6.18.1.7	Parm5	58
6.19	BoundingBoxVisitor Class Reference	59
6.19.1	Constructor & Destructor Documentation	59
6.19.1.1	BoundingBoxVisitor	59
6.19.2	Member Function Documentation	59
6.19.2.1	apply	59
6.19.3	Member Data Documentation	59
6.19.3.1	mBoundingBox	59
6.20	BoundTL2< TL, IdsTL > Struct Template Reference	60
6.20.1	Member Typedef Documentation	60
6.20.1.1	Result	60
6.21	BreakOverride Struct Reference	61
6.21.1	Detailed Description	61
6.22	CallParms< TYPELIST_0()> Struct Template Reference	62
6.22.1	Member Typedef Documentation	62
6.22.1.1	ParmsListType	62
6.22.2	Member Function Documentation	62
6.22.2.1	Make	62
6.23	CallParms< TYPELIST_1(P1)> Struct Template Reference	63
6.23.1	Member Typedef Documentation	63
6.23.1.1	ParmsListType	63
6.23.2	Member Function Documentation	63
6.23.2.1	Make	63
6.24	CallParms< TYPELIST_2(P1, P2)> Struct Template Reference	64

6.24.1	Member Typedef Documentation . . . . .	64
6.24.1.1	ParmsListType . . . . .	64
6.24.2	Member Function Documentation . . . . .	64
6.24.2.1	Make . . . . .	64
6.25	CallParms< TYPELIST_3(P1, P2, P3)> Struct Template Reference . . . . .	65
6.25.1	Member Typedef Documentation . . . . .	65
6.25.1.1	ParmsListType . . . . .	65
6.25.2	Member Function Documentation . . . . .	65
6.25.2.1	Make . . . . .	65
6.26	CallParms< TYPELIST_4(P1, P2, P3, P4)> Struct Template Reference . . . . .	66
6.26.1	Member Typedef Documentation . . . . .	66
6.26.1.1	ParmsListType . . . . .	66
6.26.2	Member Function Documentation . . . . .	66
6.26.2.1	Make . . . . .	66
6.27	CallParms< TYPELIST_5(P1, P2, P3, P4, P5)> Struct Template Reference . . . . .	67
6.27.1	Member Typedef Documentation . . . . .	67
6.27.1.1	ParmsListType . . . . .	67
6.27.2	Member Function Documentation . . . . .	67
6.27.2.1	Make . . . . .	67
6.28	CallParms< TYPELIST_6(P1, P2, P3, P4, P5, P6)> Struct Template Reference . . . . .	68
6.28.1	Member Typedef Documentation . . . . .	68
6.28.1.1	ParmsListType . . . . .	68
6.28.2	Member Function Documentation . . . . .	68
6.28.2.1	Make . . . . .	68
6.29	CallParms< TYPELIST_7(P1, P2, P3, P4, P5, P6, P7)> Struct Template Reference . . . . .	69
6.29.1	Member Typedef Documentation . . . . .	69
6.29.1.1	ParmsListType . . . . .	69
6.29.2	Member Function Documentation . . . . .	69
6.29.2.1	Make . . . . .	69
6.30	Command< RetT > Class Template Reference . . . . .	70
6.30.1	Detailed Description . . . . .	70
6.30.2	Member Typedef Documentation . . . . .	70
6.30.2.1	ReturnType . . . . .	70
6.30.3	Constructor & Destructor Documentation . . . . .	70
6.30.3.1	Command . . . . .	70
6.30.3.2	~Command . . . . .	70
6.30.4	Member Function Documentation . . . . .	70
6.30.4.1	operator() . . . . .	70
6.31	Command0< RetT > Class Template Reference . . . . .	71
6.31.1	Detailed Description . . . . .	71
6.31.2	Member Typedef Documentation . . . . .	71

6.31.2.1	FunctorType	71
6.31.3	Constructor & Destructor Documentation	71
6.31.3.1	Command0	71
6.31.3.2	~Command0	71
6.31.4	Member Function Documentation	71
6.31.4.1	operator()	71
6.32	Command1< RetT, ArgT > Class Template Reference	72
6.32.1	Detailed Description	72
6.32.2	Member Typedef Documentation	72
6.32.2.1	FunctorType	72
6.32.2.2	MemberType	72
6.32.2.3	Param0	72
6.32.2.4	Params	72
6.32.3	Constructor & Destructor Documentation	72
6.32.3.1	Command1	72
6.32.3.2	~Command1	72
6.32.4	Member Function Documentation	72
6.32.4.1	operator()	72
6.32.4.2	TYPELIST_1	73
6.33	Command2< RetT, ArgTMember1, ArgTMember2 > Class Template Reference	74
6.33.1	Detailed Description	74
6.33.2	Member Typedef Documentation	74
6.33.2.1	FunctorType	74
6.33.2.2	MemberType1	74
6.33.2.3	MemberType2	74
6.33.2.4	Param0	74
6.33.2.5	Param1	75
6.33.2.6	Params	75
6.33.3	Constructor & Destructor Documentation	75
6.33.3.1	Command2	75
6.33.4	Member Function Documentation	75
6.33.4.1	operator()	75
6.33.4.2	TYPELIST_2	75
6.34	ConfigProperties Class Reference	76
6.34.1	Detailed Description	76
6.34.2	Member Function Documentation	76
6.34.2.1	GetConfigPropertyValue	76
6.34.2.2	RemoveConfigPropertyValue	76
6.34.2.3	SetConfigPropertyValue	76
6.35	CoordinateConversionExceptionEnum Class Reference	77
6.35.1	Member Data Documentation	77

6.35.1.1	INVALID_INPUT . . . . .	77
6.36	Coordinates Class Reference . . . . .	78
6.36.1	Constructor & Destructor Documentation . . . . .	80
6.36.1.1	Coordinates . . . . .	80
6.36.1.2	~Coordinates . . . . .	80
6.36.1.3	Coordinates . . . . .	80
6.36.2	Member Function Documentation . . . . .	80
6.36.2.1	CalculateConvergencParamForFlatEarth . . . . .	80
6.36.2.2	CalculateMagneticNorthOffset . . . . .	80
6.36.2.3	CalculateUTMZone . . . . .	80
6.36.2.4	ConvertFlatEarthToLatLon . . . . .	81
6.36.2.5	ConvertGeocentricToGeodetic . . . . .	81
6.36.2.6	ConvertGeodeticToTransverseMercator . . . . .	81
6.36.2.7	ConvertGeodeticToUTM . . . . .	81
6.36.2.8	ConvertLatLonToFlatEarth . . . . .	82
6.36.2.9	ConvertMGRSToUTM . . . . .	82
6.36.2.10	ConvertMGRSToXYZ . . . . .	82
6.36.2.11	ConvertToLocalRotation . . . . .	82
6.36.2.12	ConvertToLocalRotation . . . . .	82
6.36.2.13	ConvertToLocalTranslation . . . . .	82
6.36.2.14	ConvertToRemoteRotation . . . . .	82
6.36.2.15	ConvertToRemoteTranslation . . . . .	82
6.36.2.16	ConvertTransverseMercatorToGeodetic . . . . .	82
6.36.2.17	ConvertUTMToGeodetic . . . . .	82
6.36.2.18	ConvertUTMtoMGRS . . . . .	83
6.36.2.19	DegreesToMils . . . . .	83
6.36.2.20	EulersToMatrix . . . . .	83
6.36.2.21	GeocentricToGeodetic . . . . .	83
6.36.2.22	GeodeticToGeocentric . . . . .	83
6.36.2.23	GetApplyRotationConversionMatrix . . . . .	84
6.36.2.24	GetFlatEarthOrigin . . . . .	84
6.36.2.25	GetGlobeRadius . . . . .	84
6.36.2.26	GetIncomingCoordinateType . . . . .	84
6.36.2.27	GetLocalCoordinateType . . . . .	84
6.36.2.28	GetLocalOffset . . . . .	84
6.36.2.29	GetMagneticNorthOffset . . . . .	84
6.36.2.30	GetOriginRotationMatrix . . . . .	84
6.36.2.31	GetOriginRotationMatrixInverse . . . . .	84
6.36.2.32	GetUTMHemisphere . . . . .	84
6.36.2.33	GetUTMZone . . . . .	84
6.36.2.34	MatrixToEulers . . . . .	84

6.36.2.35	MilsToDegrees	84
6.36.2.36	operator=	84
6.36.2.37	operator==	84
6.36.2.38	ReconfigureRotationMatrix	84
6.36.2.39	SetApplyRotationConversionMatrix	85
6.36.2.40	SetFlatEarthOrigin	85
6.36.2.41	SetGlobeRadius	85
6.36.2.42	SetIncomingCoordinateType	85
6.36.2.43	SetLocalCoordinateType	85
6.36.2.44	SetLocalOffset	85
6.36.2.45	SetMagneticNorthOffset	85
6.36.2.46	SetUTMHemisphere	85
6.36.2.47	SetUTMLocalOffsetAsLatLon	85
6.36.2.48	SetUTMZone	85
6.36.2.49	XYZToMGRS	85
6.36.2.50	ZFlop	85
6.37	CreatesTL< i1, i2, i3, i4, i5, i6, i7, i8 > Struct Template Reference	86
6.37.1	Member Typedef Documentation	86
6.37.1.1	Type	86
6.38	CreatesTL< -1,-1,-1,-1,-1,-1,-1,-1 > Struct Template Reference	87
6.38.1	Member Typedef Documentation	87
6.38.1.1	Type	87
6.39	CreateTL< T1, T2, T3, T4, T5, T6, T7, T8 > Struct Template Reference	88
6.39.1	Member Typedef Documentation	88
6.39.1.1	Type	88
6.40	CreateTL< NullType, NullType, NullType, NullType, NullType, NullType, NullType, NullType > Struct Template Reference	89
6.40.1	Member Typedef Documentation	89
6.40.1.1	Type	89
6.41	DataStream Class Reference	90
6.41.1	Constructor & Destructor Documentation	92
6.41.1.1	DataStream	92
6.41.1.2	DataStream	92
6.41.1.3	DataStream	92
6.41.1.4	~DataStream	92
6.41.2	Member Function Documentation	92
6.41.2.1	AppendDataStream	92
6.41.2.2	ClearBuffer	92
6.41.2.3	GetBuffer	92
6.41.2.4	GetBufferCapacity	92
6.41.2.5	GetBufferSize	92

6.41.2.6	GetRemainingReadSize . . . . .	92
6.41.2.7	IncreaseBufferSize . . . . .	92
6.41.2.8	IsLittleEndian . . . . .	92
6.41.2.9	operator<< . . . . .	93
6.41.2.10	operator<< . . . . .	93
6.41.2.11	operator<< . . . . .	93
6.41.2.12	operator<< . . . . .	93
6.41.2.13	operator<< . . . . .	93
6.41.2.14	operator<< . . . . .	93
6.41.2.15	operator<< . . . . .	93
6.41.2.16	operator<< . . . . .	93
6.41.2.17	operator<< . . . . .	93
6.41.2.18	operator<< . . . . .	93
6.41.2.19	operator<< . . . . .	93
6.41.2.20	operator<< . . . . .	93
6.41.2.21	operator<< . . . . .	93
6.41.2.22	operator<< . . . . .	93
6.41.2.23	operator<< . . . . .	93
6.41.2.24	operator<< . . . . .	93
6.41.2.25	operator<< . . . . .	93
6.41.2.26	operator<< . . . . .	93
6.41.2.27	operator<< . . . . .	93
6.41.2.28	operator= . . . . .	93
6.41.2.29	operator>> . . . . .	93
6.41.2.30	operator>> . . . . .	93
6.41.2.31	operator>> . . . . .	93
6.41.2.32	operator>> . . . . .	93
6.41.2.33	operator>> . . . . .	93
6.41.2.34	operator>> . . . . .	93
6.41.2.35	operator>> . . . . .	93
6.41.2.36	operator>> . . . . .	93
6.41.2.37	operator>> . . . . .	93
6.41.2.38	operator>> . . . . .	93
6.41.2.39	operator>> . . . . .	93
6.41.2.40	operator>> . . . . .	93
6.41.2.41	operator>> . . . . .	93
6.41.2.42	operator>> . . . . .	93
6.41.2.43	operator>> . . . . .	93
6.41.2.44	operator>> . . . . .	93
6.41.2.45	operator>> . . . . .	93
6.41.2.46	operator>> . . . . .	93

6.41.2.47	operator>>	93
6.41.2.48	Read	93
6.41.2.49	Read	93
6.41.2.50	Read	93
6.41.2.51	Read	93
6.41.2.52	Read	93
6.41.2.53	Read	93
6.41.2.54	Read	93
6.41.2.55	Read	93
6.41.2.56	Read	93
6.41.2.57	Read	93
6.41.2.58	Read	93
6.41.2.59	Read	93
6.41.2.60	Read	93
6.41.2.61	Read	93
6.41.2.62	Read	93
6.41.2.63	Read	93
6.41.2.64	Read	93
6.41.2.65	Read	93
6.41.2.66	Read	93
6.41.2.67	ReadBinary	93
6.41.2.68	Rewind	93
6.41.2.69	Seekg	93
6.41.2.70	Seekp	93
6.41.2.71	SetBufferSize	93
6.41.2.72	SetForceLittleEndian	93
6.41.2.73	Write	94
6.41.2.74	Write	94
6.41.2.75	Write	94
6.41.2.76	Write	94
6.41.2.77	Write	94
6.41.2.78	Write	94
6.41.2.79	Write	94
6.41.2.80	Write	94
6.41.2.81	Write	94
6.41.2.82	Write	94
6.41.2.83	Write	94
6.41.2.84	Write	94
6.41.2.85	Write	94
6.41.2.86	Write	94
6.41.2.87	Write	94

6.41.2.88	Write . . . . .	94
6.41.2.89	Write . . . . .	94
6.41.2.90	Write . . . . .	94
6.41.2.91	Write . . . . .	94
6.41.2.92	WriteBinary . . . . .	94
6.41.2.93	WriteBytes . . . . .	94
6.42	DataStreamException Class Reference . . . . .	95
6.42.1	Member Data Documentation . . . . .	95
6.42.1.1	BUFFER_INVALID . . . . .	95
6.42.1.2	BUFFER_INVALID_POS . . . . .	95
6.42.1.3	BUFFER_READ_ERROR . . . . .	95
6.42.1.4	BUFFER_WRITE_ERROR . . . . .	95
6.43	DateTime Class Reference . . . . .	96
6.43.1	Constructor & Destructor Documentation . . . . .	98
6.43.1.1	DateTime . . . . .	98
6.43.1.2	DateTime . . . . .	98
6.43.1.3	DateTime . . . . .	98
6.43.1.4	DateTime . . . . .	98
6.43.1.5	DateTime . . . . .	98
6.43.1.6	~DateTime . . . . .	98
6.43.2	Member Function Documentation . . . . .	98
6.43.2.1	AdjustTimeZone . . . . .	98
6.43.2.2	GetDay . . . . .	99
6.43.2.3	GetGMTOffset . . . . .	99
6.43.2.4	GetGMTTime . . . . .	99
6.43.2.5	GetGMTTime . . . . .	99
6.43.2.6	GetGMTTime . . . . .	99
6.43.2.7	GetGMTTime . . . . .	99
6.43.2.8	GetHour . . . . .	99
6.43.2.9	GetLocalGMTOffset . . . . .	99
6.43.2.10	GetMinute . . . . .	99
6.43.2.11	GetMonth . . . . .	99
6.43.2.12	GetSecond . . . . .	99
6.43.2.13	GetTime . . . . .	99
6.43.2.14	GetTime . . . . .	99
6.43.2.15	GetTime . . . . .	100
6.43.2.16	GetTime . . . . .	100
6.43.2.17	GetTimeFormat . . . . .	100
6.43.2.18	GetTimeInSeconds . . . . .	100
6.43.2.19	GetTimeOrigin . . . . .	100
6.43.2.20	GetTimeScale . . . . .	100

6.43.2.21	GetTimeType . . . . .	100
6.43.2.22	GetYear . . . . .	100
6.43.2.23	IncrementClock . . . . .	100
6.43.2.24	operator std::string . . . . .	101
6.43.2.25	operator time_t . . . . .	101
6.43.2.26	operator tm . . . . .	101
6.43.2.27	operator= . . . . .	101
6.43.2.28	SetDay . . . . .	101
6.43.2.29	SetGMTOffset . . . . .	101
6.43.2.30	SetGMTOffset . . . . .	101
6.43.2.31	SetGMTOffset . . . . .	101
6.43.2.32	SetHour . . . . .	101
6.43.2.33	SetMinute . . . . .	101
6.43.2.34	SetMonth . . . . .	101
6.43.2.35	SetSecond . . . . .	101
6.43.2.36	SetTime . . . . .	101
6.43.2.37	SetTime . . . . .	101
6.43.2.38	SetTime . . . . .	101
6.43.2.39	SetTime . . . . .	101
6.43.2.40	SetTimeFormat . . . . .	102
6.43.2.41	SetTimeOrigin . . . . .	102
6.43.2.42	SetTimeScale . . . . .	102
6.43.2.43	SetTimeType . . . . .	102
6.43.2.44	SetToGMTTime . . . . .	102
6.43.2.45	SetToLocalTime . . . . .	102
6.43.2.46	SetYear . . . . .	102
6.43.2.47	ToString . . . . .	102
6.43.2.48	ToString . . . . .	102
6.43.2.49	ToString . . . . .	102
6.44	DeprecatedFunction Struct Reference . . . . .	103
6.44.1	Member Data Documentation . . . . .	103
6.44.1.1	CalledFrom . . . . .	103
6.44.1.2	NewFunctionName . . . . .	103
6.44.1.3	OldFunctionName . . . . .	103
6.45	DeprecationMgr Class Reference . . . . .	104
6.45.1	Detailed Description . . . . .	104
6.45.2	Constructor & Destructor Documentation . . . . .	104
6.45.2.1	~DeprecationMgr . . . . .	104
6.45.3	Member Function Documentation . . . . .	104
6.45.3.1	AddDeprecatedFunction . . . . .	104
6.45.3.2	GetInstance . . . . .	104

6.46	DeprecationMgrImpl Class Reference	105
6.46.1	Member Data Documentation	105
6.46.1.1	mFunctions	105
6.47	DoNothing< Ret, T > Struct Template Reference	106
6.47.1	Detailed Description	106
6.47.2	Member Function Documentation	106
6.47.2.1	operator()	106
6.48	DoNothing0< Ret > Struct Template Reference	107
6.48.1	Detailed Description	107
6.48.2	Member Function Documentation	107
6.48.2.1	operator()	107
6.49	EdgeStepFilter Class Reference	108
6.49.1	Detailed Description	108
6.49.2	Constructor & Destructor Documentation	108
6.49.2.1	EdgeStepFilter	108
6.49.3	Member Function Documentation	108
6.49.3.1	operator()	108
6.50	EmptyType Struct Reference	109
6.51	Enumeration Class Reference	110
6.51.1	Detailed Description	111
6.51.2	Constructor & Destructor Documentation	111
6.51.2.1	~Enumeration	111
6.51.2.2	Enumeration	111
6.51.3	Member Function Documentation	111
6.51.3.1	GetName	111
6.51.3.2	operator!=	111
6.51.3.3	operator!=	111
6.51.3.4	operator<	111
6.51.3.5	operator<	111
6.51.3.6	operator==	111
6.51.3.7	operator==	112
6.51.3.8	operator>	112
6.52	EvaluateFunctor< FunctorType, ArgType, RetType > Struct Template Reference	113
6.52.1	Member Function Documentation	113
6.52.1.1	operator()	113
6.53	EvaluateFunctor< FunctorType *, ArgType, RetType > Struct Template Reference	114
6.53.1	Member Function Documentation	114
6.53.1.1	operator()	114
6.54	EvaluateInvoke< CompareHandler, InvokeHandler, EvaluateResult > Struct Template Reference	115
6.54.1	Member Function Documentation	115
6.54.1.1	operator()	115

6.55	Exception Class Reference	116
6.55.1	Constructor & Destructor Documentation	116
6.55.1.1	Exception	116
6.55.1.2	Exception	117
6.55.1.3	~Exception	117
6.55.2	Member Function Documentation	117
6.55.2.1	File	117
6.55.2.2	Line	117
6.55.2.3	LogException	117
6.55.2.4	LogException	117
6.55.2.5	LogException	117
6.55.2.6	Print	117
6.55.2.7	ToString	117
6.55.2.8	TypeEnum	117
6.55.2.9	What	117
6.55.3	Member Data Documentation	117
6.55.3.1	mFileName	117
6.55.3.2	mLineNum	117
6.55.3.3	mMessage	117
6.55.3.4	mType	117
6.56	ExceptionEnum Class Reference	118
6.56.1	Constructor & Destructor Documentation	118
6.56.1.1	ExceptionEnum	118
6.56.2	Member Data Documentation	118
6.56.2.1	LibraryLoadingError	118
6.57	FileExceptionEnum Class Reference	119
6.57.1	Detailed Description	119
6.57.2	Constructor & Destructor Documentation	119
6.57.2.1	FileExceptionEnum	119
6.57.3	Member Data Documentation	119
6.57.3.1	FileNotFound	119
6.57.3.2	IOException	119
6.58	FileInfo Struct Reference	120
6.58.1	Constructor & Destructor Documentation	120
6.58.1.1	FileInfo	120
6.58.2	Member Data Documentation	120
6.58.2.1	baseName	120
6.58.2.2	fileName	120
6.58.2.3	fileType	120
6.58.2.4	lastModified	120
6.58.2.5	path	120

6.58.2.6	size	120
6.59	FileUtils Class Reference	121
6.59.1	Member Function Documentation	122
6.59.1.1	ChangeDirectory	122
6.59.1.2	CurrentDirectory	122
6.59.1.3	DirCopy	122
6.59.1.4	DirDelete	122
6.59.1.5	DirExists	123
6.59.1.6	DirGetFiles	123
6.59.1.7	DirGetSubs	123
6.59.1.8	FileCopy	123
6.59.1.9	FileDelete	124
6.59.1.10	FileExists	124
6.59.1.11	FileMove	124
6.59.1.12	GetAbsolutePath	124
6.59.1.13	GetFileInfo	124
6.59.1.14	GetInstance	124
6.59.1.15	IsSameFile	125
6.59.1.16	MakeDirectory	125
6.59.1.17	PopDirectory	125
6.59.1.18	PushDirectory	125
6.59.1.19	RelativePath	125
6.59.2	Member Data Documentation	126
6.59.2.1	PATH_SEPARATOR	126
6.60	Filter2< TL, IdsTL, include > Struct Template Reference	127
6.60.1	Member Typedef Documentation	127
6.60.1.1	Result	127
6.61	Fractal< Real, Vector, Noise > Class Template Reference	128
6.61.1	Detailed Description	128
6.61.2	Member Function Documentation	128
6.61.2.1	FBM	128
6.61.2.2	HeteroFractal	128
6.61.2.3	IslandFractal	128
6.61.2.4	Marble	129
6.61.2.5	RigidMultiFractal	129
6.61.2.6	Turbulence	129
6.62	Functor< R, TList, size > Class Template Reference	130
6.62.1	Member Typedef Documentation	131
6.62.1.1	Parm1	131
6.62.1.2	Parm2	131
6.62.1.3	Parm3	131

6.62.1.4	Parm4 . . . . .	131
6.62.1.5	Parm5 . . . . .	131
6.62.1.6	Parm6 . . . . .	131
6.62.1.7	Parm7 . . . . .	131
6.62.1.8	ParmsListType . . . . .	131
6.62.1.9	ResultType . . . . .	131
6.62.1.10	TypeListType . . . . .	131
6.62.2	Constructor & Destructor Documentation . . . . .	131
6.62.2.1	Functor . . . . .	131
6.62.2.2	~Functor . . . . .	131
6.62.2.3	Functor . . . . .	131
6.62.2.4	Functor . . . . .	131
6.62.2.5	Functor . . . . .	131
6.62.3	Member Function Documentation . . . . .	131
6.62.3.1	operator! . . . . .	131
6.62.3.2	operator() . . . . .	131
6.62.3.3	operator() . . . . .	131
6.62.3.4	operator() . . . . .	131
6.62.3.5	operator() . . . . .	131
6.62.3.6	operator() . . . . .	131
6.62.3.7	operator() . . . . .	131
6.62.3.8	operator() . . . . .	131
6.62.3.9	operator() . . . . .	131
6.62.3.10	operator() . . . . .	131
6.62.3.11	operator= . . . . .	131
6.62.3.12	valid . . . . .	131
6.63	FunctorCall< CallType, R, TYPELIST_0()> Struct Template Reference . . . . .	132
6.63.1	Member Typedef Documentation . . . . .	132
6.63.1.1	ParmsListType . . . . .	132
6.63.2	Member Function Documentation . . . . .	132
6.63.2.1	Call . . . . .	132
6.63.2.2	Call . . . . .	132
6.64	FunctorCall< CallType, R, TYPELIST_1(P1)> Struct Template Reference . . . . .	133
6.64.1	Member Typedef Documentation . . . . .	133
6.64.1.1	ParmsListType . . . . .	133
6.64.2	Member Function Documentation . . . . .	133
6.64.2.1	Call . . . . .	133
6.64.2.2	Call . . . . .	133
6.65	FunctorCall< CallType, R, TYPELIST_2(P1, P2)> Struct Template Reference . . . . .	134
6.65.1	Member Typedef Documentation . . . . .	134
6.65.1.1	ParmsListType . . . . .	134

6.65.2	Member Function Documentation	134
6.65.2.1	Call	134
6.65.2.2	Call	134
6.66	FunctorCall< CallType, R, TYPELIST_3(P1, P2, P3)> Struct Template Reference	135
6.66.1	Member Typedef Documentation	135
6.66.1.1	ParmsListType	135
6.66.2	Member Function Documentation	135
6.66.2.1	Call	135
6.66.2.2	Call	135
6.67	FunctorCall< CallType, R, TYPELIST_4(P1, P2, P3, P4)> Struct Template Reference	136
6.67.1	Member Typedef Documentation	136
6.67.1.1	ParmsListType	136
6.67.2	Member Function Documentation	136
6.67.2.1	Call	136
6.67.2.2	Call	136
6.68	FunctorCall< CallType, R, TYPELIST_5(P1, P2, P3, P4, P5)> Struct Template Reference	137
6.68.1	Member Typedef Documentation	137
6.68.1.1	ParmsListType	137
6.68.2	Member Function Documentation	137
6.68.2.1	Call	137
6.68.2.2	Call	137
6.69	FunctorCall< CallType, R, TYPELIST_6(P1, P2, P3, P4, P5, P6)> Struct Template Reference	138
6.69.1	Member Typedef Documentation	138
6.69.1.1	ParmsListType	138
6.69.2	Member Function Documentation	138
6.69.2.1	Call	138
6.69.2.2	Call	138
6.70	FunctorCall< CallType, R, TYPELIST_7(P1, P2, P3, P4, P5, P6, P7)> Struct Template Reference	139
6.70.1	Member Typedef Documentation	139
6.70.1.1	ParmsListType	139
6.70.2	Member Function Documentation	139
6.70.2.1	Call	139
6.70.2.2	Call	139
6.71	FunTraits< R(*)()> Struct Template Reference	140
6.71.1	Member Typedef Documentation	140
6.71.1.1	ObjType	140
6.71.1.2	ResultType	140
6.71.1.3	TypeListType	140
6.72	FunTraits< R(*)>(P1) Struct Template Reference	141
6.72.1	Member Typedef Documentation	141
6.72.1.1	ObjType	141

6.72.1.2	Parm1	141
6.72.1.3	ResultType	141
6.72.2	Member Function Documentation	141
6.72.2.1	TYPELIST_1	141
6.73	FunTraits< R(*) (P1, P2)> Struct Template Reference	142
6.73.1	Member Typedef Documentation	142
6.73.1.1	ObjType	142
6.73.1.2	Parm1	142
6.73.1.3	Parm2	142
6.73.1.4	ResultType	142
6.73.2	Member Function Documentation	142
6.73.2.1	TYPELIST_2	142
6.74	FunTraits< R(*) (P1, P2, P3)> Struct Template Reference	143
6.74.1	Member Typedef Documentation	143
6.74.1.1	ObjType	143
6.74.1.2	Parm1	143
6.74.1.3	Parm2	143
6.74.1.4	Parm3	143
6.74.1.5	ResultType	143
6.74.2	Member Function Documentation	143
6.74.2.1	TYPELIST_3	143
6.75	FunTraits< R(*) (P1, P2, P3, P4)> Struct Template Reference	144
6.75.1	Member Typedef Documentation	144
6.75.1.1	ObjType	144
6.75.1.2	Parm1	144
6.75.1.3	Parm2	144
6.75.1.4	Parm3	144
6.75.1.5	Parm4	144
6.75.1.6	ResultType	144
6.75.2	Member Function Documentation	144
6.75.2.1	TYPELIST_4	144
6.76	FunTraits< R(*) (P1, P2, P3, P4, P5)> Struct Template Reference	145
6.76.1	Member Typedef Documentation	145
6.76.1.1	ObjType	145
6.76.1.2	Parm1	145
6.76.1.3	Parm2	145
6.76.1.4	Parm3	145
6.76.1.5	Parm4	145
6.76.1.6	Parm5	145
6.76.1.7	ResultType	145
6.76.2	Member Function Documentation	145

6.76.2.1	TYPELIST_5	145
6.77	FunTraits< R(*) (P1, P2, P3, P4, P5, P6)> Struct Template Reference	146
6.77.1	Member Typedef Documentation	146
6.77.1.1	ObjType	146
6.77.1.2	Parm1	146
6.77.1.3	Parm2	146
6.77.1.4	Parm3	146
6.77.1.5	Parm4	146
6.77.1.6	Parm5	146
6.77.1.7	Parm6	146
6.77.1.8	ResultType	146
6.77.2	Member Function Documentation	146
6.77.2.1	TYPELIST_6	146
6.78	FunTraits< R(*) (P1, P2, P3, P4, P5, P6, P7)> Struct Template Reference	147
6.78.1	Member Typedef Documentation	147
6.78.1.1	ObjType	147
6.78.1.2	Parm1	147
6.78.1.3	Parm2	147
6.78.1.4	Parm3	147
6.78.1.5	Parm4	147
6.78.1.6	Parm5	147
6.78.1.7	Parm6	147
6.78.1.8	Parm7	147
6.78.1.9	ResultType	147
6.78.2	Member Function Documentation	147
6.78.2.1	TYPELIST_7	147
6.79	FunTraits< R(O::*)() const > Struct Template Reference	148
6.79.1	Member Typedef Documentation	148
6.79.1.1	ObjType	148
6.79.1.2	ResultType	148
6.79.1.3	TypeListType	148
6.80	FunTraits< R(O::*)()> Struct Template Reference	149
6.80.1	Member Typedef Documentation	149
6.80.1.1	ObjType	149
6.80.1.2	ResultType	149
6.80.1.3	TypeListType	149
6.81	FunTraits< R(O::*) (P1) const > Struct Template Reference	150
6.81.1	Member Typedef Documentation	150
6.81.1.1	ObjType	150
6.81.1.2	Parm1	150
6.81.1.3	ResultType	150

6.81.2	Member Function Documentation	150
6.81.2.1	TYPELIST_1	150
6.82	FunTraits< R(O::*)(P1)> Struct Template Reference	151
6.82.1	Member Typedef Documentation	151
6.82.1.1	ObjType	151
6.82.1.2	Parm1	151
6.82.1.3	ResultType	151
6.82.2	Member Function Documentation	151
6.82.2.1	TYPELIST_1	151
6.83	FunTraits< R(O::*)(P1, P2) const > Struct Template Reference	152
6.83.1	Member Typedef Documentation	152
6.83.1.1	ObjType	152
6.83.1.2	Parm1	152
6.83.1.3	Parm2	152
6.83.1.4	ResultType	152
6.83.2	Member Function Documentation	152
6.83.2.1	TYPELIST_2	152
6.84	FunTraits< R(O::*)(P1, P2)> Struct Template Reference	153
6.84.1	Member Typedef Documentation	153
6.84.1.1	ObjType	153
6.84.1.2	Parm1	153
6.84.1.3	Parm2	153
6.84.1.4	ResultType	153
6.84.2	Member Function Documentation	153
6.84.2.1	TYPELIST_2	153
6.85	FunTraits< R(O::*)(P1, P2, P3) const > Struct Template Reference	154
6.85.1	Member Typedef Documentation	154
6.85.1.1	ObjType	154
6.85.1.2	Parm1	154
6.85.1.3	Parm2	154
6.85.1.4	Parm3	154
6.85.1.5	ResultType	154
6.85.2	Member Function Documentation	154
6.85.2.1	TYPELIST_3	154
6.86	FunTraits< R(O::*)(P1, P2, P3)> Struct Template Reference	155
6.86.1	Member Typedef Documentation	155
6.86.1.1	ObjType	155
6.86.1.2	Parm1	155
6.86.1.3	Parm2	155
6.86.1.4	Parm3	155
6.86.1.5	ResultType	155

6.86.2	Member Function Documentation	155
6.86.2.1	TYPELIST_3	155
6.87	FunTraits< R(O::*)(P1, P2, P3, P4) const > Struct Template Reference	156
6.87.1	Member Typedef Documentation	156
6.87.1.1	ObjType	156
6.87.1.2	Parm1	156
6.87.1.3	Parm2	156
6.87.1.4	Parm3	156
6.87.1.5	Parm4	156
6.87.1.6	ResultType	156
6.87.2	Member Function Documentation	156
6.87.2.1	TYPELIST_4	156
6.88	FunTraits< R(O::*)(P1, P2, P3, P4)> Struct Template Reference	157
6.88.1	Member Typedef Documentation	157
6.88.1.1	ObjType	157
6.88.1.2	Parm1	157
6.88.1.3	Parm2	157
6.88.1.4	Parm3	157
6.88.1.5	Parm4	157
6.88.1.6	ResultType	157
6.88.2	Member Function Documentation	157
6.88.2.1	TYPELIST_4	157
6.89	FunTraits< R(O::*)(P1, P2, P3, P4, P5) const > Struct Template Reference	158
6.89.1	Member Typedef Documentation	158
6.89.1.1	Parm1	158
6.89.1.2	Parm2	158
6.89.1.3	Parm3	158
6.89.1.4	Parm4	158
6.89.1.5	Parm5	158
6.89.1.6	ResultType	158
6.89.2	Member Function Documentation	158
6.89.2.1	TYPELIST_5	158
6.90	FunTraits< R(O::*)(P1, P2, P3, P4, P5)> Struct Template Reference	159
6.90.1	Member Typedef Documentation	159
6.90.1.1	Parm1	159
6.90.1.2	Parm2	159
6.90.1.3	Parm3	159
6.90.1.4	Parm4	159
6.90.1.5	Parm5	159
6.90.1.6	ResultType	159
6.90.2	Member Function Documentation	159

6.90.2.1	TYPELIST_5	159
6.91	FunTraits< R(O::*)(P1, P2, P3, P4, P5, P6) const > Struct Template Reference	160
6.91.1	Member Typedef Documentation	160
6.91.1.1	Parm1	160
6.91.1.2	Parm2	160
6.91.1.3	Parm3	160
6.91.1.4	Parm4	160
6.91.1.5	Parm5	160
6.91.1.6	Parm6	160
6.91.1.7	ResultType	160
6.91.2	Member Function Documentation	160
6.91.2.1	TYPELIST_6	160
6.92	FunTraits< R(O::*)(P1, P2, P3, P4, P5, P6)> Struct Template Reference	161
6.92.1	Member Typedef Documentation	161
6.92.1.1	Parm1	161
6.92.1.2	Parm2	161
6.92.1.3	Parm3	161
6.92.1.4	Parm4	161
6.92.1.5	Parm5	161
6.92.1.6	Parm6	161
6.92.1.7	ResultType	161
6.92.2	Member Function Documentation	161
6.92.2.1	TYPELIST_6	161
6.93	FunTraits< R(O::*)(P1, P2, P3, P4, P5, P6, P7) const > Struct Template Reference	162
6.93.1	Member Typedef Documentation	162
6.93.1.1	Parm1	162
6.93.1.2	Parm2	162
6.93.1.3	Parm3	162
6.93.1.4	Parm4	162
6.93.1.5	Parm5	162
6.93.1.6	Parm6	162
6.93.1.7	Parm7	162
6.93.1.8	ResultType	162
6.93.2	Member Function Documentation	162
6.93.2.1	TYPELIST_7	162
6.94	FunTraits< R(O::*)(P1, P2, P3, P4, P5, P6, P7)> Struct Template Reference	163
6.94.1	Member Typedef Documentation	163
6.94.1.1	Parm1	163
6.94.1.2	Parm2	163
6.94.1.3	Parm3	163
6.94.1.4	Parm4	163

6.94.1.5	Parm5 . . . . .	163
6.94.1.6	Parm6 . . . . .	163
6.94.1.7	Parm7 . . . . .	163
6.94.1.8	ResultType . . . . .	163
6.94.2	Member Function Documentation . . . . .	163
6.94.2.1	TYPELIST_7 . . . . .	163
6.95	GeometryCollector Class Reference . . . . .	164
6.95.1	Constructor & Destructor Documentation . . . . .	164
6.95.1.1	GeometryCollector . . . . .	164
6.95.2	Member Function Documentation . . . . .	164
6.95.2.1	apply . . . . .	164
6.95.3	Member Data Documentation . . . . .	164
6.95.3.1	mGeomList . . . . .	164
6.96	GetElementType< T > Struct Template Reference . . . . .	165
6.96.1	Detailed Description . . . . .	165
6.96.2	Member Typedef Documentation . . . . .	165
6.96.2.1	value_type . . . . .	165
6.97	GroupVisitor Class Reference . . . . .	166
6.97.1	Constructor & Destructor Documentation . . . . .	166
6.97.1.1	GroupVisitor . . . . .	166
6.97.2	Member Function Documentation . . . . .	166
6.97.2.1	apply . . . . .	166
6.97.2.2	apply . . . . .	167
6.97.2.3	apply . . . . .	167
6.97.2.4	apply . . . . .	167
6.97.2.5	apply . . . . .	167
6.97.2.6	apply . . . . .	167
6.97.2.7	apply . . . . .	167
6.98	HotSpotDefinition Struct Reference . . . . .	168
6.98.1	Member Data Documentation . . . . .	168
6.98.1.1	mLocalRotation . . . . .	168
6.98.1.2	mLocalTranslation . . . . .	168
6.98.1.3	mName . . . . .	168
6.98.1.4	mParentName . . . . .	168
6.99	HotSpotFileHandler Class Reference . . . . .	169
6.99.1	Member Typedef Documentation . . . . .	170
6.99.1.1	HotSpotDefinitionVector . . . . .	170
6.99.2	Constructor & Destructor Documentation . . . . .	170
6.99.2.1	HotSpotFileHandler . . . . .	170
6.99.2.2	~HotSpotFileHandler . . . . .	170
6.99.3	Member Function Documentation . . . . .	170

6.99.3.1	characters . . . . .	170
6.99.3.2	endDocument . . . . .	170
6.99.3.3	endElement . . . . .	170
6.99.3.4	endPrefixMapping . . . . .	170
6.99.3.5	GetData . . . . .	170
6.99.3.6	ignorableWhitespace . . . . .	170
6.99.3.7	processingInstruction . . . . .	170
6.99.3.8	setDocumentLocator . . . . .	170
6.99.3.9	skippedEntity . . . . .	170
6.99.3.10	startDocument . . . . .	170
6.99.3.11	startElement . . . . .	170
6.99.3.12	startPrefixMapping . . . . .	170
6.99.4	Member Data Documentation . . . . .	170
6.99.4.1	DEFAULT_VALUE . . . . .	170
6.99.4.2	HEADING_VEC . . . . .	170
6.99.4.3	HOT_SPOT_NODE_NAME . . . . .	170
6.99.4.4	HOT_SPOT_PARENT_NODE_NAME . . . . .	170
6.99.4.5	LOCAL_ROTATION_NODE_NAME . . . . .	170
6.99.4.6	LOCAL_TRANSLATION_NODE_NAME . . . . .	170
6.99.4.7	NAME_ATTRIBUTE_NAME . . . . .	170
6.99.4.8	PITCH_VEC . . . . .	170
6.99.4.9	ROLL_VEC . . . . .	170
6.100	IdsFromTL< TL, i > Struct Template Reference . . . . .	171
6.100.1	Member Typedef Documentation . . . . .	171
6.100.1.1	Type . . . . .	171
6.101	IdsFromTL< NullType, i > Struct Template Reference . . . . .	172
6.101.1	Member Typedef Documentation . . . . .	172
6.101.1.1	Type . . . . .	172
6.102	IncomingCoordinateType Class Reference . . . . .	173
6.102.1	Member Data Documentation . . . . .	173
6.102.1.1	GEOCENTRIC . . . . .	173
6.102.1.2	GEODETIC . . . . .	173
6.102.1.3	UTM . . . . .	173
6.103	insert_back< _Container, _InputType > Class Template Reference . . . . .	174
6.103.1	Member Typedef Documentation . . . . .	174
6.103.1.1	container_type . . . . .	174
6.103.1.2	input_type . . . . .	174
6.103.2	Constructor & Destructor Documentation . . . . .	174
6.103.2.1	insert_back . . . . .	174
6.103.3	Member Function Documentation . . . . .	174
6.103.3.1	operator() . . . . .	174

6.103.4 Member Data Documentation	174
6.103.4.1 container	174
6.104 insert_back_no_duplicates< _Container > Class Template Reference	175
6.104.1 Member Typedef Documentation	175
6.104.1.1 container_type	175
6.104.1.2 reference	175
6.104.2 Constructor & Destructor Documentation	175
6.104.2.1 insert_back_no_duplicates	175
6.104.3 Member Function Documentation	175
6.104.3.1 operator()	175
6.104.4 Member Data Documentation	175
6.104.4.1 container	175
6.105 InstantiateH< NullType, Holder, i > Struct Template Reference	176
6.105.1 Constructor & Destructor Documentation	176
6.105.1.1 InstantiateH	176
6.106 InstantiateH< TypeList< T, U >, Holder, i > Struct Template Reference	177
6.106.1 Member Typedef Documentation	177
6.106.1.1 LeftBase	177
6.106.1.2 RightBase	177
6.106.2 Member Enumeration Documentation	177
6.106.2.1 "@8	177
6.106.3 Constructor & Destructor Documentation	177
6.106.3.1 InstantiateH	177
6.106.3.2 InstantiateH	177
6.106.3.3 InstantiateH	177
6.106.3.4 InstantiateH	177
6.107 InstantiateHAccessor< 0, InstantiateH< TypeList< T, U >, Holder, i >, i > Struct Template Reference	178
6.107.1 Member Typedef Documentation	178
6.107.1.1 Instance	178
6.107.1.2 TargetHolder	178
6.107.2 Member Function Documentation	178
6.107.2.1 Get	178
6.107.2.2 Get	178
6.108 InstantiateHAccessor< j, InstantiateH< TypeList< T, U >, Holder, i >, i > Struct Template Reference	179
6.108.1 Member Typedef Documentation	179
6.108.1.1 Instance	179
6.108.1.2 RightBase	179
6.108.1.3 TargetHolder	179
6.108.2 Member Function Documentation	179
6.108.2.1 Get	179

6.108.2.2	Get	179
6.109	Int2Type< v > Struct Template Reference	180
6.109.1	Member Enumeration Documentation	180
6.109.1.1	"@2	180
6.110	InternalLibraryHandle Class Reference	181
6.110.1	Constructor & Destructor Documentation	181
6.110.1.1	InternalLibraryHandle	181
6.110.1.2	~InternalLibraryHandle	181
6.110.2	Member Function Documentation	181
6.110.2.1	FindSymbol	181
6.110.2.2	GetHandle	181
6.110.2.3	GetLibName	181
6.110.2.4	LoadSharedLibrary	181
6.111	IsConst< _Type > Struct Template Reference	182
6.111.1	Member Enumeration Documentation	182
6.111.1.1	"@15	182
6.112	IsDelimiter Class Reference	183
6.112.1	Detailed Description	183
6.112.2	Constructor & Destructor Documentation	183
6.112.2.1	IsDelimiter	183
6.112.3	Member Function Documentation	183
6.112.3.1	operator()	183
6.113	IsIntType< T, i > Struct Template Reference	184
6.113.1	Member Enumeration Documentation	184
6.113.1.1	"@3	184
6.114	IsIntType< Int2Type< i >, i > Struct Template Reference	185
6.114.1	Member Enumeration Documentation	185
6.114.1.1	"@4	185
6.115	IsPointer< _Type > Struct Template Reference	186
6.115.1	Member Enumeration Documentation	186
6.115.1.1	"@9	186
6.116	IsReference< _Type > Struct Template Reference	187
6.116.1	Member Enumeration Documentation	187
6.116.1.1	"@12	187
6.117	IsSlash Class Reference	188
6.117.1	Detailed Description	188
6.117.2	Member Function Documentation	188
6.117.2.1	operator()	188
6.118	IsSpace Class Reference	189
6.118.1	Detailed Description	189
6.118.2	Constructor & Destructor Documentation	189

6.118.2.1	IsSpace . . . . .	189
6.118.3	Member Function Documentation . . . . .	189
6.118.3.1	GetLocale . . . . .	189
6.118.3.2	operator() . . . . .	189
6.118.4	Member Data Documentation . . . . .	189
6.118.4.1	DEFAULT_LOCALE_NAME . . . . .	189
6.119	KDTree< __K, _Val, _Acc, _Dist, _Cmp, _Alloc > Class Template Reference . . . . .	190
6.119.1	Member Typedef Documentation . . . . .	193
6.119.1.1	_Base . . . . .	193
6.119.1.2	_Base_const_ptr . . . . .	193
6.119.1.3	_Base_ptr . . . . .	193
6.119.1.4	_Link_const_type . . . . .	193
6.119.1.5	_Link_type . . . . .	193
6.119.1.6	_Node_compare_ . . . . .	193
6.119.1.7	_Region_ . . . . .	193
6.119.1.8	allocator_type . . . . .	193
6.119.1.9	const_iterator . . . . .	194
6.119.1.10	const_pointer . . . . .	194
6.119.1.11	const_reference . . . . .	194
6.119.1.12	const_reverse_iterator . . . . .	194
6.119.1.13	difference_type . . . . .	194
6.119.1.14	distance_type . . . . .	194
6.119.1.15	iterator . . . . .	194
6.119.1.16	pointer . . . . .	194
6.119.1.17	reference . . . . .	194
6.119.1.18	reverse_iterator . . . . .	194
6.119.1.19	size_type . . . . .	194
6.119.1.20	subvalue_type . . . . .	194
6.119.1.21	value_type . . . . .	194
6.119.2	Constructor & Destructor Documentation . . . . .	194
6.119.2.1	KDTree . . . . .	194
6.119.2.2	KDTree . . . . .	194
6.119.2.3	KDTree . . . . .	194
6.119.2.4	~KDTree . . . . .	194
6.119.3	Member Function Documentation . . . . .	194
6.119.3.1	_M_check_children . . . . .	194
6.119.3.2	_M_check_node . . . . .	194
6.119.3.3	_M_count_within_range . . . . .	194
6.119.3.4	_M_delete_node . . . . .	194
6.119.3.5	_M_empty_initialise . . . . .	194
6.119.3.6	_M_erase . . . . .	194

6.119.3.7	_M_erase_subtree	194
6.119.3.8	_M_find	194
6.119.3.9	_M_find_exact	194
6.119.3.10	_M_find_within_range	194
6.119.3.11	_M_get_erase_replacement	194
6.119.3.12	_M_get_j_max	194
6.119.3.13	_M_get_j_min	194
6.119.3.14	_M_get_leftmost	194
6.119.3.15	_M_get_rightmost	194
6.119.3.16	_M_get_root	194
6.119.3.17	_M_get_root	194
6.119.3.18	_M_insert	194
6.119.3.19	_M_insert_left	194
6.119.3.20	_M_insert_right	194
6.119.3.21	_M_matches_node	194
6.119.3.22	_M_matches_node_in_d	194
6.119.3.23	_M_matches_node_in_other_ds	194
6.119.3.24	_M_new_node	194
6.119.3.25	_M_optimise	194
6.119.3.26	_M_set_leftmost	194
6.119.3.27	_M_set_rightmost	194
6.119.3.28	_M_set_root	194
6.119.3.29	_M_visit_within_range	194
6.119.3.30	_S_is_leaf	194
6.119.3.31	_S_left	194
6.119.3.32	_S_left	194
6.119.3.33	_S_maximum	194
6.119.3.34	_S_minimum	194
6.119.3.35	_S_parent	194
6.119.3.36	_S_parent	194
6.119.3.37	_S_right	194
6.119.3.38	_S_right	194
6.119.3.39	_S_set_left	194
6.119.3.40	_S_set_parent	194
6.119.3.41	_S_set_right	194
6.119.3.42	_S_value	194
6.119.3.43	_S_value	194
6.119.3.44	begin	194
6.119.3.45	check_tree	194
6.119.3.46	clear	194
6.119.3.47	count_within_range	194

6.119.3.48	count_within_range	194
6.119.3.49	efficient_replace_and_optimize	194
6.119.3.50	empty	194
6.119.3.51	end	194
6.119.3.52	erase	194
6.119.3.53	erase	194
6.119.3.54	erase_exact	194
6.119.3.55	find	194
6.119.3.56	find_exact	194
6.119.3.57	find_nearest	194
6.119.3.58	find_nearest	194
6.119.3.59	find_nearest_if	194
6.119.3.60	find_within_range	194
6.119.3.61	find_within_range	194
6.119.3.62	find_within_range_iterative	194
6.119.3.63	get_allocator	194
6.119.3.64	insert	195
6.119.3.65	insert	195
6.119.3.66	insert	195
6.119.3.67	insert	195
6.119.3.68	insert	195
6.119.3.69	max_size	195
6.119.3.70	operator=	195
6.119.3.71	optimize	195
6.119.3.72	rbegin	195
6.119.3.73	rend	195
6.119.3.74	size	195
6.119.3.75	value_acc	195
6.119.3.76	value_comp	195
6.119.3.77	value_distance	195
6.119.3.78	value_distance	195
6.119.3.79	visit_within_range	195
6.119.3.80	visit_within_range	195
6.119.4	Member Data Documentation	195
6.119.4.1	_M_acc	195
6.119.4.2	_M_cmp	195
6.119.4.3	_M_count	195
6.119.4.4	_M_dist	195
6.119.4.5	_M_header	195
6.119.4.6	_M_root	195
6.120	KeyFrameDecoder< RecordableT, FrameDataT > Class Template Reference	196

6.120.1 Detailed Description . . . . .	196
6.120.2 Member Typedef Documentation . . . . .	196
6.120.2.1 FrameDataPtrContainer . . . . .	196
6.120.2.2 FrameDataType . . . . .	197
6.120.2.3 KeyFrame . . . . .	197
6.120.2.4 KeyFrameContainer . . . . .	197
6.120.2.5 RecordablePtrContainer . . . . .	197
6.120.2.6 RecordableType . . . . .	197
6.120.3 Constructor & Destructor Documentation . . . . .	197
6.120.3.1 KeyFrameDecoder . . . . .	197
6.120.3.2 ~KeyFrameDecoder . . . . .	197
6.120.4 Member Function Documentation . . . . .	197
6.120.4.1 DecodeFrameStamp . . . . .	197
6.120.4.2 DecodeSourceData . . . . .	197
6.120.4.3 Walk . . . . .	197
6.121 Length< NullType > Struct Template Reference . . . . .	198
6.121.1 Member Enumeration Documentation . . . . .	198
6.121.1.1 "@6 . . . . .	198
6.122 Length< TypeList< T, U > > Struct Template Reference . . . . .	199
6.122.1 Member Enumeration Documentation . . . . .	199
6.122.1.1 "@7 . . . . .	199
6.123 LibraryHandle Class Reference . . . . .	200
6.123.1 Detailed Description . . . . .	200
6.123.2 Member Typedef Documentation . . . . .	200
6.123.2.1 HANDLE . . . . .	200
6.123.2.2 SYMBOL_ADDRESS . . . . .	200
6.123.3 Constructor & Destructor Documentation . . . . .	200
6.123.3.1 LibraryHandle . . . . .	200
6.123.3.2 ~LibraryHandle . . . . .	200
6.123.4 Member Function Documentation . . . . .	200
6.123.4.1 FindSymbol . . . . .	200
6.123.4.2 GetHandle . . . . .	200
6.123.4.3 GetLibName . . . . .	200
6.123.4.4 IsShuttingDown . . . . .	201
6.123.4.5 release . . . . .	201
6.124 LibrarySharingManager Class Reference . . . . .	202
6.124.1 Detailed Description . . . . .	202
6.124.2 Member Function Documentation . . . . .	202
6.124.2.1 AddToSearchPath . . . . .	202
6.124.2.2 ClearSearchPath . . . . .	203
6.124.2.3 FindLibraryInSearchPath . . . . .	203

6.124.2.4	GetInstance . . . . .	203
6.124.2.5	GetPlatformIndependentLibraryName . . . . .	203
6.124.2.6	GetPlatformSpecificLibraryName . . . . .	203
6.124.2.7	GetSearchPath . . . . .	203
6.124.2.8	LoadSharedLibrary . . . . .	203
6.124.2.9	RemoveFromSearchPath . . . . .	203
6.125	LocalCoordinateType Class Reference . . . . .	204
6.125.1	Member Data Documentation . . . . .	204
6.125.1.1	CARTESIAN . . . . .	204
6.125.1.2	CARTESIAN_FLAT_EARTH . . . . .	204
6.125.1.3	CARTESIAN_UTM . . . . .	204
6.125.1.4	GLOBE . . . . .	204
6.126	Log Class Reference . . . . .	205
6.126.1	Detailed Description . . . . .	206
6.126.2	Member Enumeration Documentation . . . . .	206
6.126.2.1	LogMessageType . . . . .	206
6.126.2.2	OutputStreamOptions . . . . .	206
6.126.3	Constructor & Destructor Documentation . . . . .	206
6.126.3.1	Log . . . . .	206
6.126.3.2	~Log . . . . .	206
6.126.4	Member Function Documentation . . . . .	206
6.126.4.1	GetInstance . . . . .	206
6.126.4.2	GetInstance . . . . .	206
6.126.4.3	GetLogLevel . . . . .	206
6.126.4.4	GetLogLevelForString . . . . .	207
6.126.4.5	GetLogLevelString . . . . .	207
6.126.4.6	GetName . . . . .	207
6.126.4.7	GetOutputStreamBit . . . . .	207
6.126.4.8	IsLevelEnabled . . . . .	207
6.126.4.9	LogHorizRule . . . . .	207
6.126.4.10	LogMessage . . . . .	207
6.126.4.11	LogMessage . . . . .	207
6.126.4.12	LogMessage . . . . .	207
6.126.4.13	LogMessage . . . . .	208
6.126.4.14	LogMessage . . . . .	208
6.126.4.15	SetLogLevel . . . . .	208
6.126.4.16	SetOutputStreamBit . . . . .	208
6.127	LogFile Class Reference . . . . .	209
6.127.1	Member Function Documentation . . . . .	209
6.127.1.1	GetFileName . . . . .	209
6.127.1.2	GetTitle . . . . .	209

6.127.1.3	SetFileName	209
6.127.1.4	SetTitle	209
6.128	LogImpl Struct Reference	210
6.128.1	Constructor & Destructor Documentation	210
6.128.1.1	LogImpl	210
6.128.2	Member Data Documentation	210
6.128.2.1	mDefaultName	210
6.128.2.2	mName	210
6.128.2.3	mOutputStreamBit	210
6.129	LogManager Class Reference	211
6.129.1	Constructor & Destructor Documentation	211
6.129.1.1	LogManager	211
6.129.1.2	~LogManager	211
6.129.2	Member Function Documentation	211
6.129.2.1	AddInstance	211
6.129.2.2	EndFile	211
6.129.2.3	GetInstance	211
6.129.2.4	OpenFile	211
6.129.2.5	TimeTag	211
6.129.3	Member Data Documentation	211
6.129.3.1	logFile	211
6.129.3.2	mMutex	211
6.130	MatrixUtil Class Reference	212
6.130.1	Detailed Description	212
6.130.2	Member Function Documentation	212
6.130.2.1	ClampUnity	212
6.130.2.2	GetRow3	213
6.130.2.3	GetRow4	213
6.130.2.4	HprToMatrix	213
6.130.2.5	MatrixToHpr	213
6.130.2.6	MatrixToHprAndPosition	213
6.130.2.7	PositionAndHprToMatrix	213
6.130.2.8	Print	213
6.130.2.9	Print	213
6.130.2.10	Print	213
6.130.2.11	SetRow	214
6.130.2.12	SetRow	214
6.130.2.13	TransformVec3	214
6.130.2.14	TransformVec3	214
6.130.2.15	Transpose	214
6.131	MergeParmsH< i, BoundPTL, UnboundPTL, IdsTL, TL > Struct Template Reference	215

6.131.1 Member Typedef Documentation . . . . .	215
6.131.1.1 ResultType . . . . .	215
6.131.2 Member Function Documentation . . . . .	215
6.131.2.1 MergeParms . . . . .	215
6.132 MergeParmsH< 0, BoundPTL, UnboundPTL, dtUtil::NullType, TL > Struct Template Reference . .	216
6.132.1 Member Function Documentation . . . . .	216
6.132.1.1 MergeParms . . . . .	216
6.133 MergeParmsH< i, BoundPTL, UnboundPTL, IdsTL, dtUtil::NullType > Struct Template Reference .	217
6.133.1 Member Function Documentation . . . . .	217
6.133.1.1 MergeParms . . . . .	217
6.134 NodeCollector Class Reference . . . . .	218
6.134.1 Detailed Description . . . . .	221
6.134.2 Member Typedef Documentation . . . . .	221
6.134.2.1 GeodeNodeMap . . . . .	221
6.134.2.2 GroupNodeMap . . . . .	221
6.134.2.3 LODNodeMap . . . . .	221
6.134.2.4 MatrixTransformNodeMap . . . . .	221
6.134.2.5 MultiSwitchNodeMap . . . . .	221
6.134.2.6 NodeFlag . . . . .	221
6.134.2.7 SwitchNodeMap . . . . .	221
6.134.2.8 TransformNodeMap . . . . .	221
6.134.3 Constructor & Destructor Documentation . . . . .	221
6.134.3.1 NodeCollector . . . . .	221
6.134.3.2 NodeCollector . . . . .	221
6.134.3.3 ~NodeCollector . . . . .	222
6.134.4 Member Function Documentation . . . . .	222
6.134.4.1 AddDOFTransform . . . . .	222
6.134.4.2 AddGeode . . . . .	222
6.134.4.3 AddGroup . . . . .	222
6.134.4.4 AddLOD . . . . .	222
6.134.4.5 AddMatrixTransform . . . . .	222
6.134.4.6 AddMultiSwitch . . . . .	222
6.134.4.7 AddSwitch . . . . .	222
6.134.4.8 ClearAll . . . . .	222
6.134.4.9 CollectNodes . . . . .	223
6.134.4.10 GetDOFTransform . . . . .	223
6.134.4.11 GetDOFTransform . . . . .	223
6.134.4.12 GetGeode . . . . .	223
6.134.4.13 GetGeode . . . . .	223
6.134.4.14 GetGeodeNodeMap . . . . .	223
6.134.4.15 GetGeodeNodeMap . . . . .	223

6.134.4.16	GetGroup	223
6.134.4.17	GetGroup	224
6.134.4.18	GetGroupNodeMap	224
6.134.4.19	GetGroupNodeMap	224
6.134.4.20	GetLOD	224
6.134.4.21	GetLOD	224
6.134.4.22	GetLODNodeMap	224
6.134.4.23	GetLODNodeMap	224
6.134.4.24	GetMatrixTransform	224
6.134.4.25	GetMatrixTransform	224
6.134.4.26	GetMatrixTransformNodeMap	224
6.134.4.27	GetMatrixTransformNodeMap	224
6.134.4.28	GetMultiSwitch	225
6.134.4.29	GetMultiSwitch	225
6.134.4.30	GetMultiSwitchNodeMap	225
6.134.4.31	GetMultiSwitchNodeMap	225
6.134.4.32	GetSwitch	225
6.134.4.33	GetSwitch	225
6.134.4.34	GetSwitchNodeMap	225
6.134.4.35	GetSwitchNodeMap	225
6.134.4.36	GetTransformNodeMap	225
6.134.4.37	GetTransformNodeMap	225
6.134.4.38	RemoveDOFTransform	225
6.134.4.39	RemoveGeode	226
6.134.4.40	RemoveGroup	226
6.134.4.41	RemoveLOD	226
6.134.4.42	RemoveMatrixTransform	226
6.134.4.43	RemoveMultiSwitch	226
6.134.4.44	RemoveSwitch	226
6.134.5	Member Data Documentation	226
6.134.5.1	AllNodeTypes	226
6.134.5.2	DOFTransformFlag	226
6.134.5.3	GeodeFlag	226
6.134.5.4	GroupFlag	226
6.134.5.5	LODFlag	227
6.134.5.6	MatrixTransformFlag	227
6.134.5.7	MultiSwitchFlag	227
6.134.5.8	SwitchFlag	227
6.135	NodePrintOut Class Reference	228
6.135.1	Detailed Description	228
6.135.2	Constructor & Destructor Documentation	228

6.135.2.1	NodePrintOut . . . . .	228
6.135.2.2	~NodePrintOut . . . . .	228
6.135.3	Member Function Documentation . . . . .	228
6.135.3.1	Analyze . . . . .	228
6.135.3.2	AnalyzeGeode . . . . .	228
6.135.3.3	AnalyzePrimSet . . . . .	229
6.135.3.4	CollectNodeData . . . . .	229
6.135.3.5	GetFileOutput . . . . .	229
6.135.3.6	PrintNodeToOSGFile . . . . .	229
6.135.3.7	PrintNodeToOSGFile . . . . .	229
6.136	Noise1< Real, Vector > Class Template Reference . . . . .	230
6.136.1	Detailed Description . . . . .	230
6.136.2	Constructor & Destructor Documentation . . . . .	230
6.136.2.1	Noise1 . . . . .	230
6.136.2.2	~Noise1 . . . . .	230
6.136.3	Member Function Documentation . . . . .	230
6.136.3.1	GetNoise . . . . .	230
6.136.3.2	Reseed . . . . .	230
6.137	Noise2< Real, Vector > Class Template Reference . . . . .	231
6.137.1	Detailed Description . . . . .	231
6.137.2	Constructor & Destructor Documentation . . . . .	231
6.137.2.1	Noise2 . . . . .	231
6.137.2.2	~Noise2 . . . . .	231
6.137.3	Member Function Documentation . . . . .	231
6.137.3.1	GetNoise . . . . .	231
6.137.3.2	Reseed . . . . .	231
6.138	Noise3< Real, Vector > Class Template Reference . . . . .	232
6.138.1	Detailed Description . . . . .	232
6.138.2	Constructor & Destructor Documentation . . . . .	232
6.138.2.1	Noise3 . . . . .	232
6.138.2.2	~Noise3 . . . . .	232
6.138.3	Member Function Documentation . . . . .	232
6.138.3.1	GetNoise . . . . .	232
6.138.3.2	Reseed . . . . .	232
6.139	NoiseTexture Class Reference . . . . .	233
6.139.1	Detailed Description . . . . .	233
6.139.2	Constructor & Destructor Documentation . . . . .	233
6.139.2.1	NoiseTexture . . . . .	233
6.139.2.2	NoiseTexture . . . . .	233
6.139.2.3	~NoiseTexture . . . . .	234
6.139.3	Member Function Documentation . . . . .	234

6.139.3.1	GetNoiseTexture . . . . .	234
6.139.3.2	MakeNoiseTexture . . . . .	234
6.139.3.3	SetAmplitude . . . . .	234
6.139.3.4	SetFrequency . . . . .	234
6.139.3.5	SetHeight . . . . .	234
6.139.3.6	SetOctaves . . . . .	234
6.139.3.7	SetPersistence . . . . .	234
6.139.3.8	SetSlices . . . . .	234
6.139.3.9	SetWidth . . . . .	235
6.140	NoLeakAlloc Class Reference . . . . .	236
6.140.1	Constructor & Destructor Documentation . . . . .	236
6.140.1.1	NoLeakAlloc . . . . .	236
6.140.1.2	~NoLeakAlloc . . . . .	236
6.140.2	Member Function Documentation . . . . .	236
6.140.2.1	disconnect . . . . .	236
6.140.2.2	get . . . . .	236
6.141	NotIntType< T, i > Struct Template Reference . . . . .	237
6.141.1	Member Enumeration Documentation . . . . .	237
6.141.1.1	"@5 . . . . .	237
6.142	NullType Class Reference . . . . .	238
6.143	ObjectFactory< UniqueIdTypeClass, BaseTypeClass, ItCmpClass > Class Template Reference . . . . .	239
6.143.1	Detailed Description . . . . .	239
6.143.2	Member Typedef Documentation . . . . .	240
6.143.2.1	BaseType . . . . .	240
6.143.2.2	createObjectFunc . . . . .	240
6.143.2.3	ItCmp . . . . .	240
6.143.2.4	ObjectMap . . . . .	240
6.143.2.5	ObjTypeItor . . . . .	240
6.143.2.6	ObjTypeItorConst . . . . .	240
6.143.2.7	UniqueIdType . . . . .	240
6.143.3	Constructor & Destructor Documentation . . . . .	240
6.143.3.1	ObjectFactory . . . . .	240
6.143.3.2	~ObjectFactory . . . . .	240
6.143.4	Member Function Documentation . . . . .	240
6.143.4.1	CreateObject . . . . .	240
6.143.4.2	GetMap . . . . .	240
6.143.4.3	GetSupportedTypes . . . . .	240
6.143.4.4	IsTypeSupported . . . . .	240
6.143.4.5	RegisterType . . . . .	240
6.143.4.6	RemoveType . . . . .	240
6.144	Packager Class Reference . . . . .	241

6.144.1 Detailed Description . . . . .	241
6.144.2 Constructor & Destructor Documentation . . . . .	241
6.144.2.1 Packager . . . . .	241
6.144.2.2 ~Packager . . . . .	241
6.144.3 Member Function Documentation . . . . .	242
6.144.3.1 AddFile . . . . .	242
6.144.3.2 ClosePackage . . . . .	242
6.144.3.3 FindPackDataForPath . . . . .	242
6.144.3.4 GetPackTree . . . . .	242
6.144.3.5 OpenPackage . . . . .	242
6.144.3.6 PackPackage . . . . .	242
6.144.3.7 RemoveFile . . . . .	242
6.144.3.8 UnpackPackage . . . . .	242
6.145 PackTreeData Struct Reference . . . . .	243
6.145.1 Member Data Documentation . . . . .	243
6.145.1.1 files . . . . .	243
6.145.1.2 folders . . . . .	243
6.145.1.3 isFromPack . . . . .	243
6.145.1.4 name . . . . .	243
6.145.1.5 parent . . . . .	243
6.145.1.6 seekPos . . . . .	243
6.145.1.7 source . . . . .	243
6.146 PolarDecomp Class Reference . . . . .	244
6.146.1 Detailed Description . . . . .	244
6.146.2 Member Function Documentation . . . . .	244
6.146.2.1 Decompose . . . . .	244
6.147 Predicate< j, IdsTL2 > Struct Template Reference . . . . .	245
6.147.1 Member Enumeration Documentation . . . . .	245
6.147.1.1 "@0 . . . . .	245
6.148 Predicate< j, dtUtil::NullType > Struct Template Reference . . . . .	246
6.148.1 Member Enumeration Documentation . . . . .	246
6.148.1.1 "@1 . . . . .	246
6.149 RefString Class Reference . . . . .	247
6.149.1 Detailed Description . . . . .	247
6.149.2 Constructor & Destructor Documentation . . . . .	247
6.149.2.1 RefString . . . . .	247
6.149.2.2 RefString . . . . .	247
6.149.2.3 RefString . . . . .	247
6.149.2.4 ~RefString . . . . .	247
6.149.3 Member Function Documentation . . . . .	247
6.149.3.1 c_str . . . . .	247

6.149.3.2	Get	247
6.149.3.3	GetSharedStringCount	247
6.149.3.4	operator const std::string &	248
6.149.3.5	operator!=	248
6.149.3.6	operator!=	248
6.149.3.7	operator!=	248
6.149.3.8	operator+	248
6.149.3.9	operator+	248
6.149.3.10	operator+	248
6.149.3.11	operator->	248
6.149.3.12	operator<	248
6.149.3.13	operator=	248
6.149.3.14	operator=	248
6.149.3.15	operator==	248
6.149.3.16	operator==	248
6.149.3.17	operator==	248
6.149.3.18	operator[]	248
6.150	ResourceLoader< ResourceDescriptor, Resource > Class Template Reference	249
6.150.1	Constructor & Destructor Documentation	249
6.150.1.1	~ResourceLoader	249
6.150.2	Member Function Documentation	249
6.150.2.1	FreeResource	249
6.150.2.2	LoadResource	249
6.151	ResourceManager< ResourceKey, Resource > Class Template Reference	250
6.151.1	Member Typedef Documentation	250
6.151.1.1	ResourceConstIterator	250
6.151.1.2	ResourceHandle	250
6.151.1.3	ResourceIterator	250
6.151.1.4	ResourceMap	250
6.151.2	Constructor & Destructor Documentation	250
6.151.2.1	ResourceManager	250
6.151.2.2	~ResourceManager	250
6.151.3	Member Function Documentation	250
6.151.3.1	AddResource	250
6.151.3.2	FreeAll	250
6.151.3.3	FreeResource	250
6.151.3.4	GetResource	250
6.151.3.5	GetResource	250
6.151.3.6	LoadResource	250
6.151.3.7	SetResourceLoader	250
6.152	SeamlessNoise Class Reference	251

6.152.1 Detailed Description . . . . .	251
6.152.2 Constructor & Destructor Documentation . . . . .	251
6.152.2.1 SeamlessNoise . . . . .	251
6.152.2.2 ~SeamlessNoise . . . . .	251
6.152.3 Member Function Documentation . . . . .	251
6.152.3.1 GetNoise . . . . .	251
6.152.3.2 Reseed . . . . .	251
6.152.3.3 SetRepeat . . . . .	251
6.153 SeekTypeEnum Class Reference . . . . .	252
6.153.1 Member Data Documentation . . . . .	252
6.153.1.1 CURRENT . . . . .	252
6.153.1.2 END . . . . .	252
6.153.1.3 SET . . . . .	252
6.154 Select< flag, T, U > Struct Template Reference . . . . .	253
6.154.1 Member Typedef Documentation . . . . .	253
6.154.1.1 Result . . . . .	253
6.155 Select< false, T, U > Struct Template Reference . . . . .	254
6.155.1 Member Typedef Documentation . . . . .	254
6.155.1.1 Result . . . . .	254
6.156 Serializer Struct Reference . . . . .	255
6.156.1 Detailed Description . . . . .	255
6.156.2 Member Function Documentation . . . . .	255
6.156.2.1 ToBool . . . . .	255
6.156.2.2 ToDouble . . . . .	255
6.156.2.3 ToFloat . . . . .	256
6.156.2.4 ToInt . . . . .	256
6.156.2.5 ToString . . . . .	256
6.157 squared_difference< _Tp, _Dist > Struct Template Reference . . . . .	257
6.157.1 Member Typedef Documentation . . . . .	257
6.157.1.1 distance_type . . . . .	257
6.157.2 Member Function Documentation . . . . .	257
6.157.2.1 operator() . . . . .	257
6.158 squared_difference_counted< _Tp, _Dist > Struct Template Reference . . . . .	258
6.158.1 Member Typedef Documentation . . . . .	258
6.158.1.1 distance_type . . . . .	258
6.158.2 Constructor & Destructor Documentation . . . . .	258
6.158.2.1 squared_difference_counted . . . . .	258
6.158.3 Member Function Documentation . . . . .	258
6.158.3.1 count . . . . .	258
6.158.3.2 operator() . . . . .	258
6.158.3.3 reset . . . . .	258

6.159 StateAttributeCollector Class Reference . . . . .	259
6.159.1 Member Typedef Documentation . . . . .	260
6.159.1.1 MaterialNodeMap . . . . .	260
6.159.1.2 ProgramNodeMap . . . . .	260
6.159.1.3 StateAttribFlag . . . . .	260
6.159.1.4 TextureNodeMap . . . . .	260
6.159.2 Constructor & Destructor Documentation . . . . .	260
6.159.2.1 StateAttributeCollector . . . . .	260
6.159.2.2 StateAttributeCollector . . . . .	260
6.159.2.3 ~StateAttributeCollector . . . . .	260
6.159.3 Member Function Documentation . . . . .	261
6.159.3.1 AddMaterial . . . . .	261
6.159.3.2 AddProgram . . . . .	261
6.159.3.3 AddTexture . . . . .	261
6.159.3.4 ClearAll . . . . .	261
6.159.3.5 CollectStateAttributes . . . . .	261
6.159.3.6 GetMaterial . . . . .	261
6.159.3.7 GetMaterial . . . . .	261
6.159.3.8 GetMaterialMap . . . . .	261
6.159.3.9 GetMaterialMap . . . . .	262
6.159.3.10 GetProgram . . . . .	262
6.159.3.11 GetProgram . . . . .	262
6.159.3.12 GetProgramMap . . . . .	262
6.159.3.13 GetProgramMap . . . . .	262
6.159.3.14 GetTexture . . . . .	262
6.159.3.15 GetTexture . . . . .	262
6.159.3.16 GetTextureMap . . . . .	262
6.159.3.17 GetTextureMap . . . . .	262
6.159.4 Member Data Documentation . . . . .	263
6.159.4.1 AllAttributes . . . . .	263
6.159.4.2 MaterialFlag . . . . .	263
6.159.4.3 ProgramFlag . . . . .	263
6.159.4.4 TextureFlag . . . . .	263
6.160 StateVisitor Class Reference . . . . .	264
6.160.1 Constructor & Destructor Documentation . . . . .	264
6.160.1.1 StateVisitor . . . . .	264
6.160.2 Member Function Documentation . . . . .	264
6.160.2.1 apply . . . . .	264
6.160.2.2 apply . . . . .	264
6.160.2.3 stateSetParser . . . . .	264
6.161 StringTokenizer< Pred > Class Template Reference . . . . .	265

6.161.1 Member Function Documentation . . . . .	265
6.161.1.1 tokenize . . . . .	265
6.162 StringToXMLConverter Class Reference . . . . .	266
6.162.1 Detailed Description . . . . .	266
6.162.2 Constructor & Destructor Documentation . . . . .	266
6.162.2.1 StringToXMLConverter . . . . .	266
6.162.2.2 ~StringToXMLConverter . . . . .	266
6.162.3 Member Function Documentation . . . . .	266
6.162.3.1 ToXmlString . . . . .	266
6.163 TailAt< InstantiateH< TypeList< T, U >, Holder, i >, 0, i > Struct Template Reference . . . . .	267
6.163.1 Member Typedef Documentation . . . . .	267
6.163.1.1 Result . . . . .	267
6.164 TailAt< InstantiateH< TypeList< T, U >, Holder, i >, j, i > Struct Template Reference . . . . .	268
6.164.1 Member Typedef Documentation . . . . .	268
6.164.1.1 Result . . . . .	268
6.165 TangentSpaceVisitor Class Reference . . . . .	269
6.165.1 Detailed Description . . . . .	269
6.165.2 Constructor & Destructor Documentation . . . . .	269
6.165.2.1 TangentSpaceVisitor . . . . .	269
6.165.3 Member Function Documentation . . . . .	269
6.165.3.1 apply . . . . .	269
6.166 Temp< TL2, i, IdsTL2, Predicate > Struct Template Reference . . . . .	270
6.166.1 Member Typedef Documentation . . . . .	270
6.166.1.1 Result . . . . .	270
6.167 Temp< dtUtil::NullType, i, dtUtil::NullType, Predicate > Struct Template Reference . . . . .	271
6.167.1 Member Typedef Documentation . . . . .	271
6.167.1.1 Result . . . . .	271
6.168 Temp< dtUtil::NullType, i, IdsTL2, Predicate > Struct Template Reference . . . . .	272
6.168.1 Member Typedef Documentation . . . . .	272
6.168.1.1 Result . . . . .	272
6.169 Temp< TL2, 0, dtUtil::NullType, dtUtil::IsIntType > Struct Template Reference . . . . .	273
6.169.1 Member Typedef Documentation . . . . .	273
6.169.1.1 Result . . . . .	273
6.170 Temp< TL2, 0, dtUtil::NullType, dtUtil::NotIntType > Struct Template Reference . . . . .	274
6.170.1 Member Typedef Documentation . . . . .	274
6.170.1.1 Result . . . . .	274
6.171 Temp< TL2, 0, typename dtUtil::IdsFromTL< TL2 >::Type, dtUtil::IsIntType > Struct Template Reference . . . . .	275
6.171.1 Member Typedef Documentation . . . . .	275
6.171.1.1 Result . . . . .	275
6.172 Temp< TL2, 0, typename dtUtil::IdsFromTL< TL2 >::Type, dtUtil::NotIntType > Struct Template Reference . . . . .	276

6.172.1 Member Typedef Documentation . . . . .	276
6.172.1.1 Result . . . . .	276
6.173 Temp< TL2, i, dtUtil::NullType, Predicate > Struct Template Reference . . . . .	277
6.173.1 Member Typedef Documentation . . . . .	277
6.173.1.1 Result . . . . .	277
6.174 TimeFormat Class Reference . . . . .	278
6.174.1 Detailed Description . . . . .	278
6.174.2 Constructor & Destructor Documentation . . . . .	278
6.174.2.1 ~TimeFormat . . . . .	278
6.174.3 Member Data Documentation . . . . .	278
6.174.3.1 CALENDAR_DATE_AND_TIME_FORMAT . . . . .	278
6.174.3.2 CALENDAR_DATE_FORMAT . . . . .	278
6.174.3.3 CLOCK_TIME_12_HOUR_FORMAT . . . . .	278
6.174.3.4 CLOCK_TIME_24_HOUR_FORMAT . . . . .	278
6.174.3.5 LEXICAL_DATE_FORMAT . . . . .	278
6.174.3.6 LOCAL_DATE_AND_TIME_FORMAT . . . . .	278
6.174.3.7 LOCAL_DATE_FORMAT . . . . .	278
6.174.3.8 ORDINAL_DATE_FORMAT . . . . .	278
6.174.3.9 WEEK_DATE_FORMAT . . . . .	278
6.175 TimeOrigin Class Reference . . . . .	279
6.175.1 Detailed Description . . . . .	279
6.175.2 Constructor & Destructor Documentation . . . . .	279
6.175.2.1 ~TimeOrigin . . . . .	279
6.175.3 Member Data Documentation . . . . .	279
6.175.3.1 GMT_TIME . . . . .	279
6.175.3.2 LOCAL_TIME . . . . .	279
6.176 TimeType Class Reference . . . . .	280
6.176.1 Detailed Description . . . . .	280
6.176.2 Constructor & Destructor Documentation . . . . .	280
6.176.2.1 ~TimeType . . . . .	280
6.176.3 Member Data Documentation . . . . .	280
6.176.3.1 CLOCK_TIME . . . . .	280
6.176.3.2 SCENARIO_TIME . . . . .	280
6.176.3.3 SIMULATION_TIME . . . . .	280
6.176.3.4 TIME_STAMP . . . . .	280
6.176.3.5 TIME_TYPE_OTHER . . . . .	280
6.176.3.6 TRIP_TIME . . . . .	280
6.177 ToLowerClass Class Reference . . . . .	281
6.177.1 Constructor & Destructor Documentation . . . . .	281
6.177.1.1 ToLowerClass . . . . .	281
6.177.2 Member Function Documentation . . . . .	281

6.177.2.1	operator()	281
6.178	Transformation< T > Class Template Reference	282
6.178.1	Detailed Description	282
6.178.2	Member Function Documentation	282
6.178.2.1	operator()	282
6.179	tree< T > Class Template Reference	283
6.179.1	Member Typedef Documentation	284
6.179.1.1	const_iterator	284
6.179.1.2	iterator	284
6.179.2	Constructor & Destructor Documentation	284
6.179.2.1	tree	284
6.179.2.2	tree	284
6.179.2.3	tree	284
6.179.2.4	~tree	284
6.179.3	Member Function Documentation	284
6.179.3.1	begin	284
6.179.3.2	begin	284
6.179.3.3	begin	284
6.179.3.4	clear	284
6.179.3.5	copy_tree	284
6.179.3.6	data	284
6.179.3.7	data	284
6.179.3.8	data	284
6.179.3.9	end	285
6.179.3.10	erase	285
6.179.3.11	find	285
6.179.3.12	find	285
6.179.3.13	find	285
6.179.3.14	find	285
6.179.3.15	get_tree_iterator	285
6.179.3.16	get_tree_iterator	285
6.179.3.17	in	285
6.179.3.18	in	285
6.179.3.19	in	285
6.179.3.20	insert	285
6.179.3.21	insert	285
6.179.3.22	insert	285
6.179.3.23	level	285
6.179.3.24	level	285
6.179.3.25	next	285
6.179.3.26	operator*	285

6.179.3.27	operator*	285
6.179.3.28	operator=	285
6.179.3.29	operator==	285
6.179.3.30	operator[]	285
6.179.3.31	operator[]	285
6.179.3.32	out	286
6.179.3.33	out	286
6.179.3.34	out	286
6.179.3.35	prev	286
6.179.3.36	push_back	286
6.179.3.37	push_front	286
6.179.3.38	reinsert	286
6.179.3.39	reinsert	286
6.179.3.40	remove	286
6.179.3.41	size	286
6.179.3.42	size	286
6.179.3.43	tree_find_breadth	286
6.179.3.44	tree_find_breadth	286
6.179.3.45	tree_find_breadth	286
6.179.3.46	tree_find_breadth	286
6.179.3.47	tree_find_depth	286
6.179.3.48	tree_find_depth	286
6.179.3.49	tree_find_depth	286
6.179.3.50	tree_find_depth	286
6.180	tree_iterator< T > Class Template Reference	287
6.180.1	Member Typedef Documentation	288
6.180.1.1	const_iterator	288
6.180.1.2	iterator	288
6.180.2	Constructor & Destructor Documentation	288
6.180.2.1	tree_iterator	288
6.180.2.2	tree_iterator	288
6.180.2.3	tree_iterator	288
6.180.2.4	tree_iterator	288
6.180.2.5	~tree_iterator	288
6.180.3	Member Function Documentation	288
6.180.3.1	begin	288
6.180.3.2	begin	288
6.180.3.3	clear_children	288
6.180.3.4	clear_tree	288
6.180.3.5	data	288
6.180.3.6	data	288

6.180.3.7	data	288
6.180.3.8	end	289
6.180.3.9	end_iterator	289
6.180.3.10	find	289
6.180.3.11	find	289
6.180.3.12	find	289
6.180.3.13	find	289
6.180.3.14	in	289
6.180.3.15	in	289
6.180.3.16	insert	289
6.180.3.17	insert	289
6.180.3.18	level	289
6.180.3.19	next	289
6.180.3.20	operator!=	289
6.180.3.21	operator*	289
6.180.3.22	operator*	289
6.180.3.23	operator++	289
6.180.3.24	operator++	289
6.180.3.25	operator--	289
6.180.3.26	operator->	289
6.180.3.27	operator->	289
6.180.3.28	operator=	289
6.180.3.29	operator=	289
6.180.3.30	operator=	289
6.180.3.31	operator==	289
6.180.3.32	operator[]	289
6.180.3.33	operator[]	290
6.180.3.34	out	290
6.180.3.35	out	290
6.180.3.36	push_back	290
6.180.3.37	push_front	290
6.180.3.38	reinsert	290
6.180.3.39	reinsert	290
6.180.3.40	remove	290
6.180.3.41	size	290
6.180.3.42	tree_find_breadth	290
6.180.3.43	tree_find_breadth	290
6.180.3.44	tree_find_breadth	290
6.180.3.45	tree_find_breadth	290
6.180.3.46	tree_find_depth	290
6.180.3.47	tree_find_depth	290

6.180.3.48	tree_find_depth . . . . .	290
6.180.3.49	tree_find_depth . . . . .	290
6.180.3.50	tree_ptr . . . . .	290
6.180.3.51	tree_ref . . . . .	290
6.181	TupleHolder< T, i > Struct Template Reference . . . . .	291
6.181.1	Member Typedef Documentation . . . . .	291
6.181.1.1	StoredType . . . . .	291
6.181.1.2	Type . . . . .	291
6.181.2	Constructor & Destructor Documentation . . . . .	291
6.181.2.1	TupleHolder . . . . .	291
6.181.2.2	TupleHolder . . . . .	291
6.181.3	Member Function Documentation . . . . .	291
6.181.3.1	operator= . . . . .	291
6.181.4	Member Data Documentation . . . . .	291
6.181.4.1	value . . . . .	291
6.182	TypeAt< TypeList< T, U >, 0 > Struct Template Reference . . . . .	292
6.182.1	Member Typedef Documentation . . . . .	292
6.182.1.1	Result . . . . .	292
6.183	TypeAt< TypeList< T, U >, i > Struct Template Reference . . . . .	293
6.183.1	Member Typedef Documentation . . . . .	293
6.183.1.1	Result . . . . .	293
6.184	TypeAtNonStrict< TList, i, DefType > Struct Template Reference . . . . .	294
6.184.1	Member Typedef Documentation . . . . .	294
6.184.1.1	Result . . . . .	294
6.185	TypeAtNonStrict< TypeList< T, U >, 0, DefType > Struct Template Reference . . . . .	295
6.185.1	Member Typedef Documentation . . . . .	295
6.185.1.1	Result . . . . .	295
6.186	TypeAtNonStrict< TypeList< T, U >, i, DefType > Struct Template Reference . . . . .	296
6.186.1	Member Typedef Documentation . . . . .	296
6.186.1.1	Result . . . . .	296
6.187	TypeList< T, U > Struct Template Reference . . . . .	297
6.187.1	Member Typedef Documentation . . . . .	297
6.187.1.1	Head . . . . .	297
6.187.1.2	Tail . . . . .	297
6.188	TypeTraits< U > Struct Template Reference . . . . .	298
6.188.1	Member Typedef Documentation . . . . .	298
6.188.1.1	ConstRef . . . . .	298
6.188.1.2	NonConstNoRef . . . . .	298
6.188.1.3	NonConstRef . . . . .	298
6.189	TypeTraits< _Type > Struct Template Reference . . . . .	299
6.189.1	Member Typedef Documentation . . . . .	299

6.189.1.1	const_param_type	299
6.189.1.2	const_reference	299
6.189.1.3	const_return_type	299
6.189.1.4	param_type	299
6.189.1.5	pointer_type	299
6.189.1.6	reference	299
6.189.1.7	return_type	299
6.189.1.8	value_type	299
6.190	Unbound Struct Reference	300
6.190.1	Member Function Documentation	300
6.190.1.1	MergeParms	300
6.191	UnboundHelper< Incoming, IdsTL > Struct Template Reference	301
6.191.1	Member Typedef Documentation	301
6.191.1.1	IncomingTL	301
6.191.1.2	Parm1	301
6.191.1.3	Parm2	301
6.191.1.4	Parm3	301
6.191.1.5	Parm4	301
6.191.1.6	Parm5	301
6.191.1.7	UnboundTL	301
6.192	UnboundTL2< TL, IdsTL > Struct Template Reference	302
6.192.1	Member Typedef Documentation	302
6.192.1.1	Result	302
6.193	UTMPParameters Struct Reference	303
6.193.1	Constructor & Destructor Documentation	304
6.193.1.1	UTMPParameters	304
6.193.2	Member Function Documentation	304
6.193.2.1	CalcTransverseMercatorParameters	304
6.193.2.2	DENOM	304
6.193.2.3	SPHSN	304
6.193.2.4	SPHSR	304
6.193.2.5	SPHTMD	304
6.193.3	Member Data Documentation	304
6.193.3.1	TranMerc_a	304
6.193.3.2	TranMerc_ap	304
6.193.3.3	TranMerc_bp	304
6.193.3.4	TranMerc_cp	304
6.193.3.5	TranMerc_Delta_Easting	304
6.193.3.6	TranMerc_Delta_Northing	304
6.193.3.7	TranMerc_dp	304
6.193.3.8	TranMerc_ebs	304

6.193.3.9	TranMerc_ep	304
6.193.3.10	TranMerc_es	304
6.193.3.11	TranMerc_f	304
6.193.3.12	TranMerc_False_Easting	304
6.193.3.13	TranMerc_False_Northing	304
6.193.3.14	TranMerc_Origin_Lat	304
6.193.3.15	TranMerc_Origin_Long	304
6.193.3.16	TranMerc_Scale_Factor	304
6.194	VTable Struct Reference	305
6.194.1	Member Data Documentation	305
6.194.1.1	call_	305
6.194.1.2	clone_	305
6.194.1.3	destroy_	305
6.195	XercesErrorHandler Class Reference	306
6.195.1	Detailed Description	306
6.195.2	Constructor & Destructor Documentation	306
6.195.2.1	XercesErrorHandler	306
6.195.2.2	~XercesErrorHandler	306
6.195.3	Member Function Documentation	306
6.195.3.1	error	306
6.195.3.2	fatalError	306
6.195.3.3	resetErrors	306
6.195.3.4	warning	306
6.196	XercesParser Class Reference	307
6.196.1	Detailed Description	307
6.196.2	Constructor & Destructor Documentation	307
6.196.2.1	XercesParser	307
6.196.2.2	~XercesParser	307
6.196.3	Member Function Documentation	307
6.196.3.1	Parse	307
6.197	XercesWriter Class Reference	308
6.197.1	Detailed Description	308
6.197.2	Constructor & Destructor Documentation	308
6.197.2.1	XercesWriter	308
6.197.2.2	~XercesWriter	308
6.197.3	Member Function Documentation	308
6.197.3.1	CreateDocument	308
6.197.3.2	GetDocument	308
6.197.3.3	GetDocument	308
6.197.3.4	WriteFile	308
6.198	XMLStringConverter Class Reference	309

6.198.1 Detailed Description . . . . .	309
6.198.2 Constructor & Destructor Documentation . . . . .	309
6.198.2.1 XMLStringConverter . . . . .	309
6.198.2.2 ~XMLStringConverter . . . . .	309
6.198.3 Member Function Documentation . . . . .	309
6.198.3.1 c_str . . . . .	309
6.198.3.2 ToString . . . . .	309
<b>7 File Documentation</b>	<b>311</b>
7.1 barycentric.h File Reference . . . . .	311
7.2 bits.h File Reference . . . . .	312
7.3 boundingshapeutils.h File Reference . . . . .	313
7.4 breakoverride.h File Reference . . . . .	314
7.4.1 Define Documentation . . . . .	314
7.4.1.1 BREAK_OVERRIDE . . . . .	314
7.5 collectorutil.h File Reference . . . . .	315
7.6 command.h File Reference . . . . .	316
7.7 configproperties.h File Reference . . . . .	317
7.8 coordinates.cpp File Reference . . . . .	318
7.9 coordinates.h File Reference . . . . .	319
7.10 datastream.cpp File Reference . . . . .	321
7.11 datastream.h File Reference . . . . .	322
7.12 datetime.cpp File Reference . . . . .	323
7.13 datetime.h File Reference . . . . .	324
7.14 deprecationmgr.cpp File Reference . . . . .	325
7.15 deprecationmgr.h File Reference . . . . .	326
7.15.1 Define Documentation . . . . .	326
7.15.1.1 DEPRECATE . . . . .	326
7.16 dtutil.h File Reference . . . . .	327
7.17 enumeration.cpp File Reference . . . . .	328
7.18 enumeration.h File Reference . . . . .	329
7.18.1 Define Documentation . . . . .	329
7.18.1.1 DECLARE_ENUM . . . . .	329
7.18.1.2 IMPLEMENT_ENUM . . . . .	330
7.19 exception.cpp File Reference . . . . .	331
7.20 exception.h File Reference . . . . .	332
7.21 export.h File Reference . . . . .	333
7.21.1 Define Documentation . . . . .	333
7.21.1.1 DT_UTIL_EXPORT . . . . .	333
7.22 fileutils.cpp File Reference . . . . .	334
7.22.1 Define Documentation . . . . .	334

7.22.1.1	MAX_PATH	334
7.22.1.2	S_ISDIR	334
7.22.1.3	S_ISREG	334
7.22.1.4	stat64	334
7.23	fileutils.h File Reference	335
7.24	fractal.h File Reference	336
7.25	funbind.h File Reference	337
7.26	funcall.h File Reference	339
7.27	functor.h File Reference	340
7.27.1	Define Documentation	340
7.27.1.1	DoCall	340
7.28	funtraits.h File Reference	341
7.29	generic.h File Reference	342
7.29.1	Define Documentation	343
7.29.1.1	TYPELIST_0	343
7.29.1.2	TYPELIST_1	343
7.29.1.3	TYPELIST_2	343
7.29.1.4	TYPELIST_3	343
7.29.1.5	TYPELIST_4	343
7.29.1.6	TYPELIST_5	343
7.29.1.7	TYPELIST_6	343
7.29.1.8	TYPELIST_7	343
7.29.1.9	TYPELIST_8	343
7.30	geometrycollector.h File Reference	344
7.31	hotspotdefinition.h File Reference	345
7.32	hotspotxml.cpp File Reference	346
7.33	hotspotxml.h File Reference	347
7.34	kdtree.h File Reference	348
7.34.1	Detailed Description	349
7.34.2	Define Documentation	349
7.34.2.1	KDTREE_LIB_VERSION	349
7.34.2.2	KDTREE_VERSION	349
7.35	keyframedecoder.h File Reference	350
7.35.1	Detailed Description	350
7.36	librarysharingmanager.cpp File Reference	351
7.37	librarysharingmanager.h File Reference	352
7.38	log.cpp File Reference	353
7.39	log.h File Reference	354
7.39.1	Define Documentation	354
7.39.1.1	LOG_ALWAYS	354
7.39.1.2	LOG_DEBUG	354

7.39.1.3	LOG_ERROR	355
7.39.1.4	LOG_INFO	355
7.39.1.5	LOG_WARNING	355
7.39.1.6	LOGN_ALWAYS	355
7.39.1.7	LOGN_DEBUG	355
7.39.1.8	LOGN_ERROR	355
7.39.1.9	LOGN_INFO	355
7.39.1.10	LOGN_WARNING	355
7.40	macros.h File Reference	356
7.40.1	Define Documentation	356
7.40.1.1	BIT	356
7.40.1.2	DECLARE_MANAGEMENT_LAYER	356
7.40.1.3	DTUNREFERENCED_PARAMETER	356
7.40.1.4	IMPLEMENT_MANAGEMENT_LAYER	356
7.40.1.5	IS_A	356
7.40.1.6	UNSIGNED_BIT	356
7.40.1.7	UNSIGNED_INT_BIT	356
7.41	mainpage.h File Reference	357
7.41.1	Detailed Description	357
7.42	mathdefines.h File Reference	358
7.42.1	Define Documentation	359
7.42.1.1	RAND_MAX	359
7.43	matrixutil.cpp File Reference	360
7.44	matrixutil.h File Reference	361
7.45	mwin.h File Reference	362
7.46	nodecollector.cpp File Reference	363
7.47	nodecollector.h File Reference	364
7.48	nodeprintout.cpp File Reference	365
7.49	nodeprintout.h File Reference	366
7.50	noise1.h File Reference	367
7.51	noise2.h File Reference	368
7.52	noise3.h File Reference	369
7.53	noisetexture.cpp File Reference	370
7.54	noisetexture.h File Reference	371
7.55	noiseutility.h File Reference	372
7.56	objectfactory.h File Reference	373
7.57	packager.cpp File Reference	374
7.58	packager.h File Reference	375
7.59	polardecomp.cpp File Reference	376
7.60	polardecomp.h File Reference	377
7.61	precomp.cpp File Reference	378

7.61.1	Define Documentation . . . . .	378
7.61.1.1	DELTA_PCH . . . . .	378
7.62	refstring.cpp File Reference . . . . .	379
7.62.1	Define Documentation . . . . .	379
7.62.1.1	THREAD_SAFETY . . . . .	379
7.62.1.2	USE_TABLE . . . . .	379
7.63	refstring.h File Reference . . . . .	380
7.64	resourceloader.h File Reference . . . . .	381
7.65	resourcemanager.h File Reference . . . . .	382
7.66	seamlessnoise.cpp File Reference . . . . .	383
7.66.1	Define Documentation . . . . .	383
7.66.1.1	LERP . . . . .	383
7.67	seamlessnoise.h File Reference . . . . .	384
7.68	serializer.cpp File Reference . . . . .	385
7.69	serializer.h File Reference . . . . .	386
7.69.1	Detailed Description . . . . .	386
7.70	stateattributecollector.cpp File Reference . . . . .	387
7.71	stateattributecollector.h File Reference . . . . .	388
7.72	stringutils.cpp File Reference . . . . .	389
7.73	stringutils.h File Reference . . . . .	390
7.73.1	Detailed Description . . . . .	391
7.74	tangentspacevisitor.cpp File Reference . . . . .	392
7.75	tangentspacevisitor.h File Reference . . . . .	393
7.76	templateutility.h File Reference . . . . .	394
7.77	transformation.h File Reference . . . . .	395
7.78	tree.h File Reference . . . . .	396
7.78.1	Define Documentation . . . . .	396
7.78.1.1	NULL . . . . .	396
7.79	typetraits.h File Reference . . . . .	397
7.80	version.cpp File Reference . . . . .	398
7.80.1	Function Documentation . . . . .	398
7.80.1.1	Delta3DGetLibraryName . . . . .	398
7.80.1.2	Delta3DGetVersion . . . . .	398
7.81	version.h File Reference . . . . .	399
7.81.1	Define Documentation . . . . .	399
7.81.1.1	DELTA3D_VERSION_MAJOR . . . . .	399
7.81.1.2	DELTA3D_VERSION_MINOR . . . . .	399
7.81.1.3	DELTA3D_VERSION_PATCH . . . . .	399
7.81.2	Function Documentation . . . . .	399
7.81.2.1	Delta3DGetLibraryName . . . . .	399
7.81.2.2	Delta3DGetVersion . . . . .	399

---

7.82	xerceserrorhandler.cpp File Reference . . . . .	400
7.83	xerceserrorhandler.h File Reference . . . . .	401
	7.83.1 Detailed Description . . . . .	401
7.84	xercesparser.cpp File Reference . . . . .	402
7.85	xercesparser.h File Reference . . . . .	403
	7.85.1 Detailed Description . . . . .	403
7.86	xercesutils.cpp File Reference . . . . .	404
7.87	xercesutils.h File Reference . . . . .	405
7.88	xerceswriter.cpp File Reference . . . . .	406
7.89	xerceswriter.h File Reference . . . . .	407
	7.89.1 Detailed Description . . . . .	407

## Main Page

---

Delta3D is an Open Source engine which can be used for games, simulations, or other graphical applications.

The Delta3D framework exists as a number of modules, each sitting in its own library, enclosed within its own namespace. At the very core lies the dtCore library. This contains basic, low-level functionality which is mostly required for all 3D applications written in C++.

Around and alongside this sit other supporting libraries, such as **dtUtil** (p. 13) (containing reusable features which are useful for most applications), dtTerrain (for rendering terrain databases), dtGame, dtNet, etc.

Extensive online documentation is available from the Delta3D Docs section to help in using Delta3D.

The project's original reference guides generated by Doxygen from the source code may be viewed at the Delta3D API Documentation section.

To download source code, binaries, dependencies and sample datasets visit the Delta3D Downloads page.

For more about dependencies see the Delta3D Dependencies page.

The documentation you are looking at can be downloaded from [www.3draum.ch](http://www.3draum.ch).

Enjoy!



**Todo List**

---



**Deprecated List**

---



## Directory Documentation

---

### 4.1 src/dtUtil/ Directory Reference

#### Files

- file **coordinates.cpp**
- file **datastream.cpp**
- file **datetime.cpp**
- file **deprecationmgr.cpp**
- file **enumeration.cpp**
- file **exception.cpp**
- file **fileutils.cpp**
- file **hotspotxml.cpp**
- file **librarysharingmanager.cpp**
- file **log.cpp**
- file **matrixutil.cpp**
- file **nodecollector.cpp**
- file **nodeprintout.cpp**
- file **noisetexture.cpp**
- file **packager.cpp**
- file **polardecomp.cpp**
- file **precomp.cpp**
- file **refstring.cpp**
- file **seamlessnoise.cpp**
- file **serializer.cpp**
- file **stateattributecollector.cpp**
- file **stringutils.cpp**
- file **tangentspacevisitor.cpp**
- file **version.cpp**
- file **xerceserrorhandler.cpp**
- file **xercesparser.cpp**
- file **xercesutils.cpp**
- file **xerceswriter.cpp**

## 4.2 inc/dtUtil/ Directory Reference

### Files

- file **barycentric.h**
- file **bits.h**
- file **boundingshapeutils.h**
- file **breakoverride.h**
- file **collectorutil.h**
- file **command.h**
- file **configproperties.h**
- file **coordinates.h**
- file **datastream.h**
- file **datetime.h**
- file **deprecationmgr.h**
- file **dtutil.h**
- file **enumeration.h**
- file **exception.h**
- file **export.h**
- file **fileutils.h**
- file **fractal.h**
- file **funbind.h**
- file **funcall.h**
- file **functor.h**
- file **funtraits.h**
- file **generic.h**
- file **geometrycollector.h**
- file **hotspotdefinition.h**
- file **hotspotxml.h**
- file **kdtree.h**
  - *Defines the interface for the KDTree class.*
- file **keyframedecoder.h**
  - *Utility methods for using strings, often for XML purposes.*
- file **librarysharingmanager.h**
- file **log.h**
- file **macros.h**
- file **mainpage.h**
- file **mathdefines.h**
- file **matrixutil.h**
- file **mswin.h**
- file **nodecollector.h**
- file **nodeprintout.h**
- file **noise1.h**
- file **noise2.h**
- file **noise3.h**
- file **noisetexture.h**
- file **noiseutility.h**
- file **objectfactory.h**
- file **packager.h**
- file **polardecomp.h**
- file **refstring.h**
- file **resourceloader.h**
- file **resourcemanager.h**
- file **seamlessnoise.h**
- file **serializer.h**

*Utility functions for serialization.*

- file **stateattributecollector.h**
- file **stringutils.h**

*Utility methods for using strings.*

- file **tangentspacevisitor.h**
- file **templateutility.h**
- file **transformation.h**
- file **tree.h**
- file **typetraits.h**
- file **version.h**
- file **xerceserrorhandler.h**
- file **xercesparser.h**
- file **xercesutils.h**
- file **xerceswriter.h**

## 4.3 inc/ Directory Reference

### Directories

- directory **dtUtil**

## 4.4 src/ Directory Reference

### Directories

- directory dtUtil



# Namespace Documentation

---

## 5.1 dtUtil Namespace Reference

Contains generic, reusable features which are useful for most applications.

### Namespaces

- namespace **Bits**  
*Contains bit-wise operation functionality which makes using bits a little "bit" easier.*
- namespace **CollectorUtil**
- namespace **details**

### Classes

- class **\_Alloc\_base**
- class **\_Base\_iterator**
- struct **\_Bracket\_accessor**
- class **\_Iterator**
- struct **\_Node**
- struct **\_Node\_base**
- class **\_Node\_compare**
- struct **\_Region**
- struct **always\_true**
- struct **AppendTL**
- struct **AppendTL< NullType, T >**
- class **array\_remove**
- class **AttributeSearch**  
*Searches a Xerces XML Attribute list for names of interest.*
- class **BarycentricSpace**  
*Transforms Cartesian data points into Barycentric coordinate systems.*
- class **BaseExceptionType**
- class **Binder**
- struct **BoundHelper**
- class **BoundingBoxVisitor**
- struct **BoundTL2**
- struct **BreakOverride**  
*This macro should be used when deprecating (or removing) virtual functions.*
- struct **CallParms< TYPELIST\_0()>**
- struct **CallParms< TYPELIST\_1(P1)>**
- struct **CallParms< TYPELIST\_2(P1, P2)>**
- struct **CallParms< TYPELIST\_3(P1, P2, P3)>**

- struct **CallParms**< TYPELIST\_4(P1, P2, P3, P4)>
- struct **CallParms**< TYPELIST\_5(P1, P2, P3, P4, P5)>
- struct **CallParms**< TYPELIST\_6(P1, P2, P3, P4, P5, P6)>
- struct **CallParms**< TYPELIST\_7(P1, P2, P3, P4, P5, P6, P7)>
- class **Command**
  - An abstract class for all types which provide a uniform interface for executing Functors.*
- class **Command0**
  - A **Command** (p. 70) which does not need arguments and has no return value.*
- class **Command1**
  - A **Command** (p. 70) which uses one argument and has no return value.*
- class **Command2**
  - A **Command** (p. 70) which uses two arguments and has no return value.*
- class **ConfigProperties**
  - Interface for having configuration properties.*
- class **CoordinateConversionExceptionEnum**
- class **Coordinates**
- struct **CreatesTL**
- struct **CreatesTL**<-1,-1,-1,-1,-1,-1,-1,-1 >
- struct **CreateTL**
- struct **CreateTL**< NullType, NullType, NullType, NullType, NullType, NullType, NullType, NullType >
- class **DataStream**
- class **DataStreamException**
- class **DateTime**
- struct **DoNothing**
  - This odd functor is actually useful for supplying a default template parameter when a functors is expected.*
- struct **DoNothing0**
  - This odd functor is actually useful for supplying a default template parameter when a functors is expected.*
- class **EdgeStepFilter**
  - The **EdgeStepFilter** (p. 108) will return back 1.0, 0.0, or -1.0 depending if the supplied input value is greater than the high, in between high and low, or less than the low.*
- struct **EmptyType**
- class **Enumeration**
  - This class represents a type-safe enumeration pattern.*
- struct **EvaluateFunctor**
- struct **EvaluateFunctor**< FunctorType \*, ArgType, RetType >
- struct **EvaluateInvoke**
- class **Exception**
- class **FileExceptionEnum**
  - An enumeration of exception types that will be thrown by fileutils.*
- struct **FileInfo**
- class **FileUtils**
- struct **Filter2**
- class **Fractal**
  - Fractal** (p. 128): This class is made to complement the noise classes where as the noise class hashes vectors to floats between -1 to 1 this class can be used to make summations of the noise to use this class I recommend including **NoiseUtility.h** (p. 372) and using the appropriate typedefs.*

- class **Functor**
- struct **FunctorCall**< **CallType**, **R**, **TYPELIST\_0**()>
- struct **FunctorCall**< **CallType**, **R**, **TYPELIST\_1**(**P1**)>
- struct **FunctorCall**< **CallType**, **R**, **TYPELIST\_2**(**P1**, **P2**)>
- struct **FunctorCall**< **CallType**, **R**, **TYPELIST\_3**(**P1**, **P2**, **P3**)>
- struct **FunctorCall**< **CallType**, **R**, **TYPELIST\_4**(**P1**, **P2**, **P3**, **P4**)>
- struct **FunctorCall**< **CallType**, **R**, **TYPELIST\_5**(**P1**, **P2**, **P3**, **P4**, **P5**)>
- struct **FunctorCall**< **CallType**, **R**, **TYPELIST\_6**(**P1**, **P2**, **P3**, **P4**, **P5**, **P6**)>
- struct **FunctorCall**< **CallType**, **R**, **TYPELIST\_7**(**P1**, **P2**, **P3**, **P4**, **P5**, **P6**, **P7**)>
- struct **FunTraits**< **R**(\*)()>
- struct **FunTraits**< **R**(\*)(**P1**)>
- struct **FunTraits**< **R**(\*)(**P1**, **P2**)>
- struct **FunTraits**< **R**(\*)(**P1**, **P2**, **P3**)>
- struct **FunTraits**< **R**(\*)(**P1**, **P2**, **P3**, **P4**)>
- struct **FunTraits**< **R**(\*)(**P1**, **P2**, **P3**, **P4**, **P5**)>
- struct **FunTraits**< **R**(\*)(**P1**, **P2**, **P3**, **P4**, **P5**, **P6**)>
- struct **FunTraits**< **R**(\*)(**P1**, **P2**, **P3**, **P4**, **P5**, **P6**, **P7**)>
- struct **FunTraits**< **R**(O::\*)() const >
- struct **FunTraits**< **R**(O::\*)(P1) const >
- struct **FunTraits**< **R**(O::\*)(P1) const >
- struct **FunTraits**< **R**(O::\*)(P1, P2) const >
- struct **FunTraits**< **R**(O::\*)(P1, P2) const >
- struct **FunTraits**< **R**(O::\*)(P1, P2, P3) const >
- struct **FunTraits**< **R**(O::\*)(P1, P2, P3) const >
- struct **FunTraits**< **R**(O::\*)(P1, P2, P3, P4) const >
- struct **FunTraits**< **R**(O::\*)(P1, P2, P3, P4) const >
- struct **FunTraits**< **R**(O::\*)(P1, P2, P3, P4, P5) const >
- struct **FunTraits**< **R**(O::\*)(P1, P2, P3, P4, P5) const >
- struct **FunTraits**< **R**(O::\*)(P1, P2, P3, P4, P5, P6) const >
- struct **FunTraits**< **R**(O::\*)(P1, P2, P3, P4, P5, P6) const >
- struct **FunTraits**< **R**(O::\*)(P1, P2, P3, P4, P5, P6, P7) const >
- struct **FunTraits**< **R**(O::\*)(P1, P2, P3, P4, P5, P6, P7) const >
- class **GeometryCollector**
- class **GroupVisitor**
- struct **HotSpotDefinition**
- class **HotSpotFileHandler**
- struct **IdsFromTL**
- struct **IdsFromTL**< **NullType**, **i** >
- class **IncomingCoordinateType**
- class **insert\_back**
- class **insert\_back\_no\_duplicates**
- struct **InstantiateH**< **NullType**, **Holder**, **i** >
- struct **InstantiateH**< **TypeList**< **T**, **U** >, **Holder**, **i** >
- struct **InstantiateHAccessor**< **0**, **InstantiateH**< **TypeList**< **T**, **U** >, **Holder**, **i** >, **i** >
- struct **InstantiateHAccessor**< **j**, **InstantiateH**< **TypeList**< **T**, **U** >, **Holder**, **i** >, **i** >
- struct **Int2Type**
- class **InternalLibraryHandle**
- struct **IsConst**
- class **IsDelimiter**
- Generic string delimiter check function class.*
- struct **IsIntType**
- struct **IsIntType**< **Int2Type**< **i** >, **i** >
- struct **IsPointer**
- struct **IsReference**

- class **IsSlash**  
*Determines if the current character is a forward slash.*
- class **IsSpace**  
*A functor which tests if a character is whitespace.*
- class **KDTree**
- class **KeyFrameDecoder**  
*A class that fills key frame data with DOM parsing.*
- struct **Length**< **NullType** >
- struct **Length**< **TypeList**< **T, U** > >
- class **LibrarySharingManager**  
*Singleton for controlling loading and unloading libraries shared by multiple bodies of code.*
- class **LocalCoordinateType**
- class **Log**  
*Log (p. 205) class which the engine uses for all of its logging needs.*
- class **LogFile**
- struct **LogImpl**
- class **LogManager**
- class **MatrixUtil**  
*MatrixUtil (p. 212) is a utility class for operating on an osg::Matrix.*
- struct **MergeParmsH**
- struct **MergeParmsH**< **0, BoundPTL, UnboundPTL, dtUtil::NullType, TL** >
- struct **MergeParmsH**< **i, BoundPTL, UnboundPTL, IdsTL, dtUtil::NullType** >
- class **NodeCollector**  
*NodeCollector (p. 218) is used to gather osg Group nodes, DOFTransform nodes, MatrixTransform nodes, Switch Nodes and Geode nodes (which have Drawable objects and Material objects).*
- class **NodePrintOut**  
*Utility class used to traverse a node and generate a formatted text output.*
- class **Noise1**  
*An implementation of 1D Gradient Noise.*
- class **Noise2**  
*An implementation of 2D Gradient Noise.*
- class **Noise3**  
*An implementation of 3D Gradient Noise.*
- class **NoiseTexture**  
*Noise Texture is a class that uses **SeamlessNoise** (p. 251) to generate an osg::Image.*
- struct **NotIntType**
- class **NullType**
- class **ObjectFactory**  
*This class is a template object factory.*
- class **Packager**  
*The **Packager** (p. 241) is used to package multiple files into a single .dtpkg file.*
- class **PolarDecomp**

***PolarDecomp** (p. 244) is a class that will take a 4x4 Matrix and break it up into the following components Rotation, Scale, and Translation.*

- class **RefString**

*A string wrapper that will "intern" all of the strings so that strings with the same value will point to the same memory.*

- class **ResourceLoader**
- class **ResourceManager**
- class **SeamlessNoise**

***SeamlessNoise** (p. 251) is a noise class that creates tileable noise.*

- struct **Select**
- struct **Select**< **false**, **T**, **U** >
- struct **Serializer**

*A place to implement functions for serialization.*

- struct **squared\_difference**
- struct **squared\_difference\_counted**
- class **StateAttributeCollector**
- class **StateVisitor**
- class **StringTokenizer**
- class **StringToXMLConverter**

*Utility methods for using strings, often for XML purposes.*

- struct **TailAt**< **InstantiateH**< **TypeList**< **T**, **U** >, **Holder**, **i** >, **0**, **i** >
- struct **TailAt**< **InstantiateH**< **TypeList**< **T**, **U** >, **Holder**, **i** >, **j**, **i** >
- class **TangentSpaceVisitor**

*This visitor is used to generate tangents for your node.*

- class **ToLowerClass**
- class **Transformation**

*An interface class to use for "things that transform" i.e., things that return a value given an input value.*

- class **tree**
- class **tree\_iterator**
- struct **TupleHolder**
- struct **TypeAt**< **TypeList**< **T**, **U** >, **0** >
- struct **TypeAt**< **TypeList**< **T**, **U** >, **i** >
- struct **TypeAtNonStrict**
- struct **TypeAtNonStrict**< **TypeList**< **T**, **U** >, **0**, **DefType** >
- struct **TypeAtNonStrict**< **TypeList**< **T**, **U** >, **i**, **DefType** >
- struct **TypeList**
- struct **TypeTraits**
- struct **UnboundHelper**
- struct **UnboundTL2**
- struct **UTMParameters**
- class **XercesErrorHandler**

*Logs Xerces parsing errors.*

- class **XercesParser**

*A class to unify the usage of Xerces parsing.*

- class **XercesWriter**

*A class that manages one XML DOM document.*

- class **XMLStringConverter**

*Utility methods for using strings, often for XML purposes.*

## Typedefs

- typedef std::vector< std::string > **DirectoryContents**
- typedef std::vector< std::string > **FileExtensionList**
- typedef **Fractal**< double, osg::Vec2d, **Noise2d** > **Fractal2d**
- typedef **Fractal**< float, osg::Vec2f, **Noise2f** > **Fractal2f**
- typedef **Fractal**< double, osg::Vec3d, **Noise3d** > **Fractal3d**
- typedef **Fractal**< float, osg::Vec3f, **Noise3f** > **Fractal3f**
- typedef **Noise1**< double, double > **Noise1d**
- typedef **Noise1**< float, float > **Noise1f**

*NoiseUtility.h* (p. 372) contains all necessary defines to use the Noise Library in *dtUtil* (p. 13).

- typedef **Noise2**< double, osg::Vec2d > **Noise2d**
- typedef **Noise2**< float, osg::Vec2f > **Noise2f**
- typedef **Noise3**< double, osg::Vec3d > **Noise3d**
- typedef **Noise3**< float, osg::Vec3f > **Noise3f**
- typedef **Fractal**< float, osg::Vec3f, **SeamlessNoise** > **SeamlessFractal**

## Enumerations

- enum **FileType** { **FILE\_NOT\_FOUND**, **REGULAR\_FILE**, **DIRECTORY** }

## Functions

- template<typename \_ValA , typename \_ValB , typename \_Dist , typename \_Acc >  
\_Dist::distance\_type **S\_accumulate\_node\_distance** (const size\_t \_\_dim, const \_Dist &\_\_dist, const \_Acc &\_\_acc, const \_ValA &\_\_a, const \_ValB &\_\_b)
- template<typename \_ValA , typename \_ValB , typename \_Cmp , typename \_Acc >  
bool **S\_node\_compare** (const size\_t \_\_dim, const \_Cmp &\_\_cmp, const \_Acc &\_\_acc, const \_ValA &\_\_a, const \_ValB &\_\_b)
- template<typename \_Val , typename \_Cmp , typename \_Acc , typename NodeType >  
NodeType \* **S\_node\_descend** (const size\_t \_\_dim, const \_Cmp &\_\_cmp, const \_Acc &\_\_acc, const \_Val &\_\_val, const NodeType \*\_\_node)
- template<typename \_ValA , typename \_ValB , typename \_Dist , typename \_Acc >  
\_Dist::distance\_type **S\_node\_distance** (const size\_t \_\_dim, const \_Dist &\_\_dist, const \_Acc &\_\_acc, const \_ValA &\_\_a, const \_ValB &\_\_b)
- template<class SearchVal , typename NodeType , typename \_Cmp , typename \_Acc , typename \_Dist , typename \_Predicate >  
std::pair< const NodeType \*, std::pair< size\_t, typename \_Dist::distance\_type > > **S\_node\_nearest** (const size\_t \_\_k, size\_t \_\_dim, SearchVal const &\_\_val, const NodeType \*\_\_node, const **Node\_base** \*\_\_end, const NodeType \*\_\_best, typename \_Dist::distance\_type \_\_max, const \_Cmp &\_\_cmp, const \_Acc &\_\_acc, const \_Dist &\_\_dist, \_Predicate \_\_p)
- template<typename Real >  
Real **Abs** (Real x)
- template<int i1, int i2, int i3, int i4, int i5, int i6, int i7, class Incoming >  
**Binder**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2, i3, i4, i5, i6, i7 >::Type >::Outgoing **Bind** (Incoming const &fun, typename **BoundHelper**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2, i3, i4, i5, i6, i7 >::Type >::Parm1 p1, typename **BoundHelper**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2, i3, i4, i5, i6, i7 >::Type >::Parm2 p2, typename **BoundHelper**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2, i3, i4, i5, i6, i7 >::Type >::Parm3 p3, typename **BoundHelper**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2, i3, i4, i5, i6, i7 >::Type >::Parm4 p4, typename **BoundHelper**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2, i3, i4, i5, i6, i7 >::Type >::Parm5 p5, typename **BoundHelper**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2, i3, i4, i5, i6, i7 >::Type >::Parm5 p6, typename **BoundHelper**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2, i3, i4, i5, i6, i7 >::Type >::Parm5 p7)
- template<int i1, int i2, int i3, int i4, int i5, int i6, class Incoming >  
**Binder**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2, i3, i4, i5, i6 >::Type >::Outgoing **Bind** (Incoming const &fun, typename **BoundHelper**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2, i3, i4, i5, i6 >::Type >::Parm1 p1, typename **BoundHelper**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2, i3, i4, i5, i6 >::Type >::Parm2 p2, typename **BoundHelper**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2, i3, i4, i5, i6 >::Type >::Parm3 p3, typename **BoundHelper**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2,

- i3, i4, i5, i6 >::Type >::Parm4 p4, typename **BoundHelper**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2, i3, i4, i5, i6 >::Type >::Parm5 p5, typename **BoundHelper**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2, i3, i4, i5, i6 >::Type >::Parm5 p6)
- template<int i1, int i2, int i3, int i4, int i5, class Incoming >  
**Binder**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2, i3, i4, i5 >::Type >::Outgoing **Bind** (Incoming const &fun, typename **BoundHelper**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2, i3, i4, i5 >::Type >::Parm1 p1, typename **BoundHelper**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2, i3, i4, i5 >::Type >::Parm2 p2, typename **BoundHelper**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2, i3, i4, i5 >::Type >::Parm3 p3, typename **BoundHelper**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2, i3, i4, i5 >::Type >::Parm4 p4, typename **BoundHelper**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2, i3, i4, i5 >::Type >::Parm5 p5)
  - template<int i1, int i2, int i3, int i4, class Incoming >  
**Binder**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2, i3, i4 >::Type >::Outgoing **Bind** (Incoming const &fun, typename **BoundHelper**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2, i3, i4 >::Type >::Parm1 p1, typename **BoundHelper**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2, i3, i4 >::Type >::Parm2 p2, typename **BoundHelper**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2, i3, i4 >::Type >::Parm3 p3, typename **BoundHelper**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2, i3, i4 >::Type >::Parm4 p4)
  - template<int i1, int i2, int i3, class Incoming >  
**Binder**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2, i3 >::Type >::Outgoing **Bind** (Incoming const &fun, typename **BoundHelper**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2, i3 >::Type >::Parm1 p1, typename **BoundHelper**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2, i3 >::Type >::Parm2 p2, typename **BoundHelper**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2, i3 >::Type >::Parm3 p3)
  - template<int i1, int i2, class Incoming >  
**Binder**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2 >::Type >::Outgoing **Bind** (Incoming const &fun, typename **BoundHelper**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2 >::Type >::Parm1 p1, typename **BoundHelper**< Incoming, typename **dtUtil::CreateldsTL**< i1, i2 >::Type >::Parm2 p2)
  - template<int i1, class Incoming >  
**Binder**< Incoming, typename **dtUtil::CreateldsTL**< i1 >::Type >::Outgoing **Bind** (Incoming const &fun, typename **BoundHelper**< Incoming, typename **dtUtil::CreateldsTL**< **dtUtil::Int2Type**< i1 > >::Type >::Parm1 p1)
  - template<class Incoming >  
**Binder**< Incoming, typename **dtUtil::CreateldsTL**<>::Type >::Outgoing **Bind** (Incoming const &fun)
  - template<class V >  
**V CalculateBarycentricCenter** (const V &a, const V &b, const V &c)
  - template<typename T >  
**T CalculateNormal** (T sX, T sMin, T sMax)  
*Normalizes a value within a specified space range.*
  - template<typename Real >  
void **Clamp** (Real &x, const Real low, const Real high)
  - template<typename Real >  
void **ClampMax** (Real &x, const Real high)
  - template<typename Real >  
void **ClampMin** (Real &x, const Real low)
  - template<typename BaseType , typename DerivedType >  
BaseType \* **construct** ()  
*Templated function to provide a generic object construction utility.*
  - template<typename TVec >  
bool **Equivalent** (const TVec &lhs, const TVec &rhs)  
*Does an epsilon equals on an any osg::Vec#, but auto calculates the epsilon per comparison.*
  - template<typename TVec , typename Real >  
bool **Equivalent** (const TVec &lhs, const TVec &rhs, Real epsilon)  
*Does an epsilon equals on an any osg::Vec#.*
  - template<typename TVec , typename Real >  
bool **Equivalent** (const TVec &lhs, const TVec &rhs, size\_t size, Real epsilon)

*Does an epsilon equals on an any osg::Vec# or array.*

- bool **Equivalent** (double double1, double double2, double baseEpsilon=DBL\_EPSILON)  
*This does a relative comparison of doubles.*
- bool **Equivalent** (float float1, float float2, float baseEpsilon=FLT\_EPSILON)  
*This does a relative comparison of floats.*
- std::string DT\_UTIL\_EXPORT **FindAttributeValueFor** (const char \*attributeName, XERCES\_CPP\_NAMESPACE\_QUALIFIER DOMNamedNodeMap \*attrs)  
*A utility that finds the string value for a specifically named attribute when a DOM Node is available.*
- std::string **getFileExtensionIncludingDot** (const std::string &fileName)
- template<unsigned int j, class Instantiated >  
InstantiateHAccessor< j, Instantiated, Instantiated::ordern >::TargetHolder const & **GetH** (Instantiated const &h)
- template<unsigned int j, class Instantiated >  
InstantiateHAccessor< j, Instantiated, Instantiated::ordern >::TargetHolder & **GetH** (Instantiated &h)
- bool **iMakeDirectory** (const std::string &path)
- **IMPLEMENT\_ENUM (BaseExceptionType)**
- **IMPLEMENT\_ENUM (DateTime::TimeFormat)**
- **IMPLEMENT\_ENUM (DateTime::TimeType)**
- **IMPLEMENT\_ENUM (DateTime::TimeOrigin)**
- template<typename T >  
bool **IsFinite** (const T value)
- template<typename VecType >  
bool **IsFiniteVec** (const VecType value)
- template<typename T >  
bool **IsNAN** (const T value)
- template<typename Real >  
Real **Lerp** (Real x, Real y, Real t)  
*Apply a linear interpolation between the two supplied numbers using a third percentage value.*
- template<typename CallType, class Fun >  
**Functor**< typename dtUtil::FunTraits< CallType >::ResultType, typename dtUtil::FunTraits< CallType >::TypeListType > **MakeFunctor** (Fun const &fun)
- template<typename CallType, class PObj >  
**Functor**< typename dtUtil::FunTraits< CallType >::ResultType, typename dtUtil::FunTraits< CallType >::TypeListType > **MakeFunctor** (CallType memfun, PObj const &pobj)
- template<typename CallType >  
**Functor**< typename dtUtil::FunTraits< CallType >::ResultType, typename dtUtil::FunTraits< CallType >::TypeListType > **MakeFunctor** (CallType fun)
- void **MakeIndexString** (unsigned index, std::string &toFill, unsigned paddedLength=4)
- static dtCore::RefPtr< **LogManager** > **manager** (NULL)
- template<typename T >  
T **MapRangeValue** (T sX, T xMin, T xMax, T yMin, T yMax)  
*Calculates the corresponding value for a mirrored space.*
- bool **Match** (const char \*Wildcards, const char \*str)
- template<typename Real >  
Real **Max** (Real a, Real b)
- template<class TL, class IdsTL, class BoundPTL, class UnboundPTL >  
CallParms< TL >::ParmsListType **MergeParms** (BoundPTL const &bound, UnboundPTL const &unbound)
- template<typename Real >  
Real **Min** (Real a, Real b)
- bool **operator!=** (const std::string &s1, const **RefString** &s2)

- `template<typename _Val >`  
`bool operator!= (_Iterator< _Val, _Val &, _Val * > const &, _Iterator< _Val, const _Val &, const _Val * >`  
`const &)`
- `template<typename _Val >`  
`bool operator!= (_Iterator< _Val, const _Val &, const _Val * > const &, _Iterator< _Val, _Val &, _Val * >`  
`const &)`
- `template<typename _Val, typename _Ref, typename _Ptr >`  
`bool operator!= (_Iterator< _Val, _Ref, _Ptr > const &, _Iterator< _Val, _Ref, _Ptr > const &)`
- `std::string operator+ (const std::string &s1, const RefString &s2)`
- `std::ostream & operator<< (std::ostream &stream, const RefString &rs)`
- `DT_UTIL_EXPORT std::ostream & operator<< (std::ostream &o, const Exception &ex)`
- `std::ostream & operator<< (std::ostream &os, const Enumeration &e)`  
*Helper method to print enumerations to an output stream.*
  
- `bool operator== (const std::string &s1, const RefString &s2)`
- `template<typename _Val >`  
`bool operator== (_Iterator< _Val, _Val &, _Val * > const &, _Iterator< _Val, const _Val &, const _Val * >`  
`const &)`
- `template<typename _Val >`  
`bool operator== (_Iterator< _Val, const _Val &, const _Val * > const &, _Iterator< _Val, _Val &, _Val * >`  
`const &)`
- `template<typename _Val, typename _Ref, typename _Ptr >`  
`bool operator== (_Iterator< _Val, _Ref, _Ptr > const &, _Iterator< _Val, _Ref, _Ptr > const &)`
- `template<class VecType >`  
`bool ParseVec (const std::string &value, VecType &vec, unsigned size, unsigned numberPrecision=16)`  
*A templated function for taking any of the osg vector types and reading the data from a string.*
  
- `float RandFloat (float min, float max)`
- `float RandPercent ()`
- `int RandRange (int from, int to)`
- `template<typename T >`  
`T safeASIN (T x)`
- `static bool Scan (const char *&wildCards, const char *&str)`
- `static std::string sTitle ("Delta 3D Engine Log File")`
- `double ToDouble (const std::string &str)`  
*converts a std::string to a 'double'*
  
- `float ToFloat (const std::string &d)`  
*Converts a string to a 'float'.*
  
- `template<typename T >`  
`std::string ToString (const T &t, int precision=-1)`  
*A utility function to convert a basic type into a string.*
  
- `template<typename T >`  
`T ToType (const std::string &u)`  
*Converts a string to a specified type.*
  
- `template<>`  
`bool ToType< bool > (const std::string &u)`  
*Special exception for bool where things like "True", "TRUE", and "true" should be accepted.*
  
- `unsigned int ToUnsignedInt (const std::string &u)`  
*converts a std::string to an 'unsigned int'*
  
- `DEPRECATE_FUNC const std::string & trim (std::string &toTrim)`  
*Call Trim instead.*

- const std::string & **Trim** (std::string &toTrim)  
*Trims whitespace off the front and end of a string.*
- static bool **WildMatch** (const char \*wildCards, const char \*str)
- template<typename Real >  
bool **WithinRange** (Real lhs, Real rhs, Real epsilon)

## Variables

- const double **AD\_C** = 1.0026000
- const double **CentralMeridianScale** = 0.9996  
*Scale used in UTM calculations.*
- const double **COS\_67P5** = 0.38268343236508977
- const double **flatteningReciprocal** = 298.257223563  
*The reciprocal of the flattening parameter (WGS 84).*
- const double **Geocent\_a** = **semiMajorAxis**
- const double **Geocent\_e2** = (2.0 - **Geocent\_f**) \* **Geocent\_f**
- const double **Geocent\_e2\_2** = **Geocent\_e2** \* **Geocent\_e2**
- const double **Geocent\_e2\_3** = **Geocent\_e2\_2** \* **Geocent\_e2**
- const double **Geocent\_ef** = **Geocent\_f** / (2.0 - **Geocent\_f**)
- const double **Geocent\_ef\_3** = **Geocent\_ef** \* **Geocent\_ef** \* **Geocent\_ef**
- const double **Geocent\_ef\_4** = **Geocent\_ef\_3** \* **Geocent\_ef**
- const double **Geocent\_ep2** = **Geocent\_e2** / (1.0 - **Geocent\_e2**)
- const double **Geocent\_f** = 1 / **flatteningReciprocal**
- static OpenThreads::Mutex **gStringSetMutex**
- static const char \* **LOGNAME** = "DataStream"
- const double **MagneticNorthLatitude** = 82.116
- const double **MagneticNorthLongitude** = 114.0666
- const double **MAX\_DELTA\_LONG** = ((osg::PI \* 90)/180.0)
- const double **MAX\_EASTING** = 900000
- const double **MAX\_LAT** = ( 84.5 \* osg::PI) / 180.0 )
- const double **MAX\_NORTHING** = 10000000
- const double **MAX\_SCALE\_FACTOR** = 3.0
- const double **METERS\_PER\_DEGREE** = 1852.0\*60
- const double **MIN\_EASTING** = 100000
- const double **MIN\_LAT** = ( (-80.5 \* osg::PI) / 180.0 )
- const double **MIN\_NORTHING** = 0
- const double **MIN\_SCALE\_FACTOR** = 0.3
- int **p** [512]
- int **permutation** [256]
- const double **semiMajorAxis** = 6378137.0  
*The length of the semi-major axis, in meters (WGS 84).*
- static const char \* **sLogFileName** = "delta3d\_log.html"
- static size\_t **StringCount** = 0
- static std::set< std::string > **StringSet**

### 5.1.1 Detailed Description

Contains generic, reusable features which are useful for most applications.

## 5.1.2 Typedef Documentation

5.1.2.1 typedef std::vector<std::string> DirectoryContents

5.1.2.2 typedef std::vector<std::string> FileExtensionList

5.1.2.3 typedef Fractal<double, osg::Vec2d, Noise2d> Fractal2d

5.1.2.4 typedef Fractal<float, osg::Vec2f, Noise2f> Fractal2f

5.1.2.5 typedef Fractal<double, osg::Vec3d, Noise3d> Fractal3d

5.1.2.6 typedef Fractal<float, osg::Vec3f, Noise3f> Fractal3f

5.1.2.7 typedef Noise1<double, double> Noise1d

5.1.2.8 typedef Noise1<float, float> Noise1f

**NoiseUtility.h** (p. 372) contains all necessary defines to use the Noise Library in **dtUtil** (p. 13). The noise classes are templated to allow varying precision and dimensions ex: for a noise class with floating point precision and two components use Noise2f

5.1.2.9 typedef Noise2<double, osg::Vec2d> Noise2d

5.1.2.10 typedef Noise2<float, osg::Vec2f> Noise2f

5.1.2.11 typedef Noise3<double, osg::Vec3d> Noise3d

5.1.2.12 typedef Noise3<float, osg::Vec3f> Noise3f

5.1.2.13 typedef Fractal<float, osg::Vec3f, SeamlessNoise> SeamlessFractal

## 5.1.3 Enumeration Type Documentation

5.1.3.1 enum FileType

Enumerator:

*FILE\_NOT\_FOUND*

*REGULAR\_FILE*

*DIRECTORY*

## 5.1.4 Function Documentation

5.1.4.1 **\_Dist::distance\_type dtUtil::\_S\_accumulate\_node\_distance** (const size\_t \_\_dim, const \_Dist & \_\_dist, const \_Acc & \_\_acc, const \_ValA & \_\_a, const \_ValB & \_\_b) [inline]

Compute the distance between two values and accumulate the result for all dimensions.

The distance functor and the accessor are references to the template parameters of the **KDTree** (p. 190).

5.1.4.2 **bool dtUtil::\_S\_node\_compare** (const size\_t \_\_dim, const \_Cmp & \_\_cmp, const \_Acc & \_\_acc, const \_ValA & \_\_a, const \_ValB & \_\_b) [inline]

Compare two values on the same dimension using a comparison functor \_Cmp and an accessor \_Acc.

The comparison functor and the accessor are references to the template parameters of the **KDTree** (p. 190).

5.1.4.3 **NodeType\* dtUtil::\_S\_node\_descend** (const size\_t \_\_dim, const \_Cmp & \_\_cmp, const \_Acc & \_\_acc, const \_Val & \_\_val, const NodeType \* \_\_node) [inline]

Descend on the left or the right of the node according to the comparison between the node's value and the value.

Note it's the caller responsibility to check if node is NULL.

5.1.4.4 **\_Dist::distance\_type dtUtil::\_S\_node\_distance** (const size\_t \_\_dim, const \_Dist & \_\_dist, const \_Acc & \_\_acc, const \_ValA & \_\_a, const \_ValB & \_\_b) [inline]

Compute the distance between two values for one dimension only.

The distance functor and the accessor are references to the template parameters of the **KDTree** (p. 190).

5.1.4.5 `std::pair<const NodeType*, std::pair<size_t, typename _Dist::distance_type> >`  
`dtUtil::_S_node_nearest` (`const size_t __k`, `size_t __dim`, `SearchVal const & __val`, `const`  
`NodeType * __node`, `const _Node_base * __end`, `const NodeType * __best`, `typename`  
`_Dist::distance_type __max`, `const _Cmp & __cmp`, `const _Acc & __acc`, `const _Dist & __dist`,  
`_Predicate __p`) [`inline`]

Find the nearest node to `__val` from `__node`

If many nodes are equidistant to `__val`, the node with the lowest memory address is returned.

Returns the nearest node of `__end` node if no nearest node was found for the given arguments.

- 5.1.4.6 Real dtUtil::Abs (Real *x*) [inline]
- 5.1.4.7 Binder<Incoming, typename dtUtil::CreateldsTL<i1, i2, i3, i4, i5, i6, i7>::Type>::Outgoing dtUtil::Bind (Incoming const & *fun*, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5, i6, i7 >::Type >::Parm1 *p1*, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5, i6, i7 >::Type >::Parm2 *p2*, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5, i6, i7 >::Type >::Parm3 *p3*, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5, i6, i7 >::Type >::Parm4 *p4*, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5, i6, i7 >::Type >::Parm5 *p5*, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5, i6, i7 >::Type >::Parm5 *p6*, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5, i6, i7 >::Type >::Parm5 *p7*) [inline]
- 5.1.4.8 Binder<Incoming, typename dtUtil::CreateldsTL<i1, i2, i3, i4, i5, i6>::Type>::Outgoing dtUtil::Bind (Incoming const & *fun*, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5, i6 >::Type >::Parm1 *p1*, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5, i6 >::Type >::Parm2 *p2*, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5, i6 >::Type >::Parm3 *p3*, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5, i6 >::Type >::Parm4 *p4*, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5, i6 >::Type >::Parm5 *p5*, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5, i6 >::Type >::Parm5 *p6*) [inline]
- 5.1.4.9 Binder<Incoming, typename dtUtil::CreateldsTL<i1, i2, i3, i4, i5>::Type>::Outgoing dtUtil::Bind (Incoming const & *fun*, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5 >::Type >::Parm1 *p1*, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5 >::Type >::Parm2 *p2*, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5 >::Type >::Parm3 *p3*, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5 >::Type >::Parm4 *p4*, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5 >::Type >::Parm5 *p5*) [inline]
- 5.1.4.10 Binder<Incoming, typename dtUtil::CreateldsTL<i1, i2, i3, i4>::Type>::Outgoing dtUtil::Bind (Incoming const & *fun*, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4 >::Type >::Parm1 *p1*, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4 >::Type >::Parm2 *p2*, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4 >::Type >::Parm3 *p3*, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4 >::Type >::Parm4 *p4*) [inline]
- 5.1.4.11 Binder<Incoming, typename dtUtil::CreateldsTL<i1, i2, i3>::Type>::Outgoing dtUtil::Bind (Incoming const & *fun*, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3 >::Type >::Parm1 *p1*, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3 >::Type >::Parm2 *p2*, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3 >::Type >::Parm3 *p3*) [inline]
- 5.1.4.12 Binder<Incoming, typename dtUtil::CreateldsTL<i1, i2>::Type>::Outgoing dtUtil::Bind (Incoming const & *fun*, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2 >::Type >::Parm1 *p1*, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2 >::Type >::Parm2 *p2*) [inline]
- 5.1.4.13 Binder<Incoming, typename dtUtil::CreateldsTL<i1>::Type>::Outgoing dtUtil::Bind (Incoming const & *fun*, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1 >::Type >::Parm1 *p1*) [inline]
- 5.1.4.14 Binder<Incoming, typename dtUtil::CreateldsTL<>::Type>::Outgoing dtUtil::Bind (Incoming const & *fun*) [inline]
- 5.1.4.15 V dtUtil::CalculateBarycentricCenter (const V & *a*, const V & *b*, const V & *c*) [inline]
- 5.1.4.16 T dtUtil::CalculateNormal (T *sX*, T *sMin*, T *sMax*) [inline]

Normalizes a value within a specified space range. Usage: To find the normalized value for a range: float *nX* = CalculateNormal( *valueX* , *xMin* , *xMax* ); Parameters

**sX** the value with respect to the specified range to be normalized.

***sMin*** specifies the left bound of the range.

***sMax*** specifies the right bound of the range.

Returns the normalized coefficient for the input to the range.

**5.1.4.17 void dtUtil::Clamp (Real & x, const Real *low*, const Real *high*) [inline]**

**5.1.4.18 void dtUtil::ClampMax (Real & x, const Real *high*) [inline]**

**5.1.4.19 void dtUtil::ClampMin (Real & x, const Real *low*) [inline]**

**5.1.4.20 BaseType\* dtUtil::construct () [inline]**

Templated function to provide a generic object construction utility.

**5.1.4.21 bool dtUtil::Equivalent (const TVec & *lhs*, const TVec & *rhs*) [inline]**

Does an epsilon equals on an any osg::Vec#, but auto calculates the epsilon per comparison. Parameters

***lhs*** The first vector.

***rhs*** The second vector.

**5.1.4.22 bool dtUtil::Equivalent (const TVec & *lhs*, const TVec & *rhs*, Real *epsilon*) [inline]**

Does an epsilon equals on an any osg::Vec#. Parameters

***lhs*** The first vector.

***rhs*** The second vector.

***epsilon*** the epsilon to use in the compare.

**5.1.4.23 bool dtUtil::Equivalent (const TVec & *lhs*, const TVec & *rhs*, size\_t *size*, Real *epsilon*) [inline]**

Does an epsilon equals on an any osg::Vec# or array. Parameters

***lhs*** The first vector.

***rhs*** The second vector.

***size*** The size or the vec.

***epsilon*** the epsilon to use in the compare.

**5.1.4.24 bool dtUtil::Equivalent (double *double1*, double *double2*, double *baseEpsilon* = DBL\_EPSILON) [inline]**

This does a relative comparison of doubles. This is a SAFE comparison that doesn't use cheesy 0.0001 type epsilon values. The epsilon is scaled based on the precision of the numbers passed in. This was taken from Christer Ericson's GDC '07 presentation: [http://realtimecollisiondetection.net/pubs/GDC06\\_Ericson\\_Physics\\_Tutorial\\_Numerical\\_Robustness.ppt](http://realtimecollisiondetection.net/pubs/GDC06_Ericson_Physics_Tutorial_Numerical_Robustness.ppt) Note - This should be used when comparing very large and/or very small numbers. Parameters

***double1*** The first value

***double2*** The second value

Returns True if the values are equal within the relative precision of their values.

**5.1.4.25 bool dtUtil::Equivalent (float *float1*, float *float2*, float *baseEpsilon* = FLT\_EPSILON) [inline]**

This does a relative comparison of floats. This is a SAFE comparison that doesn't use cheesy 0.0001 type epsilon values. The epsilon is scaled based on the precision of the numbers passed in. This was taken from Christer Ericson's GDC '07 presentation: [http://realtimecollisiondetection.net/pubs/GDC06\\_Ericson\\_Physics\\_Tutorial\\_Numerical\\_Robustness.ppt](http://realtimecollisiondetection.net/pubs/GDC06_Ericson_Physics_Tutorial_Numerical_Robustness.ppt) Note - This should be used when comparing very large and/or very small numbers. Parameters

***float1*** The first float

***float2*** The second float

Returns True if the values are equal within the relative precision of their values.

**5.1.4.26 XERCES\_CPP\_NAMESPACE\_USE std::string FindAttributeValueFor (const char \* *attributeName*, XERCES\_CPP\_NAMESPACE\_QUALIFIER DOMNamedNodeMap \* *attrs*)**

A utility that finds the string value for a specifically named attribute when a DOM Node is available. A Xerces utility that finds the string value for a specifically named attribute.

Needed for DOM Document traversal. Parameters

***attributeName*** the name of the attribute of interest.

***attrs*** the NamedNodeMap (attributes) to be searched.

***attributeName*** the name of the attribute of interest.

***attrs*** the node with attributes to be searched.

**5.1.4.27 std::string dtUtil::getFileExtensionIncludingDot (const std::string & *fileName*)****5.1.4.28 InstantiateHAccessor<j, Instantiated, Instantiated::ordern>::TargetHolder const& dtUtil::GetH (Instantiated const & *h*) [inline]****5.1.4.29 InstantiateHAccessor<j, Instantiated, Instantiated::ordern>::TargetHolder& dtUtil::GetH (Instantiated & *h*) [inline]****5.1.4.30 bool dtUtil::iMakeDirectory (const std::string & *path*)****5.1.4.31 dtUtil::IMPLEMENT\_ENUM (BaseExceptionType)****5.1.4.32 dtUtil::IMPLEMENT\_ENUM (DateTime::TimeFormat)****5.1.4.33 dtUtil::IMPLEMENT\_ENUM (DateTime::TimeType)****5.1.4.34 dtUtil::IMPLEMENT\_ENUM (DateTime::TimeOrigin)****5.1.4.35 bool dtUtil::IsFinite (const T *value*) [inline]****5.1.4.36 bool dtUtil::IsFiniteVec (const VecType *value*) [inline]****5.1.4.37 bool dtUtil::IsNAN (const T *value*) [inline]****5.1.4.38 Real dtUtil::Lerp (Real *x*, Real *y*, Real *t*) [inline]**

Apply a linear interpolation between the two supplied numbers using a third percentage value. Parameters

***x*** : specifies the left bound of the range.

***y*** : specifies the right bound of the range.

***t*** : the normalized value with respect to the specified range to be interpolated.

Returns the interpolated value for the coefficient of the range.

5.1.4.39 `Functor<typename dtUtil::FunTraits<CallType>::ResultType, typename dtUtil::FunTraits<CallType>::TypeListType> dtUtil::MakeFunctor (Fun const & fun) [inline]`

5.1.4.40 `Functor<typename dtUtil::FunTraits<CallType>::ResultType, typename dtUtil::FunTraits<CallType>::TypeListType> dtUtil::MakeFunctor (CallType memfun, POBJ const & pobj) [inline]`

5.1.4.41 `Functor<typename dtUtil::FunTraits<CallType>::ResultType, typename dtUtil::FunTraits<CallType>::TypeListType> dtUtil::MakeFunctor (CallType fun) [inline]`

5.1.4.42 `void DT_UTIL_EXPORT MakeIndexString (unsigned index, std::string & toFill, unsigned paddedLength = 4)`

Returns a string with text as an int value padded to the size specified.

5.1.4.43 `static dtCore::RefPtr<LogManager> dtUtil::manager (NULL) [static]`

5.1.4.44 `T dtUtil::MapRangeValue (T sX, T xMin, T xMax, T yMin, T yMax) [inline]`

Calculates the corresponding value for a mirrored space. Given defined ranges for X space and Y space, and a known value in X space, where X space and Y space are linearly related, find the corresponding value in Y space. Usage: float y = MapRangeValue(x,xMin,xMax,yMin,yMax); Parameters

*sX* the value with respect to the X range to be transformed into the Y range.

*xMin* specifies the left bound of the X range.

*xMax* specifies the right bound of the X range.

*yMin* specifies the left bound of the Y range.

*yMax* specifies the right bound of the Y range.

5.1.4.45 `bool DT_UTIL_EXPORT Match (const char * Wildcards, const char * str)`

5.1.4.46 `Real dtUtil::Max (Real a, Real b) [inline]`

5.1.4.47 `CallParms<TL>::ParmsListType dtUtil::MergeParms (BoundPTL const & bound, UnboundPTL const & unbound) [inline]`

5.1.4.48 `Real dtUtil::Min (Real a, Real b) [inline]`

5.1.4.49 `bool dtUtil::operator!= (const std::string & s1, const RefString & s2) [inline]`

5.1.4.50 `bool operator!= (_Iterator<_Val, _Val &, _Val * > const & __X, _Iterator<_Val, const _Val &, const _Val * > const & __Y) [inline]`

5.1.4.51 `bool operator!= (_Iterator<_Val, const _Val &, const _Val * > const & __X, _Iterator<_Val, _Val &, _Val * > const & __Y) [inline]`

5.1.4.52 `bool operator!= (_Iterator<_Val, _Ref, _Ptr > const & __X, _Iterator<_Val, _Ref, _Ptr > const & __Y) [inline]`

5.1.4.53 `std::string dtUtil::operator+ (const std::string & s1, const RefString & s2) [inline]`

5.1.4.54 `DT_UTIL_EXPORT std::ostream & operator<< (std::ostream & stream, const RefString & rs)`

5.1.4.55 `DT_UTIL_EXPORT std::ostream & operator<< (std::ostream & o, const Exception & ex)`

5.1.4.56 `DT_UTIL_EXPORT std::ostream & operator<< (std::ostream & os, const Enumeration & e)`

Helper method to print enumerations to an output stream.

5.1.4.57 **bool dtUtil::operator== (const std::string & s1, const RefString & s2) [inline]**

5.1.4.58 **bool operator== (\_Iterator< \_Val, \_Val &, \_Val \* > const & \_\_X, \_Iterator< \_Val, const \_Val &, const \_Val \* > const & \_\_Y) [inline]**

5.1.4.59 **bool operator== (\_Iterator< \_Val, const \_Val &, const \_Val \* > const & \_\_X, \_Iterator< \_Val, \_Val &, \_Val \* > const & \_\_Y) [inline]**

5.1.4.60 **bool operator== (\_Iterator< \_Val, \_Ref, \_Ptr > const & \_\_X, \_Iterator< \_Val, \_Ref, \_Ptr > const & \_\_Y) [inline]**

5.1.4.61 **bool dtUtil::ParseVec (const std::string & value, VecType & vec, unsigned size, unsigned numberPrecision = 16) [inline]**

A templated function for taking any of the osg vector types and reading the data from a string. If the string is empty or "NULL" it will set the vector to all 0s. It expects the data to be the proper number floating point values. The function will fail if there are not enough values.

Parameters

**value** the string data.

**vec** the vector to fill.

**size** the length of the vector since the osg types have no way to query that.

**numberPrecision** This value indicates how much precision the numbers will contain when read from the string. (setprecision on std::istream)

Returns true if reading the data was successful or false if not.

5.1.4.62 **float dtUtil::RandFloat (float min, float max) [inline]**

5.1.4.63 **float dtUtil::RandPercent () [inline]**

5.1.4.64 **int dtUtil::RandRange (int from, int to) [inline]**

5.1.4.65 **T dtUtil::safeASIN (T x) [inline]**

5.1.4.66 **static bool dtUtil::Scan (const char \*& wildCards, const char \*& str) [static]**

5.1.4.67 **static std::string dtUtil::sTitle ("Delta 3D Engine Log File") [static]**

5.1.4.68 **double DT\_UTIL\_EXPORT ToDouble (const std::string & str)**

converts a std::string to a 'double'

#### Deprecated

Favor ToType<double> instead

5.1.4.69 **float DT\_UTIL\_EXPORT ToFloat (const std::string & d)**

Converts a string to a 'float'.

#### Deprecated

Favor ToType<float> instead

5.1.4.70 **std::string dtUtil::ToString (const T & t, int precision = -1) [inline]**

A utility function to convert a basic type into a string. Use template argument T for the type you'd like to convert.

Parameters

**t** the instance of the type to converted.

**5.1.4.71 T dtUtil::ToType (const std::string & u) [inline]**

Converts a string to a specified type. Parameters

*the* string to be converted to the specified template argument type.

Returns the type that you specify as the template argument. Typical use:

```
std::string mystring("0");
bool mybool = dtUtil::ToType<bool>( mystring );
```

**Todo**

make a specialization for 'bool' supporting "false" and "true".

**5.1.4.72 bool DT\_UTIL\_EXPORT ToType< bool > (const std::string & u) [inline]**

Special exception for bool where things like "True", "TRUE", and "true" should be accepted.

**5.1.4.73 unsigned int DT\_UTIL\_EXPORT ToUnsignedInt (const std::string & u)**

converts a std::string to an 'unsigned int'

**Deprecated**

Favor ToType<unsigned int> instead

**5.1.4.74 DEPRECATE\_FUNC const std::string& dtUtil::trim (std::string & toTrim) [inline]**

Call Trim instead.

**5.1.4.75 DT\_UTIL\_EXPORT const std::string & Trim (std::string & toTrim)**

Trims whitespace off the front and end of a string. Parameters

*toTrim* the string to trim.

**5.1.4.76 static bool dtUtil::WildMatch (const char \* wildCards, const char \* str) [static]****5.1.4.77 bool dtUtil::WithinRange (Real lhs, Real rhs, Real epsilon) [inline]****5.1.5 Variable Documentation****5.1.5.1 const double AD\_C = 1.0026000****5.1.5.2 const double CentralMeridianScale = 0.9996**

Scale used in UTM calculations.

**5.1.5.3 const double COS\_67P5 = 0.38268343236508977****5.1.5.4 const double flatteningReciprocal = 298.257223563**

The reciprocal of the flattening parameter (WGS 84).

- 5.1.5.5 `const double Geocent_a = semiMajorAxis`
- 5.1.5.6 `const double Geocent_e2 = (2.0 - Geocent_f) * Geocent_f`
- 5.1.5.7 `const double Geocent_e2_2 = Geocent_e2 * Geocent_e2`
- 5.1.5.8 `const double Geocent_e2_3 = Geocent_e2_2 * Geocent_e2`
- 5.1.5.9 `const double Geocent_ef = Geocent_f / (2.0 - Geocent_f)`
- 5.1.5.10 `const double Geocent_ef_3 = Geocent_ef * Geocent_ef * Geocent_ef`
- 5.1.5.11 `const double Geocent_ef_4 = Geocent_ef_3 * Geocent_ef`
- 5.1.5.12 `const double Geocent_ep2 = Geocent_e2 / (1.0 - Geocent_e2)`
- 5.1.5.13 `const double Geocent_f = 1 / flatteningReciprocal`
- 5.1.5.14 `OpenThreads::Mutex gStringSetMutex [static]`
- 5.1.5.15 `const char* LOGNAME = "DataStream" [static]`
- 5.1.5.16 `const double MagneticNorthLatitude = 82.116`
- 5.1.5.17 `const double MagneticNorthLongitude = 114.0666`
- 5.1.5.18 `const double MAX_DELTA_LONG = ((osg::PI * 90)/180.0)`
- 5.1.5.19 `const double MAX_EASTING = 900000`
- 5.1.5.20 `const double MAX_LAT = ( (84.5 * osg::PI) / 180.0 )`
- 5.1.5.21 `const double MAX_NORTHING = 10000000`
- 5.1.5.22 `const double MAX_SCALE_FACTOR = 3.0`
- 5.1.5.23 `const double METERS_PER_DEGREE = 1852.0*60`
- 5.1.5.24 `const double MIN_EASTING = 100000`
- 5.1.5.25 `const double MIN_LAT = ( (-80.5 * osg::PI) / 180.0 )`
- 5.1.5.26 `const double MIN_NORTHING = 0`
- 5.1.5.27 `const double MIN_SCALE_FACTOR = 0.3`
- 5.1.5.28 `int p[512]`
- 5.1.5.29 `int permutation[256]`
- 5.1.5.30 `const double semiMajorAxis = 6378137.0`  
The length of the semi-major axis, in meters (WGS 84).
- 5.1.5.31 `const char* sLogFileName = "delta3d_log.html" [static]`
- 5.1.5.32 `size_t StringCount = 0 [static]`
- 5.1.5.33 `std::set<std::string> StringSet [static]`

## 5.2 dtUtil::Bits Namespace Reference

Contains bit-wise operation functionality which makes using bits a little "bit" easier.

### Functions

- `template<class N , class B >`  
**N Add** (N number, B bits)  
*Add the "bits" to "number".*
- `template<class N , class B >`  
**bool Has** (N number, B bits)  
*See if the "bits" are in "number".*
- `template<class N , class B >`  
**N Remove** (N number, B bits)  
*Remove the "bits" from "number".*
- `template<class N , class B >`  
**N Toggle** (N number, B bits)  
*Toggle the "bits" in "number".*

### 5.2.1 Detailed Description

Contains bit-wise operation functionality which makes using bits a little "bit" easier.

### 5.2.2 Function Documentation

#### 5.2.2.1 N dtUtil::Bits::Add (N number, B bits) [inline]

Add the "bits" to "number".

```
unsigned int accum = 1;
accum = Bits::Add(accum, 7); //Has(accum, 1) and Has(accum, 7) == true
```

#### 5.2.2.2 bool dtUtil::Bits::Has (N number, B bits) [inline]

See if the "bits" are in "number".

```
unsigned int accum = 3;
Bits::Has( accum, 1 ); //true
Bits::Has( accum, 7 ); //false
```

#### 5.2.2.3 N dtUtil::Bits::Remove (N number, B bits) [inline]

Remove the "bits" from "number".

```
unsigned int accum = 3;
accum = Bits::Remove(accum, 2); //Has(accum,3) == false, Has(accum,1) == true
```

#### 5.2.2.4 N dtUtil::Bits::Toggle (N number, B bits) [inline]

Toggle the "bits" in "number".

```
unsigned int accum = 3;
unsigned int newBits;
newBits = Bits::Toggle(accum, 1); //newBits = 2
newBits = Bits::Toggle(accum, 1); //newBits = 3
```

## 5.3 dtUtil::CollectorUtil Namespace Reference

### Classes

- struct **GetElementType**

*Template Function that is used to get the RefPtr's Element type from the map.*

### Functions

- template<class mapType >  
bool **AddNode** (const std::string &nodeName, typename **GetElementType**< mapType >::value\_type \*nodeType, mapType &nodeMap)

*Template function that is used to Add a unique node or object to its respective map.*

- template<class mapType >  
**GetElementType**< mapType >::value\_type \* **FindNodePointer** (const std::string &nodeName, mapType &nodeMap)

*Template function that is used to find a node or object that is inside a map.*

- template<class mapType >  
const **GetElementType**< mapType >::value\_type \* **FindNodePointer** (const std::string &nodeName, const mapType &nodeMap)

*Template function that is used to find a node or object that is inside a map.*

- template<class mapType >  
bool **RemoveNode** (const std::string &nodeName, mapType &nodeMap)

*Template function that is used to Remove a unique node or object from its respective map.*

### 5.3.1 Function Documentation

#### 5.3.1.1 bool dtUtil::CollectorUtil::AddNode (const std::string & *nodeName*, typename GetElementType< mapType >::value\_type \* *nodeType*, mapType & *nodeMap*) [inline]

Template function that is used to Add a unique node or object to its respective map. Parameters

***nodeName*** The name of the Node or Object that is being added to its map

***nodeType*** The address of the node or object that you wish to add to its map

***nodeMap*** The map that you want to add the node or object to

Returns Will return true if the node / object was added; otherwise false will be returned

#### 5.3.1.2 GetElementType<mapType>::value\_type\* dtUtil::CollectorUtil::FindNodePointer (const std::string & *nodeName*, mapType & *nodeMap*) [inline]

Template function that is used to find a node or object that is inside a map. Parameters

***nodeName*** The name of the Node or Object that you wish to find

***nodeMap*** The map that you wish to search for the Node / Object

Returns Will return a const pointer to the node or object that you were looking for if it is in the map. Otherwise it will return null.

**5.3.1.3** `const GetElementType<mapType>::value_type* dtUtil::CollectorUtil::FindNodePointer (const std::string & nodeName, const mapType & nodeMap) [inline]`

Template function that is used to find a node or object that is inside a map. Parameters

***nodeName*** The name of the Node or Object that you wish to find

***nodeMap*** The map that you wish to search for the Node / Object

Returns Will return a const pointer to a node or object that you were looking for if it is in the map. Otherwise it will return null.

**5.3.1.4** `bool dtUtil::CollectorUtil::RemoveNode (const std::string & nodeName, mapType & nodeMap) [inline]`

Template function that is used to Remove a unique node or object from its respective map. Parameters

***nodeName*** The name of the Node or Object that is being removed from its map

***nodeMap*** The map that you want to remove the node or object from

Returns Will return true if the node / object was found AND removed; otherwise false will be returned

## 5.4 dtUtil::details Namespace Reference

### Classes

- struct **TypeTraits**



# Class Documentation

---

## 6.1 `_Alloc_base<_Tp, _Alloc >` Class Template Reference

```
#include <inc/dtUtil/kdtree.h>
```

### Classes

- class `NoLeakAlloc`

### Public Types

- typedef `_Node_::_Base_ptr Base_ptr`
- typedef `_Node<_Tp > _Node_`
- typedef `_Alloc allocator_type`

### Public Member Functions

- `_Alloc_base` (`allocator_type` const &\_\_A)
- `allocator_type` `get_allocator` () const

### Protected Member Functions

- `_Node_ * _M_allocate_node` ()
- void `_M_construct_node` (`_Node_ * __p`, `_Tp` const \_\_V=`_Tp`(), `_Base_ptr` const \_\_PARENT=NULL, `_Base_ptr` const \_\_LEFT=NULL, `_Base_ptr` const \_\_RIGHT=NULL)
- void `_M_deallocate_node` (`_Node_ * const __P`)
- void `_M_destroy_node` (`_Node_ * __p`)

### Protected Attributes

- `allocator_type` `_M_node_allocator`

```
template<typename _Tp, typename _Alloc> class dtUtil::_Alloc_base<_Tp, _Alloc >
```

### 6.1.1 Member Typedef Documentation

#### 6.1.1.1 typedef `_Node_::_Base_ptr Base_ptr`

Reimplemented in `KDTree< __K, _Val, _Acc, _Dist, _Cmp, _Alloc >` (p. 193).

#### 6.1.1.2 typedef `_Node<_Tp> _Node_`

#### 6.1.1.3 typedef `_Alloc allocator_type`

Reimplemented in `KDTree< __K, _Val, _Acc, _Dist, _Cmp, _Alloc >` (p. 193).

## 6.1.2 Constructor & Destructor Documentation

6.1.2.1 `_Alloc_base (allocator_type const & __A) [inline]`

## 6.1.3 Member Function Documentation

6.1.3.1 `_Node_* _M_allocate_node () [inline, protected]`

6.1.3.2 `void _M_construct_node (_Node_* __p, _Tp const __V = _Tp(), _Base_ptr const __PARENT = NULL, _Base_ptr const __LEFT = NULL, _Base_ptr const __RIGHT = NULL) [inline, protected]`

6.1.3.3 `void _M_deallocate_node (_Node_* const __P) [inline, protected]`

6.1.3.4 `void _M_destroy_node (_Node_* __p) [inline, protected]`

6.1.3.5 `allocator_type get_allocator () const [inline]`

Reimplemented in `KDTree< __K, _Val, _Acc, _Dist, _Cmp, _Alloc >` (p. 194).

## 6.1.4 Member Data Documentation

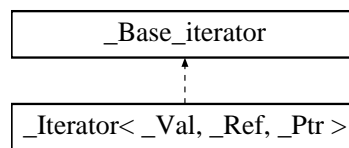
6.1.4.1 `allocator_type _M_node_allocator [protected]`

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

- `kdtree.h`

## 6.2 `_Base_iterator` Class Reference

`#include <inc/dtUtil/kdtree.h>`Inheritance diagram for `_Base_iterator`:



### Protected Types

- typedef `_Node_base::_Base_const_ptr` `_Base_const_ptr`

### Protected Member Functions

- `_Base_iterator` (`_Base_iterator` const &\_\_THAT)
- `_Base_iterator` (`_Base_const_ptr` const \_\_N=NULL)
- void `_M_decrement` ()
- void `_M_increment` ()

### Protected Attributes

- `_Base_const_ptr` `_M_node`

### Friends

- class `KDTree`

#### 6.2.1 Member Typedef Documentation

6.2.1.1 typedef `_Node_base::_Base_const_ptr` `_Base_const_ptr` [protected]

#### 6.2.2 Constructor & Destructor Documentation

6.2.2.1 `_Base_iterator` (`_Base_const_ptr` const \_\_N= NULL) [inline, protected]

6.2.2.2 `_Base_iterator` (`_Base_iterator` const &\_\_THAT) [inline, protected]

#### 6.2.3 Member Function Documentation

6.2.3.1 void `_M_decrement` () [inline, protected]

6.2.3.2 void `_M_increment` () [inline, protected]

#### 6.2.4 Friends And Related Function Documentation

6.2.4.1 friend class `KDTree` [friend]

#### 6.2.5 Member Data Documentation

6.2.5.1 `_Base_const_ptr` `_M_node` [protected]

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

- `kdtree.h`

## 6.3 `_Bracket_accessor<_Val >` Struct Template Reference

```
#include <inc/dtUtil/kdtree.h>
```

### Public Types

- `typedef _Val::value_type result_type`

### Public Member Functions

- `result_type operator() (_Val const &V, size_t const N) const`

```
template<typename _Val> struct dtUtil::_Bracket_accessor<_Val >
```

#### 6.3.1 Member Typedef Documentation

6.3.1.1 `typedef _Val::value_type result_type`

#### 6.3.2 Member Function Documentation

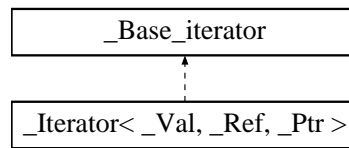
6.3.2.1 `result_type operator() (_Val const & V, size_t const N) const` [`inline`]

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

- `kdtree.h`

## 6.4 `_Iterator<_Val, _Ref, _Ptr>` Class Template Reference

`#include <inc/dtUtil/kdtree.h>`Inheritance diagram for `_Iterator<_Val, _Ref, _Ptr>::`



### Public Types

- typedef `_Node<_Val>` const \* `_Link_const_type`
- typedef `_Iterator<_Val, _Ref, _Ptr>` `_Self`
- typedef `_Iterator<_Val, _Val const &, _Val const * >` `const_iterator`
- typedef ptrdiff\_t `difference_type`
- typedef `_Iterator<_Val, _Val &, _Val * >` `iterator`
- typedef std::bidirectional\_iterator\_tag `iterator_category`
- typedef `_Ptr` `pointer`
- typedef `_Ref` `reference`
- typedef `_Val` `value_type`

### Public Member Functions

- `_Iterator` (`iterator` const &\_\_THAT)
- `_Iterator` (`_Link_const_type` const \_\_N)
- `_Iterator` ()
- `_Link_const_type` `get_raw_node` () const
- `reference` `operator*` () const
- `_Self` `operator++` (int)
- `_Self` `operator++` ()
- `_Self` `operator--` (int)
- `_Self` & `operator--` ()
- `pointer` `operator->` () const

### Friends

- bool `operator!=` (`_Iterator<_Val, _Val &, _Val * >` const &, `_Iterator<_Val, const _Val &, const _Val * >` const &)
- bool `operator!=` (`_Iterator<_Val, const _Val &, const _Val * >` const &, `_Iterator<_Val, _Val &, _Val * >` const &)
- bool `operator!=` (`_Iterator<_Val, _Ref, _Ptr >` const &, `_Iterator<_Val, _Ref, _Ptr >` const &)
- bool `operator==` (`_Iterator<_Val, _Val &, _Val * >` const &, `_Iterator<_Val, const _Val &, const _Val * >` const &)
- bool `operator==` (`_Iterator<_Val, const _Val &, const _Val * >` const &, `_Iterator<_Val, _Val &, _Val * >` const &)
- bool `operator==` (`_Iterator<_Val, _Ref, _Ptr >` const &, `_Iterator<_Val, _Ref, _Ptr >` const &)

```
template<typename _Val, typename _Ref, typename _Ptr> class dtUtil:: Iterator< _Val, _Ref, _Ptr >
```

### 6.4.1 Member Typedef Documentation

6.4.1.1 typedef `_Node<_Val> const* _Link_const_type`

6.4.1.2 typedef `_Iterator<_Val, _Ref, _Ptr> _Self`

6.4.1.3 typedef `_Iterator<_Val, _Val const&, _Val const*> const_iterator`

6.4.1.4 typedef `ptrdiff_t difference_type`

6.4.1.5 typedef `_Iterator<_Val, _Val&, _Val*> iterator`

6.4.1.6 typedef `std::bidirectional_iterator_tag iterator_category`

6.4.1.7 typedef `_Ptr pointer`

6.4.1.8 typedef `_Ref reference`

6.4.1.9 typedef `_Val value_type`

### 6.4.2 Constructor & Destructor Documentation

6.4.2.1 `_Iterator () [inline]`

6.4.2.2 `_Iterator (_Link_const_type const __N) [inline]`

6.4.2.3 `_Iterator (iterator const & __THAT) [inline]`

### 6.4.3 Member Function Documentation

6.4.3.1 `_Link_const_type get_raw_node () const [inline]`

6.4.3.2 `reference operator* () const [inline]`

6.4.3.3 `_Self operator++ (int) [inline]`

6.4.3.4 `_Self operator++ () [inline]`

6.4.3.5 `_Self operator-- (int) [inline]`

6.4.3.6 `_Self& operator-- () [inline]`

6.4.3.7 `pointer operator-> () const [inline]`

### 6.4.4 Friends And Related Function Documentation

6.4.4.1 `bool operator!= (_Iterator<_Val, _Val &, _Val * > const & __X, _Iterator<_Val, const _Val &, const _Val * > const & __Y) [friend]`

6.4.4.2 `bool operator!= (_Iterator<_Val, const _Val &, const _Val * > const & __X, _Iterator<_Val, _Val &, _Val * > const & __Y) [friend]`

6.4.4.3 `bool operator!= (_Iterator<_Val, _Ref, _Ptr > const & __X, _Iterator<_Val, _Ref, _Ptr > const & __Y) [friend]`

6.4.4.4 `bool operator== (_Iterator<_Val, _Val &, _Val * > const & __X, _Iterator<_Val, const _Val &, const _Val * > const & __Y) [friend]`

6.4.4.5 `bool operator== (_Iterator<_Val, const _Val &, const _Val * > const & __X, _Iterator<_Val, _Val &, _Val * > const & __Y) [friend]`

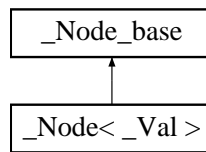
6.4.4.6 `bool operator== (_Iterator<_Val, _Ref, _Ptr > const & __X, _Iterator<_Val, _Ref, _Ptr > const & __Y) [friend]`

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

- `kdtree.h`

## 6.5 `_Node<_Val>` Struct Template Reference

`#include <inc/dtUtil/kdtree.h>`Inheritance diagram for `_Node<_Val>`:



### Public Types

- `typedef _Node_base * _Base_ptr`
- `typedef _Node * _Link_type`

### Public Member Functions

- `_Node (_Val const &__VALUE=_Val(), _Base_ptr const __PARENT=NULL, _Base_ptr const __LEFT=NULL, _Base_ptr const __RIGHT=NULL)`

### Public Attributes

- `_Val _M_value`

```
template<typename _Val> struct dtUtil::_Node<_Val>
```

#### 6.5.1 Member Typedef Documentation

6.5.1.1 `typedef _Node_base* _Base_ptr`

6.5.1.2 `typedef _Node* _Link_type`

#### 6.5.2 Constructor & Destructor Documentation

6.5.2.1 `_Node (_Val const &__VALUE =_Val(), _Base_ptr const __PARENT = NULL, _Base_ptr const __LEFT = NULL, _Base_ptr const __RIGHT = NULL) [inline]`

#### 6.5.3 Member Data Documentation

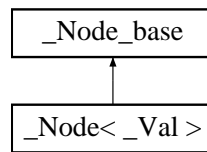
6.5.3.1 `_Val _M_value`

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

- `kdtree.h`

## 6.6 `_Node_base` Struct Reference

`#include <inc/dtUtil/kdtree.h>`Inheritance diagram for `_Node_base`:



### Public Types

- typedef `_Node_base` const \* `_Base_const_ptr`
- typedef `_Node_base` \* `_Base_ptr`

### Public Member Functions

- `_Node_base` (`_Base_ptr` const `__PARENT=`NULL, `_Base_ptr` const `__LEFT=`NULL, `_Base_ptr` const `__RIGHT=`NULL)

### Static Public Member Functions

- static `_Base_ptr` `_S_maximum` (`_Base_ptr` `__x`)
- static `_Base_ptr` `_S_minimum` (`_Base_ptr` `__x`)

### Public Attributes

- `_Base_ptr` `_M_left`
- `_Base_ptr` `_M_parent`
- `_Base_ptr` `_M_right`

#### 6.6.1 Member Typedef Documentation

6.6.1.1 typedef `_Node_base` const\* `_Base_const_ptr`

6.6.1.2 typedef `_Node_base`\* `_Base_ptr`

#### 6.6.2 Constructor & Destructor Documentation

6.6.2.1 `_Node_base` (`_Base_ptr` const `__PARENT=`NULL, `_Base_ptr` const `__LEFT=`NULL, `_Base_ptr` const `__RIGHT=`NULL) [`inline`]

#### 6.6.3 Member Function Documentation

6.6.3.1 static `_Base_ptr` `_S_maximum` (`_Base_ptr` `__x`) [`inline`, `static`]

6.6.3.2 static `_Base_ptr` `_S_minimum` (`_Base_ptr` `__x`) [`inline`, `static`]

#### 6.6.4 Member Data Documentation

6.6.4.1 `_Base_ptr` `_M_left`

6.6.4.2 `_Base_ptr` `_M_parent`

6.6.4.3 `_Base_ptr` `_M_right`

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

- `kdtree.h`

## 6.7 `_Node_compare<_Val, _Acc, _Cmp >` Class Template Reference

```
#include <inc/dtUtil/kdtree.h>
```

### Public Member Functions

- `_Node_compare` (`size_t` const `__DIM`, `_Acc` const & `acc`, `_Cmp` const & `cmp`)
- `bool operator()` (`_Val` const & `__A`, `_Val` const & `__B`) const

```
template<typename _Val, typename _Acc, typename _Cmp> class dtUtil::_Node_compare<_Val, _Acc, _Cmp >
```

### 6.7.1 Constructor & Destructor Documentation

6.7.1.1 `_Node_compare` (`size_t` const `__DIM`, `_Acc` const & `acc`, `_Cmp` const & `cmp`) [`inline`]

### 6.7.2 Member Function Documentation

6.7.2.1 `bool operator()` (`_Val` const & `__A`, `_Val` const & `__B`) const [`inline`]

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

- `kdtree.h`

## 6.8 `_Region< __K, _Val, _SubVal, _Acc, _Cmp >` Struct Template Reference

```
#include <inc/dtUtil/kdtree.h>
```

### Public Types

- typedef std::pair< `_Region`, `_SubVal` > `_CenterPt`
- typedef `_SubVal` `subvalue_type`
- typedef `_Val` `value_type`

### Public Member Functions

- template<typename Val >  
`_Region` (Val const &\_\_V, `subvalue_type` const &\_\_R, `_Acc` const &\_\_acc=`_Acc`(), const `_Cmp` &\_\_cmp=`_Cmp`())
- template<typename Val >  
`_Region` (Val const &\_\_V, `_Acc` const &\_\_acc=`_Acc`(), const `_Cmp` &\_\_cmp=`_Cmp`())
- `_Region` (`_Acc` const &\_\_acc=`_Acc`(), const `_Cmp` &\_\_cmp=`_Cmp`())
- bool `encloses` (`value_type` const &\_\_V) const
- bool `intersects_with` (`_Region` const &\_\_THAT) const
- bool `intersects_with` (`_CenterPt` const &\_\_THAT) const
- `_Region` & `set_high_bound` (`value_type` const &\_\_V, size\_t const \_\_L)
- `_Region` & `set_low_bound` (`value_type` const &\_\_V, size\_t const \_\_L)

### Public Attributes

- `_Acc` `_M_acc`
- `_Cmp` `_M_cmp`
- `subvalue_type` `_M_high_bounds` [`_K`]
- `subvalue_type` `_M_low_bounds` [`_K`]

```
template<size_t const __K, typename _Val, typename _SubVal, typename _Acc, typename _Cmp> struct
dtUtil::_Region< __K, _Val, _SubVal, _Acc, _Cmp >
```

### 6.8.1 Member Typedef Documentation

6.8.1.1 `typedef std::pair<_Region,_SubVal> _CenterPt`

6.8.1.2 `typedef _SubVal subvalue_type`

6.8.1.3 `typedef _Val value_type`

### 6.8.2 Constructor & Destructor Documentation

6.8.2.1 `_Region (_Acc const & __acc = _Acc(), const _Cmp & __cmp = _Cmp()) [inline]`

6.8.2.2 `_Region (Val const & __V, _Acc const & __acc = _Acc(), const _Cmp & __cmp = _Cmp()) [inline]`

6.8.2.3 `_Region (Val const & __V, subvalue_type const & __R, _Acc const & __acc = _Acc(), const _Cmp & __cmp = _Cmp()) [inline]`

### 6.8.3 Member Function Documentation

6.8.3.1 `bool encloses (value_type const & __V) const [inline]`

6.8.3.2 `bool intersects_with (_Region< __K, _Val, _SubVal, _Acc, _Cmp > const & __THAT) const [inline]`

6.8.3.3 `bool intersects_with (_CenterPt const & __THAT) const [inline]`

6.8.3.4 `_Region& set_high_bound (value_type const & __V, size_t const __L) [inline]`

6.8.3.5 `_Region& set_low_bound (value_type const & __V, size_t const __L) [inline]`

### 6.8.4 Member Data Documentation

6.8.4.1 `_Acc _M_acc`

6.8.4.2 `_Cmp _M_cmp`

6.8.4.3 `subvalue_type _M_high_bounds[__K]`

6.8.4.4 `subvalue_type _M_low_bounds[__K]`

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

- `kdtree.h`

## 6.9 always\_true<\_Tp > Struct Template Reference

```
#include <inc/dtUtil/kdtree.h>
```

### Public Member Functions

- `bool operator() (const _Tp &) const`

```
template<typename _Tp> struct dtUtil::always_true<_Tp >
```

### 6.9.1 Member Function Documentation

#### 6.9.1.1 `bool operator() (const _Tp &) const` [inline]

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

- `kdtree.h`

## 6.10 AppendTL< TList, T > Struct Template Reference

```
#include <inc/dtUtil/generic.h>
```

### Public Types

- typedef **TypeList**< typename TList::Head, typename **AppendTL**< typename TList::Tail, T >::Type > **Type**

```
template<class TList, class T> struct dtUtil::AppendTL< TList, T >
```

### 6.10.1 Member Typedef Documentation

#### 6.10.1.1 typedef TypeList<typename TList::Head, typename AppendTL<typename TList::Tail, T>::Type> Type

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

- **generic.h**

## 6.11 AppendTL< NullType, T > Struct Template Reference

```
#include <inc/dtUtil/generic.h>
```

### Public Types

- typedef **TypeList**< T, **NullType** > **Type**

```
template<class T> struct dtUtil::AppendTL< NullType, T >
```

### 6.11.1 Member Typedef Documentation

#### 6.11.1.1 typedef **TypeList**<T, **NullType**> **Type**

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

- **generic.h**

## 6.12 array\_remove< \_Container > Class Template Reference

```
#include <inc/dtUtil/templateutility.h>
```

### Public Types

- typedef \_Container **container\_type**
- typedef \_Container::reference **reference**

### Public Member Functions

- **array\_remove** (\_Container &\_Cont)
- **bool operator()** (typename \_Container::reference \_Val)

### Protected Attributes

- \_Container \* **container**

```
template<class _Container> class dtUtil::array_remove< _Container >
```

#### 6.12.1 Member Typedef Documentation

6.12.1.1 typedef \_Container container\_type

6.12.1.2 typedef \_Container::reference reference

#### 6.12.2 Constructor & Destructor Documentation

6.12.2.1 array\_remove (\_Container &\_Cont) [inline, explicit]

#### 6.12.3 Member Function Documentation

6.12.3.1 bool operator() (typename \_Container::reference \_Val) [inline]

#### 6.12.4 Member Data Documentation

6.12.4.1 \_Container\* container [protected]

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

- templateutility.h

## 6.13 AttributeSearch Class Reference

Searches a Xerces XML Attribute list for names of interest.

```
#include <inc/dtUtil/xercesutils.h>
```

### Public Types

- typedef std::map< std::string, std::string > **ResultMap**

### Public Member Functions

- **AttributeSearch ()**
- **~AttributeSearch ()**
- **ResultMap operator()** (const XERCES\_CPP\_NAMESPACE\_QUALIFIER Attributes &attrs)  
*Process the Attributes and make an easier structure to use.*

#### 6.13.1 Detailed Description

Searches a Xerces XML Attribute list for names of interest. A helper used when searching with a known string. e.g. "mystring", "Type", "Name", or "Value". For example:

```
void myHandler::startElement(... Attributes& attrs)
{
    AttributeSearch attSearch;
    AttributeSearch::ResultMap results = attSearch(attrs)
    AttributeSearch::ResultMap::iterator itr = results.find("Name");
    if (itr != results.end())
    {
        std::cout << "name is: " << itr->second << std::endl;
    }
}
```

#### 6.13.2 Member Typedef Documentation

##### 6.13.2.1 typedef std::map<std::string,std::string> ResultMap

#### 6.13.3 Constructor & Destructor Documentation

##### 6.13.3.1 AttributeSearch ()

##### 6.13.3.2 ~AttributeSearch ()

#### 6.13.4 Member Function Documentation

##### 6.13.4.1 AttributeSearch::ResultMap operator() (const XERCES\_CPP\_NAMESPACE\_QUALIFIER Attributes & attrs)

Process the Attributes and make an easier structure to use. Returns produces an associative container indexed on the attribute names.

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

- **xercesutils.h**
- **xercesutils.cpp**

## 6.14 BarycentricSpace< VecT > Class Template Reference

Transforms Cartesian data points into Barycentric coordinate systems.

```
#include <inc/dtUtil/barycentric.h>
```

### Public Member Functions

- **BarycentricSpace** (const VecT &a, const VecT &b, const VecT &c)
- **VecT Transform** (const VecT &p)  
*transform the data point into the barycentric system.*

### 6.14.1 Detailed Description

```
template<class VecT> class dtUtil::BarycentricSpace< VecT >
```

Transforms Cartesian data points into Barycentric coordinate systems. [http://en.wikipedia.org/wiki/Barycentric\\_coordinates\\_%28mathematics%29](http://en.wikipedia.org/wiki/Barycentric_coordinates_%28mathematics%29)

### 6.14.2 Constructor & Destructor Documentation

**6.14.2.1 BarycentricSpace (const VecT & a, const VecT & b, const VecT & c)**

### 6.14.3 Member Function Documentation

**6.14.3.1 VecT Transform (const VecT & p)**

transform the data point into the barycentric system. Parameters

**p** the data point to be transformed into the Barycentric coordinate system.

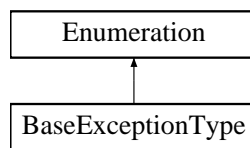
Returns the Barycentric coordinate.

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

- **barycentric.h**

## 6.15 BaseExceptionType Class Reference

#include <inc/dtUtil/exception.h>Inheritance diagram for BaseExceptionType::



### Static Public Attributes

- static **BaseExceptionType** GENERAL\_EXCEPTION

### Protected Member Functions

- **BaseExceptionType** (const std::string &name)

### 6.15.1 Constructor & Destructor Documentation

6.15.1.1 **BaseExceptionType** (const std::string & *name*) [inline, protected]

### 6.15.2 Member Data Documentation

6.15.2.1 **BaseExceptionType** GENERAL\_EXCEPTION [static]

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

- **exception.h**
- **exception.cpp**

## 6.16 Binder< Incoming, BoundIdsTL > Class Template Reference

```
#include <inc/dtUtil/funbind.h>
```

### Public Types

- typedef **BoundHelper**< Incoming, BoundIdsTL >::BoundTL **BoundParamsTL**
- typedef CallParms< **BoundParamsTL** >::ParmsListType **BoundPTL**
- typedef **Functor**< typename Incoming::ResultType, **UnboundParamsTL** > **Outgoing**
- typedef **UnboundHelper**< Incoming, BoundIdsTL >::Parm1 **Parm1**
- typedef **UnboundHelper**< Incoming, BoundIdsTL >::Parm2 **Parm2**
- typedef **UnboundHelper**< Incoming, BoundIdsTL >::Parm3 **Parm3**
- typedef **UnboundHelper**< Incoming, BoundIdsTL >::Parm4 **Parm4**
- typedef **UnboundHelper**< Incoming, BoundIdsTL >::Parm5 **Parm5**
- typedef Incoming::ResultType **ResultType**
- typedef Incoming::TypeListType **TypeListType**
- typedef **UnboundHelper**< Incoming, BoundIdsTL >::UnboundTL **UnboundParamsTL**
- typedef CallParms< **UnboundParamsTL** >::ParmsListType **UnboundPTL**

### Public Member Functions

- **Binder** (Incoming fun, **BoundPTL** bound)
- **Binder** ()
- **ResultType operator()** (**Parm1** p1, **Parm2** p2, **Parm3** p3, **Parm4** p4, **Parm5** p5) const
- **ResultType operator()** (**Parm1** p1, **Parm2** p2, **Parm3** p3, **Parm4** p4) const
- **ResultType operator()** (**Parm1** p1, **Parm2** p2, **Parm3** p3) const
- **ResultType operator()** (**Parm1** p1, **Parm2** p2) const
- **ResultType operator()** (**Parm1** p1) const
- **ResultType operator()** () const

template<typename Incoming, typename BoundIdsTL> class dtUtil::Binder< Incoming, BoundIdsTL >

### 6.16.1 Member Typedef Documentation

6.16.1.1 typedef BoundHelper<Incoming, BoundIdsTL>::BoundTL BoundParamsTL

6.16.1.2 typedef CallParams<BoundParamsTL>::ParamsListType BoundPTL

6.16.1.3 typedef Functor<typename Incoming::ResultType, UnboundParamsTL> Outgoing

6.16.1.4 typedef UnboundHelper<Incoming, BoundIdsTL>::Parm1 Parm1

6.16.1.5 typedef UnboundHelper<Incoming, BoundIdsTL>::Parm2 Parm2

6.16.1.6 typedef UnboundHelper<Incoming, BoundIdsTL>::Parm3 Parm3

6.16.1.7 typedef UnboundHelper<Incoming, BoundIdsTL>::Parm4 Parm4

6.16.1.8 typedef UnboundHelper<Incoming, BoundIdsTL>::Parm5 Parm5

6.16.1.9 typedef Incoming::ResultType ResultType

6.16.1.10 typedef Incoming::TypeListType TypeListType

6.16.1.11 typedef UnboundHelper<Incoming, BoundIdsTL>::UnboundTL UnboundParamsTL

6.16.1.12 typedef CallParams<UnboundParamsTL>::ParamsListType UnboundPTL

### 6.16.2 Constructor & Destructor Documentation

6.16.2.1 Binder () [inline]

6.16.2.2 Binder (Incoming *fun*, BoundPTL *bound*) [inline]

### 6.16.3 Member Function Documentation

6.16.3.1 ResultType operator() (Parm1 *p1*, Parm2 *p2*, Parm3 *p3*, Parm4 *p4*, Parm5 *p5*) const [inline]

6.16.3.2 ResultType operator() (Parm1 *p1*, Parm2 *p2*, Parm3 *p3*, Parm4 *p4*) const [inline]

6.16.3.3 ResultType operator() (Parm1 *p1*, Parm2 *p2*, Parm3 *p3*) const [inline]

6.16.3.4 ResultType operator() (Parm1 *p1*, Parm2 *p2*) const [inline]

6.16.3.5 ResultType operator() (Parm1 *p1*) const [inline]

6.16.3.6 ResultType operator() () const [inline]

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

- funbind.h

## 6.17 Bound Struct Reference

```
#include <inc/dtUtil/funbind.h>
```

### Static Public Member Functions

- static **ResultType MergeParms** (BoundPTL const &bound, UnboundPTL const &unbound)

```
template<int i, class BoundPTL, class UnboundPTL, class IdsTL, class TL> struct dtUtil::MergeParmsH<i, BoundPTL, UnboundPTL, IdsTL, TL >::Bound
```

### 6.17.1 Member Function Documentation

**6.17.1.1 static ResultType MergeParms (BoundPTL const & *bound*, UnboundPTL const & *unbound*)**  
[inline, static]

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

- **funbind.h**

## 6.18 BoundHelper< Incoming, IdsTL > Struct Template Reference

```
#include <inc/dtUtil/funbind.h>
```

### Public Types

- typedef **BoundTL2**< **IncomingTL**, **IdsTL** >::Result **BoundTL**
- typedef Incoming::TypeListType **IncomingTL**
- typedef dtUtil::TypeAtNonStrict< **BoundTL**, 0, dtUtil::NullType >::Result **Parm1**
- typedef dtUtil::TypeAtNonStrict< **BoundTL**, 1, dtUtil::NullType >::Result **Parm2**
- typedef dtUtil::TypeAtNonStrict< **BoundTL**, 2, dtUtil::NullType >::Result **Parm3**
- typedef dtUtil::TypeAtNonStrict< **BoundTL**, 3, dtUtil::NullType >::Result **Parm4**
- typedef dtUtil::TypeAtNonStrict< **BoundTL**, 4, dtUtil::NullType >::Result **Parm5**

```
template<class Incoming, class IdsTL> struct dtUtil::BoundHelper< Incoming, IdsTL >
```

### 6.18.1 Member Typedef Documentation

6.18.1.1 typedef **BoundTL2**<IncomingTL, IdsTL>::Result **BoundTL**

6.18.1.2 typedef Incoming::TypeListType **IncomingTL**

6.18.1.3 typedef dtUtil::TypeAtNonStrict<BoundTL, 0, dtUtil::NullType>::Result **Parm1**

6.18.1.4 typedef dtUtil::TypeAtNonStrict<BoundTL, 1, dtUtil::NullType>::Result **Parm2**

6.18.1.5 typedef dtUtil::TypeAtNonStrict<BoundTL, 2, dtUtil::NullType>::Result **Parm3**

6.18.1.6 typedef dtUtil::TypeAtNonStrict<BoundTL, 3, dtUtil::NullType>::Result **Parm4**

6.18.1.7 typedef dtUtil::TypeAtNonStrict<BoundTL, 4, dtUtil::NullType>::Result **Parm5**

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

- **funbind.h**

## 6.19 BoundingBoxVisitor Class Reference

```
#include <inc/dtUtil/boundingshapeutils.h>
```

### Public Member Functions

- **BoundingBoxVisitor** ()
- virtual void **apply** (osg::Geode &node)  
*Visits the specified geode.*

### Public Attributes

- osg::BoundingBox **mBoundingBox**  
*The aggregate bounding box.*

### 6.19.1 Constructor & Destructor Documentation

6.19.1.1 **BoundingBoxVisitor** () [inline]

### 6.19.2 Member Function Documentation

6.19.2.1 virtual void **apply** (osg::Geode & *node*) [inline, virtual]

Visits the specified geode. Parameters

*node* the geode to visit

### 6.19.3 Member Data Documentation

6.19.3.1 **osg::BoundingBox mBoundingBox**

The aggregate bounding box.

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

- **boundingshapeutils.h**

## 6.20 BoundTL2< TL, IdsTL > Struct Template Reference

```
#include <inc/dtUtil/funbind.h>
```

### Public Types

- typedef **Filter2**< TL, IdsTL, true >::Result Result

```
template<class TL, class IdsTL> struct dtUtil::BoundTL2< TL, IdsTL >
```

### 6.20.1 Member Typedef Documentation

#### 6.20.1.1 typedef Filter2<TL, IdsTL, true>::Result Result

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

- funbind.h

## 6.21 BreakOverride Struct Reference

This macro should be used when deprecating (or removing) virtual functions.

```
#include <inc/dtUtil/breakoverride.h>
```

### 6.21.1 Detailed Description

This macro should be used when deprecating (or removing) virtual functions. The effect is to force a compile error on client code that may be overriding the virtual function. Here is an example.

```
virtual void GetTransform(Transform& xform, CoordSysEnum cs = ABS_CS) co
nst;
//virtual void GetTransform(Transform* xform, CoordSysEnum cs = ABS_CS)
const;
private:
BREAK_OVERRIDE(GetTransform(const Transform*, CoordSysEnum))
```

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

- **breakoverride.h**

## 6.22 CallParms< TYPELIST\_0()> Struct Template Reference

```
#include <inc/dtUtil/funcall.h>
```

### Public Types

- typedef dtUtil::InstantiateH< dtUtil::NullType, dtUtil::TupleHolder > **ParmsListType**

### Static Public Member Functions

- static **ParmsListType** Make ()

```
template<> struct dtUtil::CallParms< TYPELIST_0()>
```

#### 6.22.1 Member Typedef Documentation

6.22.1.1 typedef dtUtil::InstantiateH<dtUtil::NullType, dtUtil::TupleHolder> **ParmsListType**

#### 6.22.2 Member Function Documentation

6.22.2.1 static **ParmsListType** Make () [inline, static]

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

- **funcall.h**

## 6.23 CallParms< TYPELIST\_1(P1)> Struct Template Reference

```
#include <inc/dtUtil/funcall.h>
```

### Public Types

- typedef dtUtil::InstantiateH< typename dtUtil::CreateTL< P1 >::Type, dtUtil::TupleHolder > **ParmsListType**

### Static Public Member Functions

- static **ParmsListType Make** (P1 p1)

```
template<typename P1> struct dtUtil::CallParms< TYPELIST_1(P1)>
```

#### 6.23.1 Member Typedef Documentation

6.23.1.1 typedef dtUtil::InstantiateH<typename dtUtil::CreateTL<P1>::Type, dtUtil::TupleHolder> **ParmsListType**

#### 6.23.2 Member Function Documentation

6.23.2.1 static **ParmsListType Make** (P1 *p1*) [*inline, static*]

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

- **funcall.h**

## 6.24 CallParms< TYPELIST\_2(P1, P2)> Struct Template Reference

```
#include <inc/dtUtil/funcall.h>
```

### Public Types

- typedef dtUtil::InstantiateH< typename dtUtil::CreateTL< P1, P2 >::Type, dtUtil::TupleHolder > ParamsListType

### Static Public Member Functions

- static ParamsListType Make (P1 p1, P2 p2)

```
template<typename P1, typename P2> struct dtUtil::CallParms< TYPELIST_2(P1, P2)>
```

#### 6.24.1 Member Typedef Documentation

6.24.1.1 typedef dtUtil::InstantiateH<typename dtUtil::CreateTL<P1, P2>::Type, dtUtil::TupleHolder> ParamsListType

#### 6.24.2 Member Function Documentation

6.24.2.1 static ParamsListType Make (P1 p1, P2 p2) [inline, static]

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

- funcall.h

## 6.25 CallParms< TYPELIST\_3(P1, P2, P3)> Struct Template Reference

```
#include <inc/dtUtil/funcall.h>
```

### Public Types

- typedef dtUtil::InstantiateH< typename dtUtil::CreateTL< P1, P2, P3 >::Type, dtUtil::TupleHolder > **ParmsListType**

### Static Public Member Functions

- static **ParmsListType Make** (P1 p1, P2 p2, P3 p3)

```
template<typename P1, typename P2, typename P3> struct dtUtil::CallParms< TYPELIST_3(P1, P2, P3)>
```

#### 6.25.1 Member Typedef Documentation

6.25.1.1 typedef dtUtil::InstantiateH<typename dtUtil::CreateTL<P1, P2, P3>::Type, dtUtil::TupleHolder> **ParmsListType**

#### 6.25.2 Member Function Documentation

6.25.2.1 static **ParmsListType Make** (P1 *p1*, P2 *p2*, P3 *p3*) [*inline*, *static*]

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

- **funcall.h**

## 6.26 CallParms< TYPELIST\_4(P1, P2, P3, P4)> Struct Template Reference

```
#include <inc/dtUtil/funcall.h>
```

### Public Types

- typedef dtUtil::InstantiateH< typename dtUtil::CreateTL< P1, P2, P3, P4 >::Type, dtUtil::TupleHolder > **ParmsListType**

### Static Public Member Functions

- static **ParmsListType Make** (P1 p1, P2 p2, P3 p3, P4 p4)

```
template<typename P1, typename P2, typename P3, typename P4> struct dtUtil::CallParms< TYPELIST_4(P1, P2, P3, P4)>
```

#### 6.26.1 Member Typedef Documentation

6.26.1.1 typedef dtUtil::InstantiateH<typename dtUtil::CreateTL<P1, P2, P3, P4>::Type, dtUtil::TupleHolder> **ParmsListType**

#### 6.26.2 Member Function Documentation

6.26.2.1 static **ParmsListType Make** (P1 *p1*, P2 *p2*, P3 *p3*, P4 *p4*) [*inline*, *static*]

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

- **funcall.h**

## 6.27 CallParms< TYPELIST\_5(P1, P2, P3, P4, P5)> Struct Template Reference

```
#include <inc/dtUtil/funcall.h>
```

### Public Types

- typedef dtUtil::InstantiateH< typename dtUtil::CreateTL< P1, P2, P3, P4, P5 >::Type, dtUtil::TupleHolder > **ParmsListType**

### Static Public Member Functions

- static **ParmsListType Make** (P1 p1, P2 p2, P3 p3, P4 p4, P5 p5)

```
template<typename P1, typename P2, typename P3, typename P4, typename P5> struct  
dtUtil::CallParms< TYPELIST_5(P1, P2, P3, P4, P5)>
```

#### 6.27.1 Member Typedef Documentation

6.27.1.1 typedef dtUtil::InstantiateH<typename dtUtil::CreateTL<P1, P2, P3, P4, P5>::Type, dtUtil::TupleHolder> **ParmsListType**

#### 6.27.2 Member Function Documentation

6.27.2.1 static **ParmsListType Make** (P1 *p1*, P2 *p2*, P3 *p3*, P4 *p4*, P5 *p5*) [*inline*, *static*]

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

- **funcall.h**

## 6.28 CallParms< TYPELIST\_6(P1, P2, P3, P4, P5, P6)> Struct Template Reference

```
#include <inc/dtUtil/funcall.h>
```

### Public Types

- typedef dtUtil::InstantiateH< typename dtUtil::CreateTL< P1, P2, P3, P4, P5, P6 >::Type, dtUtil::TupleHolder > ParmsListType

### Static Public Member Functions

- static ParmsListType Make (P1 p1, P2 p2, P3 p3, P4 p4, P5 p5, P6 p6)

```
template<typename P1, typename P2, typename P3, typename P4, typename P5, typename P6> struct dtUtil::CallParms< TYPELIST_6(P1, P2, P3, P4, P5, P6)>
```

#### 6.28.1 Member Typedef Documentation

- 6.28.1.1 typedef dtUtil::InstantiateH<typename dtUtil::CreateTL<P1, P2, P3, P4, P5, P6>::Type, dtUtil::TupleHolder> ParmsListType

#### 6.28.2 Member Function Documentation

- 6.28.2.1 static ParmsListType Make (P1 *p1*, P2 *p2*, P3 *p3*, P4 *p4*, P5 *p5*, P6 *p6*) [inline, static]

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

- funcall.h

## 6.29 CallParms< TYPELIST\_7(P1, P2, P3, P4, P5, P6, P7)> Struct Template Reference

```
#include <inc/dtUtil/funcall.h>
```

### Public Types

- typedef dtUtil::InstantiateH< typename **dtUtil::CreateTL**< P1, P2, P3, P4, P5, P6, P7 >::Type, **dtUtil::TupleHolder** > **ParmsListType**

### Static Public Member Functions

- static **ParmsListType Make** (P1 p1, P2 p2, P3 p3, P4 p4, P5 p5, P6 p6, P7 p7)

```
template<typename P1, typename P2, typename P3, typename P4, typename P5, typename P6, typename P7> struct dtUtil::CallParms< TYPELIST_7(P1, P2, P3, P4, P5, P6, P7)>
```

#### 6.29.1 Member Typedef Documentation

- 6.29.1.1 typedef dtUtil::InstantiateH<typename dtUtil::CreateTL<P1, P2, P3, P4, P5, P6, P7>::Type, dtUtil::TupleHolder> **ParmsListType**

#### 6.29.2 Member Function Documentation

- 6.29.2.1 static **ParmsListType Make** (P1 *p1*, P2 *p2*, P3 *p3*, P4 *p4*, P5 *p5*, P6 *p6*, P7 *p7*) [*inline*, *static*]

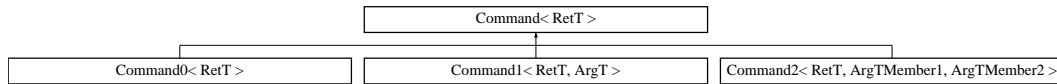
The documentation for this struct was generated from the following file:

- **funcall.h**

## 6.30 Command< RetT > Class Template Reference

An abstract class for all types which provide a uniform interface for executing Functors.

#include <inc/dtUtil/command.h> Inheritance diagram for Command< RetT >::



### Public Types

- typedef RetT **ReturnType**

### Public Member Functions

- **Command** ()
- virtual **ReturnType operator()** ()=0

### Protected Member Functions

- virtual **~Command** ()

#### 6.30.1 Detailed Description

**template<typename RetT> class dtUtil::Command< RetT >**

An abstract class for all types which provide a uniform interface for executing Functors. These classes will hold the client-defined arguments. This makes it easy to have a container of Commands which can be executed at a chosen time. This has been proven to be useful when catching GUI events and processing them during the System's preframe so that modifications to the graphics thread can be controlled.

The template parameter should be as follows:

- RetT The return type for the interface class AND function signature.

#### 6.30.2 Member Typedef Documentation

##### 6.30.2.1 typedef RetT ReturnType

#### 6.30.3 Constructor & Destructor Documentation

##### 6.30.3.1 Command () [inline]

##### 6.30.3.2 virtual ~Command () [inline, protected, virtual]

#### 6.30.4 Member Function Documentation

##### 6.30.4.1 virtual ReturnType operator() () [pure virtual]

Implemented in **Command0< RetT >** (p. 71), **Command1< RetT, ArgT >** (p. 72), and **Command2< RetT, ArgTMember1, ArgTMember2 >** (p. 75).

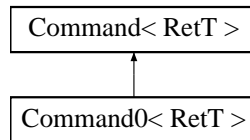
The documentation for this class was generated from the following file:

- **command.h**

## 6.31 Command0< RetT > Class Template Reference

A **Command** (p. 70) which does not need arguments and has no return value.

#include <inc/dtUtil/command.h> Inheritance diagram for Command0< RetT >::



### Public Types

- typedef dtUtil::Functor< RetT, TYPELIST\_0()> **FunctorType**

### Public Member Functions

- **Command0** (const **FunctorType** &f)
- RetT **operator()** ()

### Protected Member Functions

- ~**Command0** ()

#### 6.31.1 Detailed Description

template<typename RetT> class dtUtil::Command0< RetT >

A **Command** (p. 70) which does not need arguments and has no return value. The template parameter should be as follows:

- RetT The return type for the interface class AND function signature.

#### 6.31.2 Member Typedef Documentation

6.31.2.1 typedef dtUtil::Functor<RetT,TYPELIST\_0()> **FunctorType**

#### 6.31.3 Constructor & Destructor Documentation

6.31.3.1 **Command0** (const **FunctorType** & f) [inline]

6.31.3.2 ~**Command0** () [inline, protected]

#### 6.31.4 Member Function Documentation

6.31.4.1 RetT **operator()** () [inline, virtual]

Implements **Command**< RetT > (p. 70).

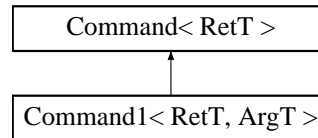
The documentation for this class was generated from the following file:

- **command.h**

## 6.32 Command1< RetT, ArgT > Class Template Reference

A **Command** (p. 70) which uses one argument and has no return value.

#include <inc/dtUtil/command.h> Inheritance diagram for Command1< RetT, ArgT >::



### Public Types

- typedef dtUtil::Functor< RetT, Types > **FunctorType**
- typedef details::TypeTraits< ArgT >::NonConstNoRef **MemberType**
- typedef **MemberType Param0**
- typedef **FunctorType::ParmsListType Params**

### Public Member Functions

- **Command1** (const **FunctorType** &f, **Param0** arg)
- RetT **operator**() ()
- typedef **TYPELIST\_1** (ArgT) Types

### Protected Member Functions

- ~**Command1** ()

#### 6.32.1 Detailed Description

template<typename RetT, typename ArgT> class dtUtil::Command1< RetT, ArgT >

A **Command** (p. 70) which uses one argument and has no return value. The template parameters should be as follows:

- RetT The return type for the interface class AND function signature.
- ArgTMember1 The type used to store the first parameter.

#### Todo

Use Loki::TypeTraits to eliminate the need for the ArgTMember template parameter.

#### 6.32.2 Member Typedef Documentation

6.32.2.1 typedef dtUtil::Functor<RetT,Types> **FunctorType**

6.32.2.2 typedef details::TypeTraits<ArgT>::NonConstNoRef **MemberType**

6.32.2.3 typedef **MemberType Param0**

6.32.2.4 typedef **FunctorType::ParmsListType Params**

#### 6.32.3 Constructor & Destructor Documentation

6.32.3.1 **Command1** (const **FunctorType** & f, **Param0** arg) [inline]

6.32.3.2 ~**Command1** () [inline, protected]

#### 6.32.4 Member Function Documentation

6.32.4.1 RetT **operator**() () [inline, virtual]

Implements **Command**< RetT > (p. 70).

#### 6.32.4.2 typedef TYPELIST\_1 (ArgT)

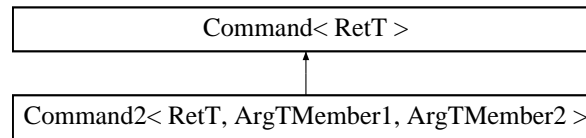
The documentation for this class was generated from the following file:

- `command.h`

## 6.33 Command2< RetT, ArgTMember1, ArgTMember2 > Class Template Reference

A **Command** (p. 70) which uses two arguments and has no return value.

```
#include <inc/dtUtil/command.h>
Inheritance diagram for Command2< RetT, ArgTMember1, ArgTMember2 >::
```



### Public Types

- typedef dtUtil::Functor< RetT, Types > **FunctorType**
- typedef details::TypeTraits< ArgTMember1 >::NonConstNoRef **MemberType1**
- typedef details::TypeTraits< ArgTMember2 >::NonConstNoRef **MemberType2**
- typedef **MemberType1** Param0
- typedef **MemberType2** Param1
- typedef dtUtil::CallParms< Types >::ParmsListType **Params**

### Public Member Functions

- **Command2** (const **FunctorType** &f, **Param0** arg1, **Param1** arg2)
- RetT **operator()** ()
- typedef **TYPELIST\_2** (**MemberType1**, **MemberType2**) Types

#### 6.33.1 Detailed Description

```
template<typename RetT, typename ArgTMember1, typename ArgTMember2> class dtUtil::Command2< RetT, ArgTMember1, ArgTMember2 >
```

A **Command** (p. 70) which uses two arguments and has no return value. The template parameters should be as follows:

- RetT The return type for the interface class AND function signature.
- ArgTMember1 The type used to store the first parameter.
- ArgTMember2 The type used to store the second parameter.

#### 6.33.2 Member Typedef Documentation

6.33.2.1 typedef dtUtil::Functor<RetT,Types> **FunctorType**

6.33.2.2 typedef details::TypeTraits<ArgTMember1>::NonConstNoRef **MemberType1**

6.33.2.3 typedef details::TypeTraits<ArgTMember2>::NonConstNoRef **MemberType2**

6.33.2.4 typedef **MemberType1** Param0

#### Todo

make these typedefs work! they are needed for the argument types in the ctor

#### Todo

take this out when the above is fixed

6.33.2.5 `typedef MemberType2 Param1`

6.33.2.6 `typedef dtUtil::CallParms<Types>::ParmsListType Params`

### 6.33.3 Constructor & Destructor Documentation

6.33.3.1 `Command2 (const FunctorType & f, Param0 arg1, Param1 arg2) [inline]`

### 6.33.4 Member Function Documentation

6.33.4.1 `RetT operator() () [inline, virtual]`

Implements `Command< RetT >` (p. 70).

6.33.4.2 `typedef TYPELIST_2 (MemberType1, MemberType2)`

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

- `command.h`

## 6.34 ConfigProperties Class Reference

Interface for having configuration properties.

```
#include <inc/dtUtil/configproperties.h>
```

### Public Member Functions

- virtual const std::string & **GetConfigPropertyValue** (const std::string &name, const std::string &default-Value="") const =0  
*Removes a property with the given name.*
- virtual void **RemoveConfigPropertyValue** (const std::string &name)=0  
*Sets the value of a given config property.*
- virtual void **SetConfigPropertyValue** (const std::string &name, const std::string &value)=0  
*Sets the value of a given config property.*

### 6.34.1 Detailed Description

Interface for having configuration properties.

### 6.34.2 Member Function Documentation

#### 6.34.2.1 virtual const std::string& GetConfigPropertyValue (const std::string & name, const std::string & defaultValue = "") const [pure virtual]

Returns a string value that is paired with the given name. The default is returned if the property is not set.

#### 6.34.2.2 virtual void RemoveConfigPropertyValue (const std::string & name) [pure virtual]

Removes a property with the given name.

#### 6.34.2.3 virtual void SetConfigPropertyValue (const std::string & name, const std::string & value) [pure virtual]

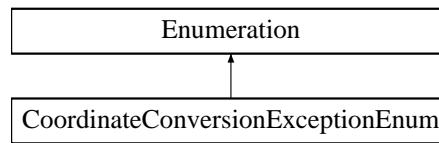
Sets the value of a given config property.

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

- **configproperties.h**

## 6.35 CoordinateConversionExceptionEnum Class Reference

#include <inc/dtUtil/coordinates.h> Inheritance diagram for CoordinateConversionExceptionEnum::



### Static Public Attributes

- static **CoordinateConversionExceptionEnum INVALID\_INPUT**  
*Thrown when the input to a conversion function is invalid.*

### 6.35.1 Member Data Documentation

#### 6.35.1.1 CoordinateConversionExceptionEnum INVALID\_INPUT [static]

Thrown when the input to a conversion function is invalid.

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

- **coordinates.h**
- **coordinates.cpp**

## 6.36 Coordinates Class Reference

```
#include <inc/dtUtil/coordinates.h>
```

### Public Member Functions

- **Coordinates** (const **Coordinates** &rhs)
- **Coordinates** ()
- virtual ~**Coordinates** ()
- osg::Vec3 **ConvertMGRSToXYZ** (const std::string &mgrs)
- const osg::Vec3 **ConvertToLocalRotation** (const osg::Vec3 &psiThetaPhi)
- const osg::Vec3 **ConvertToLocalRotation** (double psi, double theta, double phi)
 

*Converts psi theta phi coordinates in radians to a local rotation heading, pitch, roll in degrees based on the current configuration.*
- const osg::Vec3 **ConvertToLocalTranslation** (const osg::Vec3d &loc)
 

*Converts 3 part remote coordinates to a local translation vector based on the current configuration.*
- const osg::Vec3d **ConvertToRemoteRotation** (const osg::Vec3 &hpr)
 

*Converts hpr coordinates in degrees to a remote rotation psi, theta, phi in radians based on the current configuration.*
- const osg::Vec3d **ConvertToRemoteTranslation** (const osg::Vec3 &translation)
 

*Converts XYZ coordinates to a remote location vector based on the current configuration.*
- bool **GetApplyRotationConversionMatrix** () const
- void **GetFlatEarthOrigin** (osg::Vec2d &originLLOut) const
 

*Retrieves the origin point of references lat lon for the incoming coordinates.*
- float **GetGlobeRadius** () const
 

*returns the globe radius for globe local coordinates.*
- const **IncomingCoordinateType** & **GetIncomingCoordinateType** () const
- const **LocalCoordinateType** & **GetLocalCoordinateType** () const
- void **GetLocalOffset** (osg::Vec3d &offsetOut) const
 

*Retrieves local coordinate offset.*
- float **GetMagneticNorthOffset** () const
 

*Non static accessor to the magnetic north offset variable of this class.*
- const osg::Matrix & **GetOriginRotationMatrix** () const
- const osg::Matrix & **GetOriginRotationMatrixInverse** () const
- char **GetUTMHemisphere** () const
- unsigned **GetUTMZone** () const
- **Coordinates** & **operator=** (const **Coordinates** &rhs)
- bool **operator==** (const **Coordinates** &rhs) const
- void **ReconfigureRotationMatrix** ()
 

*Called to recompute the rotation conversion matrix based on the configuration.*
- void **SetApplyRotationConversionMatrix** (bool applyMatrix)
 

*Sets whether or not the rotation needs to be converted via the origin rotation matrix.*
- void **SetFlatEarthOrigin** (const osg::Vec2d &originLL)
 

*Sets an origin point of reference lat lon for the incoming coordinates.*
- void **SetGlobeRadius** (float radius)
 

*Sets the globe radius for globe local coordinates.*

- void **SetIncomingCoordinateType** (const **IncomingCoordinateType** &incomingCoordType)
- void **SetLocalCoordinateType** (const **LocalCoordinateType** &localCoordType)
- void **SetLocalOffset** (const osg::Vec3d &offset)  
*Applies a simple offset to the local coordinates when converting.*
- void **SetMagneticNorthOffset** (float magNorth)  
*Non static method to set the magnetic north offset variable on this class.*
- void **SetUTMHemisphere** (char hemisphere)  
*Set the UTM hemishere to be used when converting coodinates from cartesian (Assumed to be UTM) to lat/lon.*
- void **SetUTMLocalOffsetAsLatLon** (const osg::Vec3d &lle)  
*Sets the location of the game space origin in lat lon and converts it to an offset in UTM.*
- void **SetUTMZone** (unsigned zone)  
*Set the UTM zone to be used when converting coodinates from cartesian (Assumed to be UTM) to lat/lon.*
- std::string **XYZToMGRS** (const osg::Vec3 &pos)

### Static Public Member Functions

- static double **CalculateConvergencParamForFlatEarth** (double latitude)
- static float **CalculateMagneticNorthOffset** (const float latitude, const float longitude)  
*Adjusts for magnetic north of the earth.*
- static void **CalculateUTMZone** (double latitude, double longitude, unsigned &wZone, char &nsZone)  
*Calculates the proper UTM zone based on the latitude and longitude.*
- static void **ConvertFlatEarthToLatLon** (osg::Vec3d &lle, const osg::Vec3d &xyz, const osg::Vec2d &originll, double convergenceParam)
- static void **ConvertGeocentricToGeodetic** (double x, double y, double z, double &phi, double &lambda, double &elevation)  
*The function ConvertGeocentricToGeodetic converts geocentric coordinates (X, Y, Z) to geodetic coordinates (latitude, longitude, and height), according to the current ellipsoid parameters.*
- static void **ConvertGeodeticToTransverseMercator** (const **UTMParameters** &params, double Latitude, double Longitude, double &Easting, double &Northing)  
*The function ConvertGeodeticToTransverse\_Mercator converts geodetic (latitude and longitude) coordinates to Transverse Mercator projection (easting and northing) coordinates, according to the current ellipsoid and Transverse Mercator projection coordinates.*
- static void **ConvertGeodeticToUTM** (double Latitude, double Longitude, unsigned Zone, char Hemisphere, double &Easting, double &Northing)  
*The function ConvertGeodeticToUTM converts geodetic (latitude and longitude) coordinates to UTM projection (zone, hemisphere, easting and northing) coordinates according to the current ellipsoid and UTM zone override parameters.*
- static void **ConvertLatLonToFlatEarth** (osg::Vec3d &xyz, const osg::Vec3d &lle, const osg::Vec2d &originll, double convergenceParam)
- static void **ConvertMGRSToUTM** (unsigned defaultZone, char defaultZoneLetter, const std::string &mgrs, unsigned &zone, double &easting, double &northing)
- static void **ConvertTransverseMercatorToGeodetic** (const **UTMParameters** &params, double Easting, double Northing, double &Latitude, double &Longitude)
- static void **ConvertUTMToGeodetic** (unsigned zone, char hemisphere, double easting, double northing, double &latitude, double &longitude)  
*The function ConvertUTMToGeodetic converts UTM projection (zone, easting and northing) to geodetic (latitude and longitude) coordinates according to the current ellipsoid and UTM zone override parameters.*

- static const std::string **ConvertUTMToMGRS** (double easting, double northing, unsigned eastWestZone, char northSouthZone, unsigned resolution)  
*Converts UTM coordinates to MGRS coordinates and returns the value as a string.*
- static unsigned int **DegreesToMils** (const float degrees)  
*Converts degrees to mils.*
- static void **EulersToMatrix** (osg::Matrix &dst, float psi, float theta, float phi)  
*Creates a 4x4 rotation matrix from a set of DIS/RPR-FOM Euler angles.*
- static void **GeocentricToGeodetic** (double x, double y, double z, double &latitude, double &longitude, double &elevation)  
*Converts a set of geocentric coordinates to the equivalent geodetic coordinates.*
- static void **GeodeticToGeocentric** (double phi, double lambda, double elevation, double &x, double &y, double &z)  
*Converts a set of geodetic coordinates to the equivalent geocentric coordinates.*
- static void **MatrixToEulers** (const osg::Matrix &src, float &psi, float &theta, float &phi)  
*Returns the DIS/RPR-FOM Euler angles corresponding to the given rotation matrix.*
- static float **MilsToDegrees** (const unsigned int mils)  
*Converts mils to degrees.*
- static void **ZFlop** (osg::Matrix &toFlop)

### 6.36.1 Constructor & Destructor Documentation

#### 6.36.1.1 Coordinates ()

#### 6.36.1.2 ~Coordinates () [virtual]

#### 6.36.1.3 Coordinates (const Coordinates & rhs) [inline]

### 6.36.2 Member Function Documentation

#### 6.36.2.1 double CalculateConvergencParamForFlatEarth (double *latitude*) [static]

#### 6.36.2.2 float CalculateMagneticNorthOffset (const float *latitude*, const float *longitude*) [static]

Adjusts for magnetic north of the earth. Parameters

***latitude*** the latitude in DEGREES

***longitude*** the longitude in DEGREES

Returns The rotation offset in DEGREES

#### 6.36.2.3 void CalculateUTMZone (double *latitude*, double *longitude*, unsigned & *ewZone*, char & *nsZone*) [static]

Calculates the proper UTM zone based on the latitude and longitude. Parameters

***latitude*** the latitude position for the zone.

***longitude*** the longitude position for the zone.

***ewZone*** the east west zone number. This the normal UTM zone used in calculations. ***nsZone*** the north-south zone letter.

**6.36.2.4 void ConvertFlatEarthToLatLon (osg::Vec3d & *lle*, const osg::Vec3d & *xyz*, const osg::Vec2d & *originll*, double *convergenceParam*) [static]**

**6.36.2.5 void ConvertGeocentricToGeodetic (double *x*, double *y*, double *z*, double & *phi*, double & *lambda*, double & *elevation*) [static]**

The function ConvertGeocentricToGeodetic converts geocentric coordinates (X, Y, Z) to geodetic coordinates (latitude, longitude, and height), according to the current ellipsoid parameters. Code taken from <http://earth-info.nga.mil/GandG/geotrans/>

Parameters

***x*** : Geocentric X coordinate, in meters. (input)

***y*** : Geocentric Y coordinate, in meters. (input)

***z*** : Geocentric Z coordinate, in meters. (input)

***phi*** : Calculated latitude value in radians. (output)

***lambda*** : Calculated longitude value in radians. (output)

***elevation*** : Calculated height value, in meters. (output)

The method used here is derived from 'An Improved Algorithm for Geocentric to Geodetic Coordinate Conversion', by Ralph Toms, Feb 1996

**6.36.2.6 void ConvertGeodeticToTransverseMercator (const UTMParameters & *params*, double *Latitude*, double *Longitude*, double & *Easting*, double & *Northing*) [static]**

The function ConvertGeodeticToTransverse\_Mercator converts geodetic (latitude and longitude) coordinates to Transverse Mercator projection (easting and northing) coordinates, according to the current ellipsoid and Transverse Mercator projection coordinates. Parameters

***params*** : UTM Parameter values.

***Latitude*** : Latitude in radians (input)

***Longitude*** : Longitude in radians (input)

***Easting*** : Easting/X in meters (output)

***Northing*** : Northing/Y in meters (output)

**6.36.2.7 void ConvertGeodeticToUTM (double *Latitude*, double *Longitude*, unsigned *Zone*, char *Hemisphere*, double & *Easting*, double & *Northing*) [static]**

The function ConvertGeodeticToUTM converts geodetic (latitude and longitude) coordinates to UTM projection (zone, hemisphere, easting and northing) coordinates according to the current ellipsoid and UTM zone override parameters. The UTM Zone and hemisphere must be passed in so that the code knows which zone to use as the point of reference. Call CalculateUTMZone if you want to know the "home" zone of the lat lon before calling this. Code taken from <http://earth-info.nga.mil/GandG/geotrans/>

Parameters

***Latitude*** : Latitude in radians (input)

***Longitude*** : Longitude in radians (input)

***Zone*** : UTM zone (input)

***Hemisphere*** : North or South hemisphere (input)

***Easting*** : Easting (X) in meters (output)

***Northing*** : Northing (Y) in meters (output)

**6.36.2.8 void ConvertLatLonToFlatEarth (osg::Vec3d & xyz, const osg::Vec3d & lle, const osg::Vec2d & originll, double convergenceParam) [static]**

**6.36.2.9 void ConvertMGRSToUTM (unsigned defaultZone, char defaultZoneLetter, const std::string & mgrs, unsigned & zone, double & easting, double & northing) [static]**

Exceptions

**dtUtil::Exception** (p. 116) if the string is not in a valid format.

**6.36.2.10 osg::Vec3 ConvertMGRSToXYZ (const std::string & mgrs)**

**6.36.2.11 const osg::Vec3 ConvertToLocalRotation (const osg::Vec3 & psiThetaPhi) [inline]**

**6.36.2.12 const osg::Vec3 ConvertToLocalRotation (double psi, double theta, double phi)**

Converts psi theta phi coordinates in radians to a local rotation heading, pitch, roll in degrees based on the current configuration.

**6.36.2.13 const osg::Vec3 ConvertToLocalTranslation (const osg::Vec3d & loc)**

Converts 3 part remote coordinates to a local translation vector based on the current configuration. Parameters

**loc** the location as 3 doubles.

Returns the x y z location in local space.

**6.36.2.14 const osg::Vec3d ConvertToRemoteRotation (const osg::Vec3 & hpr)**

Converts hpr coordinates in degrees to a remote rotation psi, theta, phi in radians based on the current configuration.

**6.36.2.15 const osg::Vec3d ConvertToRemoteTranslation (const osg::Vec3 & translation)**

Converts XYZ coordinates to a remote location vector based on the current configuration. Parameters

**translation** the local x,y,z translation to convert.

Returns the location as a vec3d

**6.36.2.16 void ConvertTransverseMercatorToGeodetic (const UTMParameters & params, double Easting, double Northing, double & Latitude, double & Longitude) [static]**

**6.36.2.17 void ConvertUTMToGeodetic (unsigned zone, char hemisphere, double easting, double northing, double & latitude, double & longitude) [static]**

The function ConvertUTMToGeodetic converts UTM projection (zone, easting and northing) to geodetic (latitude and longitude) coordinates according to the current ellipsoid and UTM zone override parameters. Parameters

**zone** : UTM zone (east west) (input)

**hemisphere** : UTM hemisphere ('N' or 'S') (input)

**easting** : Easting (X) in meters (input)

**northing** : Northing (Y) in meters (input)

**latitude** : Latitude in radians (output)

**longitude** : Longitude in radians (output)

**6.36.2.18 const std::string ConvertUTMToMGRS (double *easting*, double *northing*, unsigned *eastWestZone*, char *northSouthZone*, unsigned *resolution*) [static]**

Converts UTM coordinates to MGRS coordinates and returns the value as a string. Parameters

***easting*** the UTM easting value

***northing*** the UTM northing value

***eastWestZone*** the east/west zone number (1 - 60)

***northSouthZone*** the north/south zone letter (A-Z omitting I and O)

***resolution*** the resolution number. It should be 0-5 for resolutions of 100000 meters to 1 meter in powers of 10.

**6.36.2.19 unsigned int DegreesToMils (const float *degrees*) [static]**

Converts degrees to mils. Parameters

***degrees*** The degrees to convert

**6.36.2.20 void EulersToMatrix (osg::Matrix & *dst*, float *psi*, float *theta*, float *phi*) [static]**

Creates a 4x4 rotation matrix from a set of DIS/RPR-FOM Euler angles. Parameters

***dst*** the destination matrix

***psi*** the psi angle

***theta*** the theta angle

***phi*** the phi angle

**6.36.2.21 void GeocentricToGeodetic (double *x*, double *y*, double *z*, double & *latitude*, double & *longitude*, double & *elevation*) [static]**

Converts a set of geocentric coordinates to the equivalent geodetic coordinates. Uses the formula given at [http://www.colorado.edu/geography/gcraft/notes/datum/datum\\_f.html](http://www.colorado.edu/geography/gcraft/notes/datum/datum_f.html).

Parameters

***x*** the geocentric x coordinate

***y*** the geocentric y coordinate

***z*** the geocentric z coordinate

***latitude*** the location in which to store the geodetic latitude

***longitude*** the location in which to store the geodetic longitude

***elevation*** the location in which to store the geodetic elevation

**6.36.2.22 void GeodeticToGeocentric (double *phi*, double *lambda*, double *elevation*, double & *x*, double & *y*, double & *z*) [static]**

Converts a set of geodetic coordinates to the equivalent geocentric coordinates. Uses the formula given at [http://www.colorado.edu/geography/gcraft/notes/datum/datum\\_f.html](http://www.colorado.edu/geography/gcraft/notes/datum/datum_f.html).

Parameters

***phi*** the geodetic latitude in radians

***lambda*** the geodetic longitude in radians

***elevation*** the geodetic elevation

***x*** the location in which to store the geocentric x coordinate ***y*** the location in which to store the geocentric y coordinate

***z*** the location in which to store the geocentric z coordinate

**6.36.2.23 bool GetApplyRotationConversionMatrix () const [inline]**

Returns true if when converting the rotation, the origin rotation matrix is applied. Otherwise rotation is assumed to need no conversion.

**6.36.2.24 void GetFlatEarthOrigin (osg::Vec2d & originLLOut) const**

Retrieves the origin point of references lat lon for the incoming coordinates. Parameters

*originOut* Fill this with the origin

**6.36.2.25 float GetGlobeRadius () const**

returns the globe radius for globe local coordinates. Returns the current globe radius

**6.36.2.26 const IncomingCoordinateType& GetIncomingCoordinateType () const [inline]****6.36.2.27 const LocalCoordinateType& GetLocalCoordinateType () const [inline]****6.36.2.28 void GetLocalOffset (osg::Vec3d & offsetOut) const**

Retrieves local coordinate offset. Parameters

*offsetOut* Fill this with the offset

**6.36.2.29 float GetMagneticNorthOffset () const [inline]**

Non static accessor to the magnetic north offset variable of this class. This is in degrees and should be ADDED to the heading angle. This is just a point of storage. None of the methods in this class use this angle. Returns mMagneticNorthOffset

**6.36.2.30 const osg::Matrix & GetOriginRotationMatrix () const****6.36.2.31 const osg::Matrix & GetOriginRotationMatrixInverse () const****6.36.2.32 char GetUTMHemisphere () const [inline]**

Returns the currently set UTM hemisphere.

**6.36.2.33 unsigned GetUTMZone () const [inline]**

Returns the currently set UTM zone.

**6.36.2.34 void MatrixToEulers (const osg::Matrix & src, float & psi, float & theta, float & phi) [static]**

Returns the DIS/RPR-FOM Euler angles corresponding to the given rotation matrix. Parameters

*src* the source matrix

*psi* the location in which to store the psi angle

*theta* the location in which to store the theta angle

*phi* the location in which to store the phi angle

**6.36.2.35 float MilsToDegrees (const unsigned int mils) [static]**

Converts mils to degrees. Parameters

*mils* The mils to convert

Note If mils is greater than 6400 then it will be clamped to 6400 implicitly

**6.36.2.36 Coordinates & operator= (const Coordinates & rhs)****6.36.2.37 bool operator== (const Coordinates & rhs) const****6.36.2.38 void ReconfigureRotationMatrix ()**

Called to recompute the rotation conversion matrix based on the configuration.

**6.36.2.39 void SetApplyRotationConversionMatrix (bool *applyMatrix*) [inline]**

Sets whether or not the rotation needs to be converted via the origin rotation matrix.

**6.36.2.40 void SetFlatEarthOrigin (const osg::Vec2d & *originLL*)**

Sets an origin point of reference lat lon for the incoming coordinates. Parameters

***offset*** the new offset

**6.36.2.41 void SetGlobeRadius (float *radius*)**

Sets the globe radius for globe local coordinates. Parameters

***radius*** the new radius

**6.36.2.42 void SetIncomingCoordinateType (const IncomingCoordinateType & *incomingCoordType*)****6.36.2.43 void SetLocalCoordinateType (const LocalCoordinateType & *localCoordType*)****6.36.2.44 void SetLocalOffset (const osg::Vec3d & *offset*)**

Applies a simple offset to the local coordinates when converting. Parameters

***offset*** the new offset

**6.36.2.45 void SetMagneticNorthOffset (float *magNorth*) [inline]**

Non static method to set the magnetic north offset variable on this class. Parameters

***magNorth*** The new offset to set

**6.36.2.46 void SetUTMHemisphere (char *hemisphere*)**

Set the UTM hemisphere to be used when converting coordinates from cartesian (Assumed to be UTM) to lat/lon. Parameters

***hemisphere*** the utm hemisphere. It must be 'N' or 'S' or lowercase. Anything else will just be assigned as 'N'

**6.36.2.47 void SetUTMLocalOffsetAsLatLon (const osg::Vec3d & *lle*)**

Sets the location of the game space origin in lat lon and converts it to an offset in UTM. It also sets the utm zone and rotation conversion.

Parameters

***lle*** latitude, longitude, and elevation.

We just recomputed the rotation, so its not dirty.

**6.36.2.48 void SetUTMZone (unsigned *zone*)**

Set the UTM zone to be used when converting coordinates from cartesian (Assumed to be UTM) to lat/lon. Parameters

***zone*** the utm zone. If it's not between 1 and 60 inclusive, it will be clamped.

**6.36.2.49 std::string XYZToMGRS (const osg::Vec3 & *pos*)****6.36.2.50 void ZFlop (osg::Matrix & *toFlop*) [static]**

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

- **coordinates.h**
- **coordinates.cpp**

## 6.37 CreateldsTL< i1, i2, i3, i4, i5, i6, i7, i8 > Struct Template Reference

```
#include <inc/dtUtil/generic.h>
```

### Public Types

- typedef **TypeList< Int2Type< i1 >, typename CreateldsTL< i2, i3, i4, i5, i6, i7, i8, -1 >::Type > Type**

```
template<int i1 = -1, int i2 = -1, int i3 = -1, int i4 = -1, int i5 = -1, int i6 = -1, int i7 = -1, int i8 = -1> struct dtUtil::CreateldsTL< i1, i2, i3, i4, i5, i6, i7, i8 >
```

### 6.37.1 Member Typedef Documentation

#### 6.37.1.1 typedef TypeList<Int2Type<i1>, typename CreateldsTL<i2, i3, i4, i5, i6, i7, i8, -1>::Type> Type

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

- **generic.h**

## 6.38 CreateIdsTL<-1,-1,-1,-1,-1,-1,-1,-1 > Struct Template Reference

```
#include <inc/dtUtil/generic.h>
```

### Public Types

- typedef **NullType** Type

```
template<> struct dtUtil::CreateIdsTL<-1,-1,-1,-1,-1,-1,-1,-1 >
```

### 6.38.1 Member Typedef Documentation

#### 6.38.1.1 typedef NullType Type

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

- **generic.h**

## 6.39 CreateTL< T1, T2, T3, T4, T5, T6, T7, T8 > Struct Template Reference

```
#include <inc/dtUtil/generic.h>
```

### Public Types

- typedef **TypeList**< T1, typename **CreateTL**< T2, T3, T4, T5, T6, T7, T8 >::Type > **Type**

```
template<class T1 = NullType, class T2 = NullType, class T3 = NullType, class T4 = NullType, class T5 = NullType, class T6 = NullType, class T7 = NullType, class T8 = NullType> struct dtUtil::CreateTL< T1, T2, T3, T4, T5, T6, T7, T8 >
```

### 6.39.1 Member Typedef Documentation

#### 6.39.1.1 typedef TypeList<T1, typename CreateTL<T2, T3, T4, T5, T6, T7, T8>::Type> Type

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

- **generic.h**

## 6.40 CreateTL< NullType, NullType, NullType, NullType, NullType, NullType, NullType, NullType > Struct Template Reference

```
#include <inc/dtUtil/generic.h>
```

### Public Types

- typedef **NullType** Type

```
template<> struct dtUtil::CreateTL< NullType, NullType, NullType, NullType, NullType, NullType, Null-  
Type, NullType >
```

### 6.40.1 Member Typedef Documentation

#### 6.40.1.1 typedef NullType Type

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

- **generic.h**

## 6.41 DataStream Class Reference

```
#include <inc/dtUtil/datastream.h>
```

### Classes

- class **SeekTypeEnum**

### Public Member Functions

- **DataStream** (const **DataStream** &rhs)
- **DataStream** (char \*buffer, unsigned int bufferSize, bool autoFree=true)  
*Constructs the datastream using an existing byte buffer.*
- **DataStream** ()
- virtual ~**DataStream** ()
- unsigned int **AppendDataStream** (const **DataStream** &dataStream)
- unsigned int **ClearBuffer** ()
- const char \* **GetBuffer** () const
- unsigned int **GetBufferCapacity** () const
- unsigned int **GetBufferSize** () const
- unsigned int **GetRemainingReadSize** ()
- unsigned int **IncreaseBufferSize** (const unsigned int size=0)
- bool **IsLittleEndian** () const  
*Gets the endian'ness of the current platform.*
- **DataStream** & **operator**<< (const osg::Vec4d &value)
- **DataStream** & **operator**<< (const osg::Vec4f &value)
- **DataStream** & **operator**<< (const osg::Vec3d &value)
- **DataStream** & **operator**<< (const osg::Vec3f &value)
- **DataStream** & **operator**<< (const osg::Vec2d &value)
- **DataStream** & **operator**<< (const osg::Vec2f &value)
- **DataStream** & **operator**<< (const std::string &value)
- **DataStream** & **operator**<< (long long value)
- **DataStream** & **operator**<< (double value)
- **DataStream** & **operator**<< (float value)
- **DataStream** & **operator**<< (unsigned long value)
- **DataStream** & **operator**<< (long value)
- **DataStream** & **operator**<< (unsigned value)
- **DataStream** & **operator**<< (int value)
- **DataStream** & **operator**<< (unsigned short value)
- **DataStream** & **operator**<< (short value)
- **DataStream** & **operator**<< (char value)
- **DataStream** & **operator**<< (unsigned char value)
- **DataStream** & **operator**<< (bool value)
- **DataStream** & **operator**= (const **DataStream** &rhs)
- **DataStream** & **operator**>> (osg::Vec4d &value)
- **DataStream** & **operator**>> (osg::Vec4f &value)
- **DataStream** & **operator**>> (osg::Vec3d &value)
- **DataStream** & **operator**>> (osg::Vec3f &value)
- **DataStream** & **operator**>> (osg::Vec2d &value)
- **DataStream** & **operator**>> (osg::Vec2f &value)
- **DataStream** & **operator**>> (std::string &value)
- **DataStream** & **operator**>> (long long &value)
- **DataStream** & **operator**>> (double &value)
- **DataStream** & **operator**>> (float &value)
- **DataStream** & **operator**>> (unsigned long &value)

- **DataStream & operator>>** (long &value)
- **DataStream & operator>>** (unsigned &value)
- **DataStream & operator>>** (int &value)
- **DataStream & operator>>** (unsigned short &value)
- **DataStream & operator>>** (short &value)
- **DataStream & operator>>** (char &value)
- **DataStream & operator>>** (unsigned char &value)
- **DataStream & operator>>** (bool &value)
- void **Read** (osg::Vec4d &vector)
- void **Read** (osg::Vec4f &vector)
- void **Read** (osg::Vec3d &vector)
- void **Read** (osg::Vec3f &vector)
- void **Read** (osg::Vec2d &vector)
- void **Read** (osg::Vec2f &vector)
- void **Read** (std::string &string)
- void **Read** (long long &d)
- void **Read** (double &d)
- void **Read** (float &f)
- void **Read** (unsigned long &i)
- void **Read** (long &i)
- void **Read** (unsigned &i)
- void **Read** (int &i)
- void **Read** (unsigned short &s)
- void **Read** (short &s)
- void **Read** (char &c)
- void **Read** (unsigned char &c)
- void **Read** (bool &c)
- unsigned int **ReadBinary** (char \*pBuffer, const unsigned int isize)
- void **Rewind** ()
- void **Seekg** (unsigned int offset, const **SeekTypeEnum** &type)
- void **Seekp** (unsigned int offset, const **SeekTypeEnum** &type)
- unsigned int **SetBufferSize** (unsigned int size)
- void **SetForceLittleEndian** (bool force)

*Forces the stream to interpret its data contents as little endian.*

- void **Write** (const osg::Vec4d &vector)
- void **Write** (const osg::Vec4f &vector)
- void **Write** (const osg::Vec3d &vector)
- void **Write** (const osg::Vec3f &vector)
- void **Write** (const osg::Vec2d &vector)
- void **Write** (const osg::Vec2f &vector)
- void **Write** (const std::string &string)
- void **Write** (long long d)
- void **Write** (double d)
- void **Write** (float f)
- void **Write** (unsigned long i)
- void **Write** (long i)
- void **Write** (unsigned i)
- void **Write** (int i)
- void **Write** (unsigned short s)
- void **Write** (short s)
- void **Write** (char c)
- void **Write** (unsigned char c)
- void **Write** (bool c)
- unsigned int **WriteBinary** (const char \*pBuffer, const unsigned int isize)
- void **WriteBytes** (unsigned char c, size\_t count)

## 6.41.1 Constructor & Destructor Documentation

### 6.41.1.1 `DataStream ()`

### 6.41.1.2 `DataStream (char * buffer, unsigned int bufferSize, bool autoFree = true)`

Constructs the datastream using an existing byte buffer. Parameters

***buffer*** The existing valid buffer.

***bufferSize*** The size in bytes of the buffer.

***reverseByteSwapping*** If this flag is true, the buffer will treat its contents as little endian by default instead of big endian. This is useful if working with file data where the file is always in little endian format.

***autoFree*** If true, the buffer's memory is freed when the **`DataStream`** (p. 90) instance gets destructed. If false, the caller is responsible for freeing the associated buffer memory.

### 6.41.1.3 `DataStream (const DataStream & rhs)`

### 6.41.1.4 `~DataStream () [virtual]`

## 6.41.2 Member Function Documentation

### 6.41.2.1 `unsigned int AppendDataStream (const DataStream & dataStream)`

### 6.41.2.2 `unsigned int ClearBuffer ()`

### 6.41.2.3 `const char* GetBuffer () const [inline]`

### 6.41.2.4 `unsigned int GetBufferCapacity () const [inline]`

### 6.41.2.5 `unsigned int GetBufferSize () const [inline]`

### 6.41.2.6 `unsigned int GetRemainingReadSize ()`

### 6.41.2.7 `unsigned int IncreaseBufferSize (const unsigned int size = 0)`

### 6.41.2.8 `bool IsLittleEndian () const [inline]`

Gets the endian'ness of the current platform. Returns True if little endian, false if big endian.

- 6.41.2.9 DataStream& operator<< (const osg::Vec4d & *value*) [inline]
- 6.41.2.10 DataStream& operator<< (const osg::Vec4f & *value*) [inline]
- 6.41.2.11 DataStream& operator<< (const osg::Vec3d & *value*) [inline]
- 6.41.2.12 DataStream& operator<< (const osg::Vec3f & *value*) [inline]
- 6.41.2.13 DataStream& operator<< (const osg::Vec2d & *value*) [inline]
- 6.41.2.14 DataStream& operator<< (const osg::Vec2f & *value*) [inline]
- 6.41.2.15 DataStream& operator<< (const std::string & *value*) [inline]
- 6.41.2.16 DataStream& operator<< (long long *value*) [inline]
- 6.41.2.17 DataStream& operator<< (double *value*) [inline]
- 6.41.2.18 DataStream& operator<< (float *value*) [inline]
- 6.41.2.19 DataStream& operator<< (unsigned long *value*) [inline]
- 6.41.2.20 DataStream& operator<< (long *value*) [inline]
- 6.41.2.21 DataStream& operator<< (unsigned *value*) [inline]
- 6.41.2.22 DataStream& operator<< (int *value*) [inline]
- 6.41.2.23 DataStream& operator<< (unsigned short *value*) [inline]
- 6.41.2.24 DataStream& operator<< (short *value*) [inline]
- 6.41.2.25 DataStream& operator<< (char *value*) [inline]
- 6.41.2.26 DataStream& operator<< (unsigned char *value*) [inline]
- 6.41.2.27 DataStream& operator<< (bool *value*) [inline]
- 6.41.2.28 DataStream & operator= (const DataStream & *rhs*)
- 6.41.2.29 DataStream& operator>> (osg::Vec4d & *value*) [inline]
- 6.41.2.30 DataStream& operator>> (osg::Vec4f & *value*) [inline]
- 6.41.2.31 DataStream& operator>> (osg::Vec3d & *value*) [inline]
- 6.41.2.32 DataStream& operator>> (osg::Vec3f & *value*) [inline]
- 6.41.2.33 DataStream& operator>> (osg::Vec2d & *value*) [inline]
- 6.41.2.34 DataStream& operator>> (osg::Vec2f & *value*) [inline]
- 6.41.2.35 DataStream& operator>> (std::string & *value*) [inline]
- 6.41.2.36 DataStream& operator>> (long long & *value*) [inline]
- 6.41.2.37 DataStream& operator>> (double & *value*) [inline]
- 6.41.2.38 DataStream& operator>> (float & *value*) [inline]
- 6.41.2.39 DataStream& operator>> (unsigned long & *value*) [inline]
- 6.41.2.40 DataStream& operator>> (long & *value*) [inline]
- 6.41.2.41 DataStream& operator>> (unsigned & *value*) [inline]
- 6.41.2.42 DataStream& operator>> (int & *value*) [inline]
- 6.41.2.43 DataStream& operator>> (unsigned short & *value*) [inline]
- 6.41.2.44 DataStream& operator>> (short & *value*) [inline]
- 6.41.2.45 DataStream& operator>> (char & *value*) [inline]
- 6.41.2.46 DataStream& operator>> (unsigned char & *value*) [inline]
- 6.41.2.47 DataStream& operator>> (bool & *value*) [inline]
- 6.41.2.48 void Read (osg::Vec4d & *vector*)
- 6.41.2.49 void Read (osg::Vec4f & *vector*)
- 6.41.2.50 void Read (osg::Vec3d & *vector*)
- 6.41.2.51 void Read (osg::Vec3f & *vector*)

**force** True if the stream should interpret its contents as little endian.

Note This is useful when working with binary files that are gaurenteed to be little endian. The particular file loading code can therefore, read the file contents into the data stream and on big endian machines, byte swapping will occur automatically.

6.41.2.73 void Write (const osg::Vec4d & *vector*)

6.41.2.74 void Write (const osg::Vec4f & *vector*)

6.41.2.75 void Write (const osg::Vec3d & *vector*)

6.41.2.76 void Write (const osg::Vec3f & *vector*)

6.41.2.77 void Write (const osg::Vec2d & *vector*)

6.41.2.78 void Write (const osg::Vec2f & *vector*)

6.41.2.79 void Write (const std::string & *string*)

6.41.2.80 void Write (long long *d*)

6.41.2.81 void Write (double *d*)

6.41.2.82 void Write (float *f*)

6.41.2.83 void Write (unsigned long *i*)

6.41.2.84 void Write (long *i*)

6.41.2.85 void Write (unsigned *i*)

6.41.2.86 void Write (int *i*)

6.41.2.87 void Write (unsigned short *s*)

6.41.2.88 void Write (short *s*)

6.41.2.89 void Write (char *c*)

6.41.2.90 void Write (unsigned char *c*)

6.41.2.91 void Write (bool *c*)

6.41.2.92 unsigned int WriteBinary (const char \* *pBuffer*, const unsigned int *isize*)

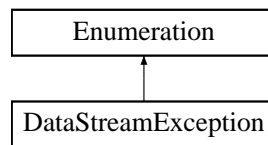
6.41.2.93 void WriteBytes (unsigned char *c*, size\_t *count*)

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

- `datastream.h`
- `datastream.cpp`

## 6.42 DataStreamException Class Reference

#include <inc/dtUtil/datastream.h>Inheritance diagram for DataStreamException::



### Static Public Attributes

- static **DataStreamException** BUFFER\_INVALID
- static **DataStreamException** BUFFER\_INVALID\_POS
- static **DataStreamException** BUFFER\_READ\_ERROR
- static **DataStreamException** BUFFER\_WRITE\_ERROR

### 6.42.1 Member Data Documentation

6.42.1.1 **DataStreamException** BUFFER\_INVALID [static]

6.42.1.2 **DataStreamException** BUFFER\_INVALID\_POS [static]

6.42.1.3 **DataStreamException** BUFFER\_READ\_ERROR [static]

6.42.1.4 **DataStreamException** BUFFER\_WRITE\_ERROR [static]

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

- **datastream.h**
- **datastream.cpp**

## 6.43 DateTime Class Reference

```
#include <inc/dtUtil/datetime.h>
```

### Classes

- class **TimeFormat**  
The **TimeFormat** (p. 278) enumeration is used to specify how to map a **DateTime** (p. 96) object to a string.
- class **TimeOrigin**  
The **TimeOrigin** (p. 279) enumeration determines how the instance of **DateTime** (p. 96) should be interpreted.
- class **TimeType**  
The **TimeType** (p. 280) enumeration is used to identify the purpose of a **DateTime** (p. 96) instance.

### Public Member Functions

- **DateTime** (const **DateTime** &)
- **DateTime** (const struct tm &)  
create a **DateTime** (p. 96) using the c standard struct tm
- **DateTime** (time\_t)  
create a **DateTime** (p. 96) using the c standard time\_t struct
- **DateTime** (const **TimeOrigin** &initAs)  
This constructor takes a **TimeOrigin** (p. 279) and sets the time accordingly.
- **DateTime** ()  
The default constructor just zeros.
- virtual ~**DateTime** ()
- void **AdjustTimeZone** (float newGMTOffset)  
Every date time has a GMT offset.
- unsigned **GetDay** () const
- float **GetGMTOffset** () const  
Gets the GMTOffset, this will be 0 unless **SetGMTOffset** was called or **SetToLocalTime()** (p. 102) was called, or **TimeOrigin::LOCAL\_TIME** (p. 279) was fed into the constructor.
- void **GetGMTTime** (tm &) const  
fills a standard c struct tm with the time but adds GMTOffset into time before struct tm is calculated
- time\_t **GetGMTTime** () const  
Gets the time in standard time\_t format, the GMTOffset is added to the time before calculating time\_t.
- void **GetGMTTime** (unsigned &year, unsigned &month, unsigned &day, unsigned &hour, unsigned &min, unsigned &sec) const
- void **GetGMTTime** (unsigned &year, unsigned &month, unsigned &day, unsigned &hour, unsigned &min, float &sec) const  
Gets the time internally stored using unsigned year, month, day, hour, minute, and seconds.
- unsigned **GetHour** () const
- unsigned **GetMinute** () const
- unsigned **GetMonth** () const
- float **GetSecond** () const
- void **GetTime** (tm &) const

*fills a standard c struct tm with the time*

- **time\_t GetTime () const**  
*Gets the time in the standard time\_t format, specified as seconds elapsed since midnight, January 1, 1970.*
- **void GetTime (unsigned &year, unsigned &month, unsigned &day, unsigned &hour, unsigned &min, unsigned &sec) const**
- **void GetTime (unsigned &year, unsigned &month, unsigned &day, unsigned &hour, unsigned &min, float &sec) const**  
*Gets the time internally stored using unsigned year, month, day, hour, minute, and seconds.*
- **const TimeFormat & GetTimeFormat () const**  
*The TimeFormat (p. 278) enumeration is used to specify how to map a DateTime (p. 96) object to a string.*
- **double GetTimeInSeconds () const**  
*returns the total clock time in seconds elapsed since midnight, January 1, 1970 this time includes fractional seconds*
- **const TimeOrigin & GetTimeOrigin () const**  
*The TimeOrigin (p. 279) enumeration determines how the instance of DateTime (p. 96) should be interpreted.*
- **float GetTimeScale () const**  
*The TimeScale can be used to scale the time when incrementing the clock.*
- **const TimeType & GetTimeType () const**  
*The TimeType (p. 280) enumeration is used to identify the purpose of a DateTime (p. 96) instance.*
- **unsigned GetYear () const**
- **void IncrementClock (double seconds)**  
*Increments the clock time by the number of seconds specified.*
- **operator std::string () const**
- **operator time\_t () const**
- **operator tm () const**
- **DateTime & operator= (const DateTime &)**
- **void SetDay (unsigned day)**
- **void SetGMTOffset (double latitude, double longitude, bool dayLightSavings)**  
*Calculates the GMTOffset using a latitude and longitude, not 100% correct due to regional time zone boundaries but it works ok as an approximation.*
- **void SetGMTOffset (tm &timeParts, bool factorLocalDayLightSavingsIntoGMTOffset)**  
*Sets the GMTOffset using the systems local time this offset is added to the time when getting GMT time,.*
- **void SetGMTOffset (float hourOffset, bool dayLightSavings)**  
*Sets the GMTOffset which is added to the time when getting GMT time,.*
- **void SetHour (unsigned hour)**
- **void SetMinute (unsigned min)**
- **void SetMonth (unsigned month)**
- **void SetSecond (float sec)**
- **void SetTime (const tm &)**  
*sets the time using the standard c struct tm*
- **void SetTime (time\_t)**  
*sets time in standard time\_t format, specified as seconds elapsed since midnight, January 1, 1970*
- **void SetTime (unsigned year, unsigned month, unsigned day, unsigned hour, unsigned min, unsigned sec)**
- **void SetTime (unsigned year, unsigned month, unsigned day, unsigned hour, unsigned min, float sec)**

*Sets the full time using year, month, day, hour, minute, and second.*

- void **SetTimeFormat** (const **TimeFormat** &)
- void **SetTimeOrigin** (const **TimeOrigin** &)
- void **SetTimeScale** (float percentScaleInSeconds)
- void **SetTimeType** (const **TimeType** &)
- void **SetToGMTTime** ()  
*changes time to be GMT- or Greenwich Mean Time*
- void **SetToLocalTime** ()  
*changes time to be system local time*
- void **SetYear** (unsigned year)
- std::string **Tostring** (const **TimeFormat** &) const  
*Converts the time to a string using a **TimeFormat** (p. 278) enumeration, see the **TimeFormat** (p. 278) enumeration above.*
- std::string **Tostring** () const  
*The no parameter version of ToString uses the internal **TimeFormat** (p. 278), see the **TimeFormat** (p. 278) enumeration above.*

## Static Public Member Functions

- static float **GetLocalGMTOffset** ()
- static std::string **Tostring** (const **DateTime** &, const **TimeFormat** &)  
*a static version of **Tostring()** (p. 102) for convenience, using the conversion operators a time\_t or struct tm can be passed for the **DateTime** (p. 96)*

## 6.43.1 Constructor & Destructor Documentation

### 6.43.1.1 DateTime ()

The default constructor just zeros.

### 6.43.1.2 DateTime (const TimeOrigin & initAs)

This constructor takes a **TimeOrigin** (p. 279) and sets the time accordingly.

### 6.43.1.3 DateTime (time\_t t)

create a **DateTime** (p. 96) using the c standard time\_t struct

### 6.43.1.4 DateTime (const struct tm & t)

create a **DateTime** (p. 96) using the c standard struct tm

### 6.43.1.5 DateTime (const DateTime & rhs)

### 6.43.1.6 ~DateTime () [virtual]

## 6.43.2 Member Function Documentation

### 6.43.2.1 void AdjustTimeZone (float newGMTOffset)

Every date time has a GMT offset. This method allows one to set a new GMT offset, and then change the clock internally to match the newoffset. For example if the clock is in 12:00 AM Eastern Time (-5) Jan 5, 2009, setting the offset to west coast time (-8) would make the date time be 9:00 PM, Jan 4, 2009. Parameters

**newGMTOffset** the offset in hours from GMT to change this to.

**6.43.2.2 unsigned GetDay () const****6.43.2.3 float GetGMTOffset () const**

Gets the GMTOffset, this will be 0 unless SetGMTOffset was called or **SetToLocalTime()** (p. 102) was called, or **TimeOrigin::LOCAL\_TIME** (p. 279) was fed into the constructor. Returns the current offset from Greenwich mean time

**6.43.2.4 void GetGMTTime (tm & *timeParts*) const**

fills a standard c struct tm with the time but adds GMTOffset into time before struct tm is calculated

**6.43.2.5 time\_t GetGMTTime () const**

Gets the time in standard time\_t format, the GMTOffset is added to the time before calculating time\_t.

**6.43.2.6 void GetGMTTime (unsigned & *year*, unsigned & *month*, unsigned & *day*, unsigned & *hour*, unsigned & *min*, unsigned & *sec*) const****6.43.2.7 void GetGMTTime (unsigned & *year*, unsigned & *month*, unsigned & *day*, unsigned & *hour*, unsigned & *min*, float & *sec*) const**

Gets the time internally stored using unsigned year, month, day, hour, minute, and seconds. The float second version will include the sub second time if **IncrementClock()** (p. 100) or **SetSecond()** (p. 101) was not rounded off. The GMTOffset will be added into the current time to obtain the GMT Time.

Parameters

**Year-** the full year, not just since 1900 epoch

**Month-** the month specified as 1-12

**Day-** the day specified as 1-31

**Hour-** hours since midnight 0-23

**Minute-** minutes after the hour 0-60

**Second-** seconds after the hour 0-61 (and extra second is added to support leap seconds)

**6.43.2.8 unsigned GetHour () const****6.43.2.9 float GetLocalGMTOffset () [static]****6.43.2.10 unsigned GetMinute () const****6.43.2.11 unsigned GetMonth () const****6.43.2.12 float GetSecond () const****6.43.2.13 void GetTime (tm & *timeParts*) const**

fills a standard c struct tm with the time

**6.43.2.14 time\_t GetTime () const**

Gets the time in the standard time\_t format, specified as seconds elapsed since midnight, January 1, 1970. Note this format only includes whole seconds

On many systems it is ambiguous if mktime() assumes the input is in GMT, or local timezone. To address this, a new function called timegm() is appearing. It works exactly like mktime() but explicitly interprets the input as GMT.

timegm() is available and documented under FreeBSD. It is available, but completely undocumented on my current Debian 2.1 distribution.

In the absence of timegm() we have to guess what mktime() might do.

Many older BSD style systems have a mktime() that assumes the input time in GMT. But FreeBSD explicitly states that mktime() assumes local time zone

The mktime() on many SYSV style systems (such as Linux) usually returns its result assuming you have specified the input time in your local timezone. Therefore, in the absence of timegm() you have to go to extra trouble to convert back to GMT.

If you are having problems with incorrectly positioned astronomical bodies, this is a really good place to start looking.

**6.43.2.15 void GetTime (unsigned & year, unsigned & month, unsigned & day, unsigned & hour, unsigned & min, unsigned & sec) const**

**6.43.2.16 void GetTime (unsigned & year, unsigned & month, unsigned & day, unsigned & hour, unsigned & min, float & sec) const**

Gets the time internally stored using unsigned year, month, day, hour, minute, and seconds. The float second version will include the sub second time if **IncrementClock()** (p. 100) or **SetSecond()** (p. 101) was not rounded off. To get time modified by a GMTOffset use **GetGMTTime()** (p. 99).

Parameters

**Year-** the full year, not just since 1900 epoch

**Month-** the month specified as 1-12

**Day-** the day specified as 1-31

**Hour-** hours since midnight 0-23

**Minute-** minutes after the hour 0-60

**Second-** seconds after the hour 0-61 (and extra second is added to support leap seconds)

**6.43.2.17 const DateTime::TimeFormat & GetTimeFormat () const**

The **TimeFormat** (p. 278) enumeration is used to specify how to map a **DateTime** (p. 96) object to a string. Set the **TimeFormat** (p. 278) if you would like to use the **ToString()** (p. 102) without any arguments.

**6.43.2.18 double GetTimeInSeconds () const**

returns the total clock time in seconds elapsed since midnight, January 1, 1970 this time includes fractional seconds

**6.43.2.19 const DateTime::TimeOrigin & GetTimeOrigin () const**

The **TimeOrigin** (p. 279) enumeration determines how the instance of **DateTime** (p. 96) should be interpreted.

**6.43.2.20 float GetTimeScale () const**

The **TimeScale** can be used to scale the time when incrementing the clock. The default value for **TimeScale** is 1.0.

**6.43.2.21 const DateTime::TimeType & GetTimeType () const**

The **TimeType** (p. 280) enumeration is used to identify the purpose of a **DateTime** (p. 96) instance.

**6.43.2.22 unsigned GetYear () const**

**6.43.2.23 void IncrementClock (double seconds)**

Increments the clock time by the number of seconds specified. The fractional part of the time is saved off and added in whole increments to support sub second times but getting time as a time\_t or struct tm will not include fractional seconds. To get the fractional seconds use GetTime with a float for seconds or **GetSecond()** (p. 99).

Parameters

**the** number of seconds to increase the clock by.

6.43.2.24 operator std::string () const

6.43.2.25 operator time\_t () const

6.43.2.26 operator tm () const

6.43.2.27 DateTime & operator= (const DateTime & rhs)

6.43.2.28 void SetDay (unsigned *day*)

6.43.2.29 void SetGMTOffset (double *latitude*, double *longitude*, bool *dayLightSavings*)

Calculates the GMTOffset using a latitude and longitude, not 100% correct due to regional time zone boundaries but it works ok as an approximation. Parameters

*latitude,the* geographical latitude of origin

*longitude,the* geographical longitude of origin

*dayLightSavings,this* flag is used to specify whether or not daylight savings is in effect setting this flag to true will add an extra hour to the GMTOffset

6.43.2.30 void SetGMTOffset (tm & *timeParts*, bool *factorLocalDayLightSavingsIntoGMTOffset*)

Sets the GMTOffset using the systems local time this offset is added to the time when getting GMT time,. Parameters

*timeParts,the* struct tm used to calculate the local timezone

*factorLocalDayLightSavingsIntoGMTOffset,this* flag is used to specify whether or not daylight savings should be obtained from the system clock, setting true will increment the GMTOffset by one hour if your system is currently on daylight savings.

6.43.2.31 void SetGMTOffset (float *hourOffset*, bool *dayLightSavings*)

Sets the GMTOffset which is added to the time when getting GMT time,. Parameters

*hourOffset,the* number of hours to offset from GMT Time, the param is a float to support half hour time zones.

*dayLightSavings,this* flag is used to specify whether or not daylight savings is in effect setting this flag to true will add an extra hour to the GMTOffset

6.43.2.32 void SetHour (unsigned *hour*)

6.43.2.33 void SetMinute (unsigned *min*)

6.43.2.34 void SetMonth (unsigned *month*)

6.43.2.35 void SetSecond (float *sec*)

6.43.2.36 void SetTime (const tm & *mt*)

sets the time using the standard c struct tm

6.43.2.37 void SetTime (time\_t *t*)

sets time in standard time\_t format, specified as seconds elapsed since midnight, January 1, 1970

6.43.2.38 void SetTime (unsigned *year*, unsigned *month*, unsigned *day*, unsigned *hour*, unsigned *min*, unsigned *sec*)

6.43.2.39 void SetTime (unsigned *year*, unsigned *month*, unsigned *day*, unsigned *hour*, unsigned *min*, float *sec*)

Sets the full time using year, month, day, hour, minute, and second. A float version is provided to support sub second times.

Parameters

**Year-** the full year, not just since 1900 epoch

**Month-** the month specified as 1-12

**Day-** the day specified as 1-31

**Hour-** hours since midnight 0-23

**Minute-** minutes after the hour 0-60

**Second-** seconds after the hour 0-61 (and extra second is added to support leap seconds)

**6.43.2.40 void SetTimeFormat (const TimeFormat & tf)**

**6.43.2.41 void SetTimeOrigin (const TimeOrigin & to)**

**6.43.2.42 void SetTimeScale (float percentScaleInSeconds)**

**6.43.2.43 void SetTimeType (const TimeType & tt)**

**6.43.2.44 void SetToGMTTime ()**

changes time to be GMT- or Greenwich Mean Time

**6.43.2.45 void SetToLocalTime ()**

changes time to be system local time

**6.43.2.46 void SetYear (unsigned year)**

**6.43.2.47 std::string ToString (const DateTime & dt, const TimeFormat & tf) [static]**

a static version of **ToString()** (p. 102) for convenience, using the conversion operators a `time_t` or `struct tm` can be passed for the **DateTime** (p. 96)

**6.43.2.48 std::string ToString (const TimeFormat & tf) const**

Converts the time to a string using a **TimeFormat** (p. 278) enumeration, see the **TimeFormat** (p. 278) enumeration above.

**6.43.2.49 std::string ToString () const**

The no parameter version of **ToString** uses the internal **TimeFormat** (p. 278), see the **TimeFormat** (p. 278) enumeration above.

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

- **datetime.h**
- **datetime.cpp**

## 6.44 DeprecatedFunction Struct Reference

### Public Attributes

- `std::set< int >` **CalledFrom**
- `const char *` **NewFunctionName**
- `const char *` **OldFunctionName**

### 6.44.1 Member Data Documentation

#### 6.44.1.1 `std::set<int>` CalledFrom

#### 6.44.1.2 `const char*` NewFunctionName

#### 6.44.1.3 `const char*` OldFunctionName

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

- `deprecationmgr.cpp`

## 6.45 DeprecationMgr Class Reference

Used to deprecate a method.

```
#include <inc/dtUtil/deprecationmgr.h>
```

### Public Member Functions

- **~DeprecationMgr** ()
- **bool AddDeprecatedFunction** (const char \*OldFunctionName, const char \*NewFunctionName, const void \*FramePtr)

### Static Public Member Functions

- **static DeprecationMgr & GetInstance** ()

#### 6.45.1 Detailed Description

Used to deprecate a method. To deprecate a method use:

```
bool NewMethod(float);
DEPRECATE_FUNC void OldMethod(int)
{
    DEPRECATE("void OldMethod(int)"
              "bool NewMethod(float)");

    NewMethod(float(int));
}
```

#### 6.45.2 Constructor & Destructor Documentation

##### 6.45.2.1 ~DeprecationMgr ()

#### 6.45.3 Member Function Documentation

##### 6.45.3.1 **bool AddDeprecatedFunction** (const char \* *OldFunctionName*, const char \* *NewFunctionName*, const void \* *FramePtr*)

##### 6.45.3.2 **DeprecationMgr & GetInstance** () [static]

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

- **deprecationmgr.h**
- **deprecationmgr.cpp**

## 6.46 DeprecationMgrImpl Class Reference

### Public Attributes

- `std::map< const char *, DeprecatedFunction > mFunctions`

### 6.46.1 Member Data Documentation

#### 6.46.1.1 `std::map<const char*, DeprecatedFunction> mFunctions`

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

- `deprecationmgr.cpp`

## 6.47 DoNothing< Ret, T > Struct Template Reference

This odd functor is actually useful for supplying a default template parameter when a functors is expected.

```
#include <inc/dtUtil/templateutility.h>
```

### Public Member Functions

- Ret operator() (T)

#### 6.47.1 Detailed Description

```
template<typename Ret, typename T> struct dtUtil::DoNothing< Ret, T >
```

This odd functor is actually useful for supplying a default template parameter when a functors is expected.

#### 6.47.2 Member Function Documentation

##### 6.47.2.1 Ret operator() (T) [inline]

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

- `templateutility.h`

## 6.48 DoNothing0< Ret > Struct Template Reference

This odd functor is actually useful for supplying a default template parameter when a functors is expected.

```
#include <inc/dtUtil/templateutility.h>
```

### Public Member Functions

- Ret operator() ()

#### 6.48.1 Detailed Description

```
template<typename Ret = void> struct dtUtil::DoNothing0< Ret >
```

This odd functor is actually useful for supplying a default template parameter when a functors is expected.

#### 6.48.2 Member Function Documentation

##### 6.48.2.1 Ret operator() () [inline]

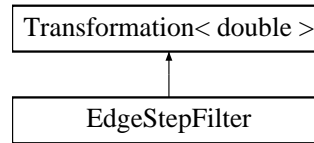
The documentation for this struct was generated from the following file:

- `templateutility.h`

## 6.49 EdgeStepFilter Class Reference

The **EdgeStepFilter** (p. 108) will return back 1.0, 0.0, or -1.0 depending if the supplied input value is greater than the high, in between high and low, or less than the low.

#include <inc/dtUtil/transformation.h> Inheritance diagram for EdgeStepFilter::



### Public Member Functions

- **EdgeStepFilter** (double low, double high)
- double **operator()** (double input) const

#### 6.49.1 Detailed Description

The **EdgeStepFilter** (p. 108) will return back 1.0, 0.0, or -1.0 depending if the supplied input value is greater than the high, in between high and low, or less than the low.

#### 6.49.2 Constructor & Destructor Documentation

##### 6.49.2.1 EdgeStepFilter (double *low*, double *high*) [inline]

#### 6.49.3 Member Function Documentation

##### 6.49.3.1 double operator() (double *input*) const [inline, virtual]

Implements **Transformation< double >** (p. 282).

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

- **transformation.h**

## 6.50 EmptyType Struct Reference

```
#include <inc/dtUtil/generic.h>
```

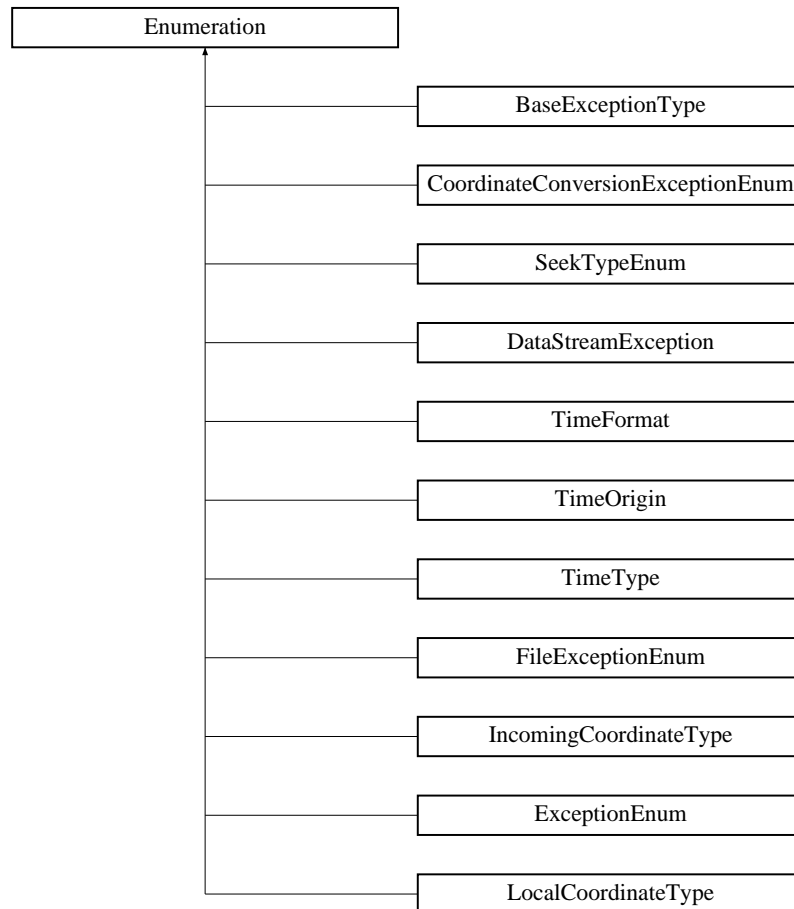
The documentation for this struct was generated from the following file:

- **generic.h**

## 6.51 Enumeration Class Reference

This class represents a type-safe enumeration pattern.

#include <inc/dtUtil/enumeration.h> Inheritance diagram for Enumeration::



### Public Member Functions

- const std::string & **GetName** () const  
*Inlined because it's called frequently.*
- bool **operator!=** (const std::string &rhs) const  
*Overloaded inequality test for this enumeration's string value.*
- bool **operator!=** (const **Enumeration** &rhs) const  
*Inequality test for an enumeration.*
- bool **operator<** (const **Enumeration** &rhs) const  
*Overloaded less than operator.*
- bool **operator<** (const std::string &rhs) const  
*Overloaded less than test for this enumeration's string value.*
- bool **operator==** (const std::string &rhs) const  
*Overloaded string compare operator for the enumeration.*
- bool **operator==** (const **Enumeration** &rhs) const  
*Equality test for an enumeration.*

- **bool operator>** (const std::string &rhs) const  
*Overloaded greater than test for this enumeration's string value.*

## Protected Member Functions

- **Enumeration** (const std::string &name)  
*Construct the enumeration.*
- **virtual ~Enumeration** ()  
*Private virtual desctructor to get rid of compile warning.*

### 6.51.1 Detailed Description

This class represents a type-safe enumeration pattern. It allows one to also enumerate an enumeration thus looking up values in a list fashion.

### 6.51.2 Constructor & Destructor Documentation

#### 6.51.2.1 ~Enumeration () [protected, virtual]

Private virtual desctructor to get rid of compile warning.

#### 6.51.2.2 Enumeration (const std::string & name) [protected]

Construct the enumeration. Note When creating a new enumeration, the constructor of derived types must call addInstance(this) in order for enumerations to be enumerated.

### 6.51.3 Member Function Documentation

#### 6.51.3.1 const std::string& GetName () const [inline]

Inlined because it's called frequently. Returns the string representation of this enumeration.

#### 6.51.3.2 bool operator!= (const std::string & rhs) const

Overloaded inequality test for this enumeration's string value.

#### 6.51.3.3 bool operator!= (const Enumeration & rhs) const [inline]

Inequality test for an enumeration. Inlined because it's called frequently See also operator==

#### 6.51.3.4 bool operator< (const Enumeration & rhs) const [inline]

Overloaded less than operator. This checks the memory addresses of the two enumerations. Inlined because it's called frequently Parameters

**rhs** The second enumeration to compare to this one.

Returns True if this enumeration is less than rhs. Note This method is supported by enumerations so they may be used as keys in STL containers. Since memory addresses are guaranteed to be unique this methods works fine. However, it really does not make sense to use this method of comparison in other circumstances.

#### 6.51.3.5 bool operator< (const std::string & rhs) const

Overloaded less than test for this enumeration's string value.

#### 6.51.3.6 bool operator== (const std::string & rhs) const

Overloaded string compare operator for the enumeration. This will compare the string to this enumerations getName() value. Parameters

**rhs**

Returns True if they are equal. Note Uses the STL string compare method implying that the rules for string equality are the same as they are for the STL string compare method.

**6.51.3.7 bool operator==(const Enumeration & rhs) const [inline]**

Equality test for an enumeration. Since enumeration objects are static, and only references to enumerations may be stored by the user, it is safe and efficient to compare enumeration objects based on their memory address. Inlined because it's called frequently

**6.51.3.8 bool operator>(const std::string & rhs) const**

Overloaded greater than test for this enumeration's string value.

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

- **enumeration.h**
- **enumeration.cpp**

## 6.52 EvaluateFuncotr< FunctorType, ArgType, RetType > Struct Template Reference

```
#include <inc/dtUtil/templateutility.h>
```

### Public Member Functions

- RetType **operator()** (FunctorType func, ArgType arg)

```
template<typename FunctorType, typename ArgType, typename RetType = void> struct  
dtUtil::EvaluateFuncotr< FunctorType, ArgType, RetType >
```

### 6.52.1 Member Function Documentation

#### 6.52.1.1 RetType operator() (FunctorType *func*, ArgType *arg*) [inline]

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

- `templateutility.h`

## 6.53 EvaluateFunctor< FunctorType \*, ArgType, RetType > Struct Template Reference

```
#include <inc/dtUtil/templateutility.h>
```

### Public Member Functions

- RetType **operator()** (FunctorType \*func, ArgType arg)

```
template<typename FunctorType, typename ArgType, typename RetType> struct  
dtUtil::EvaluateFunctor< FunctorType *, ArgType, RetType >
```

### 6.53.1 Member Function Documentation

#### 6.53.1.1 RetType operator() (FunctorType \* *func*, ArgType *arg*) [inline]

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

- `templateutility.h`

## 6.54 EvaluateInvoke< CompareHandler, InvokeHandler, EvaluateResult > Struct Template Reference

```
#include <inc/dtUtil/templateutility.h>
```

### Public Member Functions

- void **operator()** (std::pair< CompareHandler, InvokeHandler > &dataType)

```
template<typename CompareHandler, typename InvokeHandler, typename EvaluateResult> struct dtUtil::EvaluateInvoke< CompareHandler, InvokeHandler, EvaluateResult >
```

### 6.54.1 Member Function Documentation

#### 6.54.1.1 void operator() (std::pair< CompareHandler, InvokeHandler > & *dataType*) [inline]

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

- templateutility.h

## 6.55 Exception Class Reference

```
#include <inc/dtUtil/exception.h>
```

### Public Member Functions

- **Exception** (const std::string &message, const std::string &filename, unsigned int linenum)  
*Constructor - Initializes the exception and logs it.*
- **Exception** (**Enumeration** &type, const std::string &message, const std::string &filename, unsigned int linenum)  
*Constructor - Initializes the exception and logs it.*
- virtual ~**Exception** ()
- const std::string & **File** () const
- unsigned int **Line** () const
- void **LogException** (dtUtil::Log::LogMessageType level, dtUtil::Log &logger) const  
*logs the exception to the following log level using the given logger.*
- void **LogException** (dtUtil::Log::LogMessageType level, const std::string &loggerName) const  
*logs the exception to the following log level using the logger.*
- void **LogException** (dtUtil::Log::LogMessageType level=dtUtil::Log::LOG\_ERROR) const  
*logs the exception to the default logger.*
- void **Print** () const  
*Prints the exception to the console.*
- std::string **ToString** () const  
*Converts this exception to a string.*
- const **Enumeration** & **TypeEnum** () const
- const std::string & **What** () const

### Protected Attributes

- std::string **mFileName**
- unsigned int **mLineNum**
- std::string **mMessage**
- **Enumeration** & **mType**

#### 6.55.1 Constructor & Destructor Documentation

##### 6.55.1.1 Exception (**Enumeration** & *type*, const std::string & *message*, const std::string & *filename*, unsigned int *linenum*)

Constructor - Initializes the exception and logs it. Parameters

***type*** - the type of exception being thrown.

***message*** - Message to display about the exception.

***filename*** - File the exception was thrown from.

***linenum*** - Line number in the file from which the exception was thrown.

**6.55.1.2 Exception (const std::string & message, const std::string & filename, unsigned int linenum)**

Constructor - Initializes the exception and logs it. Parameters

**message** - Message to display about the exception.

**filename** - File the exception was thrown from.

**linenum** - Line number in the file from which the exception was thrown.

**6.55.1.3 virtual ~Exception () [inline, virtual]****6.55.2 Member Function Documentation****6.55.2.1 const std::string& File () const [inline]****6.55.2.2 unsigned int Line () const [inline]**

Returns the line number associated with this exception.

**6.55.2.3 void LogException (dtUtil::Log::LogMessageType level, dtUtil::Log & logger) const**

logs the exception to the following log level using the given logger. Parameters

**level** The level/type of logging

**logger** the actual log instance used to log.

**6.55.2.4 void LogException (dtUtil::Log::LogMessageType level, const std::string & loggerName) const**

logs the exception to the following log level using the logger. Parameters

**level** The level/type of logging

**loggerName** the name passed to "getInstance" of the Logger.

**6.55.2.5 void LogException (dtUtil::Log::LogMessageType level = dtUtil::Log::LOG\_ERROR) const**

logs the exception to the default logger. Parameters

**level** The level/type of logging

**6.55.2.6 void Print () const**

Prints the exception to the console.

**6.55.2.7 std::string ToString () const**

Converts this exception to a string. The string contains the reason, line number and file the exception was thrown from. Returns The string version of this exception.

**6.55.2.8 const Enumeration& TypeEnum () const [inline]**

Returns the enumerated type of the exception.

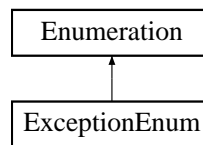
**6.55.2.9 const std::string& What () const [inline]****6.55.3 Member Data Documentation****6.55.3.1 std::string mFileName [protected]****6.55.3.2 unsigned int mLineNum [protected]****6.55.3.3 std::string mMessage [protected]****6.55.3.4 Enumeration& mType [protected]**

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

- **exception.h**
- **exception.cpp**

## 6.56 ExceptionEnum Class Reference

#include <inc/dtUtil/librarysharingmanager.h>Inheritance diagram for ExceptionEnum::



### Static Public Attributes

- static **ExceptionEnum** **LibraryLoadingError**

### Protected Member Functions

- **ExceptionEnum** (const std::string &name)

### 6.56.1 Constructor & Destructor Documentation

6.56.1.1 **ExceptionEnum** (const std::string & *name*) [*inline*, *protected*]

### 6.56.2 Member Data Documentation

6.56.2.1 **LibrarySharingManager::ExceptionEnum** **LibraryLoadingError** [*static*]

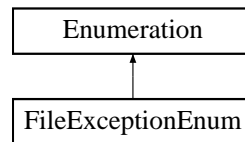
The documentation for this class was generated from the following files:

- **librarysharingmanager.h**
- **librarysharingmanager.cpp**

## 6.57 FileExceptionEnum Class Reference

An enumeration of exception types that will be thrown by fileutils.

#include <inc/dtUtil/fileutils.h>Inheritance diagram for FileExceptionEnum::



### Static Public Attributes

- static **FileExceptionEnum** FileNotFound
- static **FileExceptionEnum** IOException

### Protected Member Functions

- **FileExceptionEnum** (const std::string &name)

#### 6.57.1 Detailed Description

An enumeration of exception types that will be thrown by fileutils.

#### 6.57.2 Constructor & Destructor Documentation

6.57.2.1 **FileExceptionEnum** (const std::string & *name*) [*inline*, *protected*]

#### 6.57.3 Member Data Documentation

6.57.3.1 **FileExceptionEnum** FileNotFound [*static*]

6.57.3.2 **FileExceptionEnum** IOException [*static*]

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

- fileutils.h
- fileutils.cpp

## 6.58 FileInfo Struct Reference

```
#include <inc/dtUtil/fileutils.h>
```

### Public Member Functions

- **FileInfo** ()

### Public Attributes

- std::string **baseName**
- std::string **fileName**
- **FileType** **fileType**
- time\_t **lastModified**
- std::string **path**
- size\_t **size**

### 6.58.1 Constructor & Destructor Documentation

6.58.1.1 **FileInfo** () [inline]

### 6.58.2 Member Data Documentation

6.58.2.1 **std::string baseName**

6.58.2.2 **std::string fileName**

6.58.2.3 **FileType fileType**

6.58.2.4 **time\_t lastModified**

6.58.2.5 **std::string path**

6.58.2.6 **size\_t size**

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

- **fileutils.h**

## 6.59 FileUtils Class Reference

```
#include <inc/dtUtil/fileutils.h>
```

### Public Member Functions

- void **ChangeDirectory** (const std::string &path)
 

*Changes the current directory to the one given in "path." This will clear the stack of directories that is set by pushDirectory and popDirectory.*
- const std::string & **CurrentDirectory** () const
- void **DirCopy** (const std::string &srcPath, const std::string &destPath, bool bOverwrite, bool copyContentsOnly=false) const
 

*Copys an entire directory.*
- bool **DirDelete** (const std::string &strDir, bool bRecursive)
 

*Deletes a directory.*
- bool **DirExists** (const std::string &strDir) const
- **DirectoryContents DirGetFiles** (const std::string &path, const **FileExtensionList** &extensions=**FileExtensionList**()) const
 

*Note: throws exceptions of type dtUtil::Exception (p. 116).*
- **DirectoryContents DirGetSubs** (const std::string &path) const
- void **FileCopy** (const std::string &strSrc, const std::string &strDest, bool bOverwrite) const
 

*Copys a file.*
- void **FileDelete** (const std::string &strFile) const
 

*Deletes the given file.*
- bool **FileExists** (const std::string &strFile) const
- void **FileMove** (const std::string &strSrc, const std::string &strDest, bool bOverwrite) const
 

*Moves a file.*
- const std::string **GetAbsolutePath** (const std::string &relativePath) const
 

*Converts a relative path to an absolute path.*
- struct **FileInfo GetFileInfo** (const std::string &strFile) const
- bool **IsSameFile** (const std::string &file1, const std::string &file2) const
 

*Converts an absolute path to a relative path based on the current working directory.*
- void **MakeDirectory** (const std::string &strDir) const
 

*creates a new directory from a path.*
- void **PopDirectory** ()
 

*sets the current directory to the last directory on the stack.*
- void **PushDirectory** (const std::string &path)
 

*Changes the current directory to the one given in "path" and adds the previous current directory to an internal stack so it can be returned to via popDirectory.*
- std::string **RelativePath** (const std::string &absolutePath, const std::string &file) const
 

*Helper function that returns the relative path between absolutePath and file.*

## Static Public Member Functions

- static **FileUtils & GetInstance ()**  
*Character separating the parts of a file path.*

## Static Public Attributes

- static const char **PATH\_SEPARATOR = '/'**

### 6.59.1 Member Function Documentation

#### 6.59.1.1 void ChangeDirectory (const std::string & path)

Changes the current directory to the one given in "path." This will clear the stack of directories that is set by pushDirectory and popDirectory. If this call fails, the stack will not be cleared. See also pushDirectory

popDirectory

Parameters

**path** The path to the new directory.

Exceptions

**FileExceptionEnum::FileNotFound** (p. 119) if the path does not exist.

#### 6.59.1.2 const std::string & CurrentDirectory () const

Returns the full path to the current directory.

#### 6.59.1.3 void DirCopy (const std::string & srcPath, const std::string & destPath, bool bOverwrite, bool copyContentsOnly = false) const

Copys an entire directory. If destPath exists, then a subdirectory will be created in destPath with the same name as srcPath unless copyContentsOnly is true, in which case the contents of srcPath will be copied into destPath. If destPath does not exist, destPath will be created if the parent exists and the contents of srcPath will be copied to destPath whether copyContentsOnly is true or false.

Parameters

**srcPath** the source directory to copy.

**destPath** the destination directory.

**bOverwrite** true if this call should overwrite the destination file if it exists.

**copyContentsOnly** true if the contents of srcPath should be copied into destPath rather than create a subdirectory

Exceptions

**FileExceptionEnum::FileNotFound** (p. 119) if the source file is not found.

**FileExceptionEnum::IOException** (p. 119) if an error occurs copying the data or bOverwrite was false and a destination file exists.

#### 6.59.1.4 bool DirDelete (const std::string & strDir, bool bRecursive)

Deletes a directory. If bRecursive is true, the directory and all it's contents will be removed. If it's false, the call will fail unless the directory is empty. Parameters

**strDir** The path of the directory to delete.

**bRecursive** true if the directory should be deleted recursively.

Returns true if successful or false if the directory is NOT empty and bRecursive is false.

Exceptions

**FileNotFoundException::FileNotFoundException** (p. 119) if the path does not exist.

**FileNotFoundException::IOException** (p. 119) if an error occurs deleting the directory

#### 6.59.1.5 bool DirExists (const std::string & strDir) const

Parameters

**strDir** The directory to check.

Returns true if the path exists and is a directory

#### 6.59.1.6 DirectoryContents DirGetFiles (const std::string & path, const FileExtensionList & extensions = FileExtensionList()) const

Note: throws exceptions of type dtUtil::Exception (p. 116). Parameters

**path** the path to the directory to list the contents of.

**extensions** Optional list of file extensions to filter on, including the "dot". (e.g., ".txt", ".xml")

Returns a vector of file names.

Exceptions

**FileNotFoundException::FileNotFoundException** (p. 119) if the path does not exist.

**FileNotFoundException::IOException** (p. 119) if the path is not an actual directory

#### 6.59.1.7 DirectoryContents DirGetSubs (const std::string & path) const

Parameters

**path** the path to the directory to get the subdirectories for.

Returns a vector holding the list of subdirectories.

Exceptions

**FileNotFoundException::FileNotFoundException** (p. 119) if the path does not exist.

#### 6.59.1.8 void FileCopy (const std::string & strSrc, const std::string & strDest, bool bOverwrite) const

Copys a file. Parameters

**strSrc** The path to the source file.

**strDest** The path to the destination file or directory.

**bOverwrite** true if this call should overwrite the destination file if it exists.

Exceptions

**FileNotFoundException::FileNotFoundException** (p. 119) if the source file is not found.

**FileNotFoundException::IOException** (p. 119) if an error occurs copying the data or bOverwrite was false and the destination file exists.

### 6.59.1.9 void FileDelete (const std::string & *strFile*) const

Deletes the given file. Parameters

***strFile*** the path to the file to delete

Exceptions

***FileExceptionEnum::IOException*** (p. 119) if an error occurs deleting the data

### 6.59.1.10 bool FileExists (const std::string & *strFile*) const

Parameters

***strFile*** the path to the file to check.

Returns true if the file exists.

### 6.59.1.11 void FileMove (const std::string & *strSrc*, const std::string & *strDest*, bool *bOverwrite*) const

Moves a file. This call will attempt to move the file without moving the data, but if it can't, the file will first be copied then the source file will be removed. Parameters

***strSrc*** The path to the source file

***strDest*** The path to the destination file or directory

***bOverwrite*** true if this call should overwrite the destination file if it exists.

Exceptions

***FileExceptionEnum::FileNotFound*** (p. 119) if the source file is not found.

***FileExceptionEnum::IOException*** (p. 119) if an error occurs moving the data or *bOverwrite* was false and the destination file exists.

### 6.59.1.12 const std::string GetAbsolutePath (const std::string & *relativePath*) const

Converts a relative path to an absolute path. Parameters

***relativePath*** the relative path to convert to absolute.

Returns the absolute path.

Exceptions

***FileExceptionEnum::FileNotFound*** (p. 119) if the path does not exist.

### 6.59.1.13 struct FileInfo GetFileInfo (const std::string & *strFile*) const [read]

Note If the file is not found, the *fileType* value will be set to `FILE_NOT_FOUND` and all other values will be undefined. Returns the `FileInfo` struct for the given file. See also `dtUtil::FileInfo` (p. 120)

### 6.59.1.14 static FileUtils& GetInstance () [inline, static]

Character separating the parts of a file path. Returns the single instance of this class..

**6.59.1.15 bool IsSameFile (const std::string & file1, const std::string & file2) const**

Converts an absolute path to a relative path based on the current working directory. Parameters

**absPath** the absolute path to process

**relPath** output parameter that will contain the relative path. It is possible for two different path strings to point at the same file on disk. (Things like relative paths and filesystem links make this possible).

This function makes absolutely certain that the two files aren't the same by checking the inodes of the two files -- preventing things like having the **FileUtils::FileCopy** (p. 123) accidentally blow away a file by copying it onto itself.

Parameters

**file1** -- Path to first file.

**file2** -- Path to second file.

Returns True if inodes match, false otherwise (if one or both files are inaccessible, returns false).

**6.59.1.16 void MakeDirectory (const std::string & strDir) const**

creates a new directory from a path. Parameters

**strDir** the directory to create.

Exceptions

**FileExceptionEnum::FileNotFound** (p. 119) if the parent path does not exist.

**FileExceptionEnum::IOException** (p. 119) if an error occurs creating the directory

**6.59.1.17 void PopDirectory ()**

sets the current directory to the last directory on the stack. See also pushDirectory

Exceptions

**FileExceptionEnum::FileNotFound** (p. 119) if the previous directory no longer exists.

**6.59.1.18 void PushDirectory (const std::string & path)**

Changes the current directory to the one given in "path" and adds the previous current directory to an internal stack so it can be returned to via popDirectory. If this call fails, the stack will not be changed. See also popDirectory

Parameters

**path** The path to the new directory.

Exceptions

**FileExceptionEnum::FileNotFound** (p. 119) if the path does not exist.

**6.59.1.19 std::string RelativePath (const std::string & absolutePath, const std::string & file) const**

Helper function that returns the relative path between absolutePath and file. Parameters

**absolutePath** The absolute path to search

**file** The absolute path to the file

Returns The relative path or empty string for failure Note This function assumes that directory separators are equal for both paths

## 6.59.2 Member Data Documentation

### 6.59.2.1 `const char PATH_SEPARATOR = '/'` [static]

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

- `fileutils.h`
- `fileutils.cpp`

## 6.60 Filter2< TL, IdsTL, include > Struct Template Reference

```
#include <inc/dtUtil/funbind.h>
```

### Classes

- struct **Temp**
- struct **Temp**< dtUtil::NullType, i, dtUtil::NullType, Predicate >
- struct **Temp**< dtUtil::NullType, i, IdsTL2, Predicate >
- struct **Temp**< TL2, 0, dtUtil::NullType, dtUtil::IsIntType >
- struct **Temp**< TL2, 0, dtUtil::NullType, dtUtil::NotIntType >
- struct **Temp**< TL2, 0, typename dtUtil::IdsFromTL< TL2 >::Type, dtUtil::IsIntType >
- struct **Temp**< TL2, 0, typename dtUtil::IdsFromTL< TL2 >::Type, dtUtil::NotIntType >
- struct **Temp**< TL2, i, dtUtil::NullType, Predicate >

### Public Types

- typedef **dtUtil::Select**< include, typename **Temp**< TL, 0, IdsTL, dtUtil::IsIntType >::Result, typename **Temp**< TL, 0, IdsTL, dtUtil::NotIntType >::Result >::Result Result

```
template<class TL, class IdsTL, bool include> struct dtUtil::Filter2< TL, IdsTL, include >
```

#### 6.60.1 Member Typedef Documentation

**6.60.1.1** typedef dtUtil::Select< include, typename Temp<TL, 0, IdsTL, dtUtil::IsIntType>::Result, typename Temp<TL, 0, IdsTL, dtUtil::NotIntType>::Result >::Result Result

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

- **funbind.h**

## 6.61 Fractal< Real, Vector, Noise > Class Template Reference

**Fractal** (p. 128): This class is made to complement the noise classes where as the noise class hashes vectors to floats between -1 to 1 this class can be used to make summations of the noise to use this class I recommend including **NoiseUtility.h** (p. 372) and using the appropriate typedefs.

```
#include <inc/dtUtil/fractal.h>
```

### Public Member Functions

- Real **FBM** (Vector vect\_in, int octaves=2, Real freq=1.0f, Real persistence=0.5f, Real lacunarity=2.0f)  
*FBM: The standard **Fractal** (p. 128) Brownian Motion summation.*
- Real **HeteroFractal** (Vector vect\_in, int octaves=2, Real freq=1.0f, Real persistence=0.5f, Real lacunarity=2.0f, Real offset=0.5f)
- Real **IslandFractal** (Vector vect\_in, int octaves=4, Real freq=1.0f, Real persistence=0.5f, Real lacunarity=2.0f, Real oscarity=2.37f)  
*This one I made myself which basically just scales the output exponentially to make an island look.*
- Real **Marble** (Vector vect\_in, int octaves=1, Real freq=1.0f, Real persistence=0.5f, Real lacunarity=2.0f)  
*A basic function that makes a pattern of lines in the x direction that could be interpreted as a marble look try to make the output of this swirl, for added realism.*
- Real **RigidMultiFractal** (Vector vect\_in, int octaves=2, Real freq=1.0f, Real persistence=0.5f, Real lacunarity=2.0f, Real offset=0.5f, Real gain=2.0f)
- Real **Turbulence** (Vector vect\_in, int octaves=1, Real freq=1.0f, Real persistence=0.5f, Real lacunarity=2.0f)  
*Turbulence: this should be used for gases, clouds, lava, etc, generates an inconsistent bubbly pattern.*

### 6.61.1 Detailed Description

**template<class Real, class Vector, class Noise> class dtUtil::Fractal< Real, Vector, Noise >**

**Fractal** (p. 128): This class is made to complement the noise classes where as the noise class hashes vectors to floats between -1 to 1 this class can be used to make summations of the noise to use this class I recommend including **NoiseUtility.h** (p. 372) and using the appropriate typedefs. Definitions: Octaves: The number of successive steps to accumulate noise Frequency: The fractal dimension, or the number of noise interpolations per Real resolution Persistence: This defines the how the octaves will be weighted, where the weight at each octave is the persistence <sup>^</sup> octave Lacunarity: The gap between successive steps, or the rate at which the frequency is changing Offset: The amount to raise the terrain from sea level Gain: The scale value for the current noise to multiply into the next octave, used to produce smooth valleys and jagged mountain tops Oscarity: This convoluted word I made up which appears in IslandFractal, it is used to scale the squared value of the noise function higher values make larger differences between peaks and sea level

See also <http://www.texturingandmodeling.com/> Note: This book is awesome and is where most of this material came from

### 6.61.2 Member Function Documentation

**6.61.2.1 Real FBM (Vector vect\_in, int octaves = 2, Real freq = 1.0f, Real persistence = 0.5f, Real lacunarity = 2.0f) [inline]**

FBM: The standard **Fractal** (p. 128) Brownian Motion summation.

**6.61.2.2 Real HeteroFractal (Vector vect\_in, int octaves = 2, Real freq = 1.0f, Real persistence = 0.5f, Real lacunarity = 2.0f, Real offset = 0.5f) [inline]**

**6.61.2.3 Real IslandFractal (Vector vect\_in, int octaves = 4, Real freq = 1.0f, Real persistence = 0.5f, Real lacunarity = 2.0f, Real oscarity = 2.37f) [inline]**

This one I made myself which basically just scales the output exponentially to make an island look.

**6.61.2.4 Real Marble (Vector *vect\_in*, int *octaves* = 1, Real *freq* = 1.0f, Real *persistance* = 0.5f, Real *lacunarity* = 2.0f) [inline]**

A basic function that makes a pattern of lines in the x direction that could be interpreted as a marble look try to make the output of this swirl, for added realism.

**6.61.2.5 Real RigidMultiFractal (Vector *vect\_in*, int *octaves* = 2, Real *freq* = 1.0f, Real *persistance* = 0.5f, Real *lacunarity* = 2.0f, Real *offset* = 0.5f, Real *gain* = 2.0f) [inline]****6.61.2.6 Real Turbulence (Vector *vect\_in*, int *octaves* = 1, Real *freq* = 1.0f, Real *persistance* = 0.5f, Real *lacunarity* = 2.0f) [inline]**

Turbulence: this should be used for gases, clouds, lava, etc, generates an inconsistent bubbly pattern.

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

- **fractal.h**

## 6.62 Functor< R, TList, size > Class Template Reference

```
#include <inc/dtUtil/functor.h>
```

### Classes

- struct **ByValue**
- struct **FunctorImpl**
- struct **FunImplBase**
- struct **FunStorageImpl**
- struct **MemberFnImpl**
- struct **NewAlloc**
- struct **SelectStored**
- struct **Stored**
- struct **Typeless**

### Public Types

- typedef **dtUtil::TypeAtNonStrict**< TList, 0, **dtUtil::NullType** >::Result **Parm1**
- typedef **dtUtil::TypeAtNonStrict**< TList, 1, **dtUtil::NullType** >::Result **Parm2**
- typedef **dtUtil::TypeAtNonStrict**< TList, 2, **dtUtil::NullType** >::Result **Parm3**
- typedef **dtUtil::TypeAtNonStrict**< TList, 3, **dtUtil::NullType** >::Result **Parm4**
- typedef **dtUtil::TypeAtNonStrict**< TList, 4, **dtUtil::NullType** >::Result **Parm5**
- typedef **dtUtil::TypeAtNonStrict**< TList, 5, **dtUtil::NullType** >::Result **Parm6**
- typedef **dtUtil::TypeAtNonStrict**< TList, 6, **dtUtil::NullType** >::Result **Parm7**
- typedef CallParms< TList >::**ParmsListType** **ParmsListType**
- typedef R **ResultType**
- typedef TList **TypeListType**

### Public Member Functions

- template<class P, typename MF >  
**Functor** (P const &pobj, MF memfun)
- template<typename F >  
**Functor** (F const &fun)
- **Functor** (**Functor** const &src)
- **Functor** ()
- **~Functor** ()
- bool **operator!** () const
- R **operator()** (**Parm1** p1, **Parm2** p2, **Parm3** p3, **Parm4** p4, **Parm5** p5, **Parm6** p6, **Parm7** p7) const
- R **operator()** (**Parm1** p1, **Parm2** p2, **Parm3** p3, **Parm4** p4, **Parm5** p5, **Parm6** p6) const
- R **operator()** (**Parm1** p1, **Parm2** p2, **Parm3** p3, **Parm4** p4, **Parm5** p5) const
- R **operator()** (**Parm1** p1, **Parm2** p2, **Parm3** p3, **Parm4** p4) const
- R **operator()** (**Parm1** p1, **Parm2** p2, **Parm3** p3) const
- R **operator()** (**Parm1** p1, **Parm2** p2) const
- R **operator()** (**Parm1** p1) const
- R **operator()** () const
- R **operator()** (**ParmsListType** const &parms) const
- **Functor** & **operator=** (**Functor** const &src)
- bool **valid** () const

```
template<typename R, class TList, unsigned int size = 4 * sizeof(void*)> class dtUtil::Functor< R, TList, size >
```

### 6.62.1 Member Typedef Documentation

- 6.62.1.1 `typedef dtUtil::TypeAtNonStrict<TList, 0, dtUtil::NullType>::Result Parm1`
- 6.62.1.2 `typedef dtUtil::TypeAtNonStrict<TList, 1, dtUtil::NullType>::Result Parm2`
- 6.62.1.3 `typedef dtUtil::TypeAtNonStrict<TList, 2, dtUtil::NullType>::Result Parm3`
- 6.62.1.4 `typedef dtUtil::TypeAtNonStrict<TList, 3, dtUtil::NullType>::Result Parm4`
- 6.62.1.5 `typedef dtUtil::TypeAtNonStrict<TList, 4, dtUtil::NullType>::Result Parm5`
- 6.62.1.6 `typedef dtUtil::TypeAtNonStrict<TList, 5, dtUtil::NullType>::Result Parm6`
- 6.62.1.7 `typedef dtUtil::TypeAtNonStrict<TList, 6, dtUtil::NullType>::Result Parm7`
- 6.62.1.8 `typedef CallParams<TList>::ParamsListType ParamsListType`
- 6.62.1.9 `typedef R ResultType`
- 6.62.1.10 `typedef TList TypeListType`

### 6.62.2 Constructor & Destructor Documentation

- 6.62.2.1 `Functor () [inline]`
- 6.62.2.2 `~Functor () [inline]`
- 6.62.2.3 `Functor (Functor< R, TList, size > const & src) [inline]`
- 6.62.2.4 `Functor (F const & fun) [inline, explicit]`
- 6.62.2.5 `Functor (P const & pobj, MF memfun) [inline, explicit]`

### 6.62.3 Member Function Documentation

- 6.62.3.1 `bool operator! () const [inline]`
- 6.62.3.2 `R operator() (Parm1 p1, Parm2 p2, Parm3 p3, Parm4 p4, Parm5 p5, Parm6 p6, Parm7 p7) const [inline]`
- 6.62.3.3 `R operator() (Parm1 p1, Parm2 p2, Parm3 p3, Parm4 p4, Parm5 p5, Parm6 p6) const [inline]`
- 6.62.3.4 `R operator() (Parm1 p1, Parm2 p2, Parm3 p3, Parm4 p4, Parm5 p5) const [inline]`
- 6.62.3.5 `R operator() (Parm1 p1, Parm2 p2, Parm3 p3, Parm4 p4) const [inline]`
- 6.62.3.6 `R operator() (Parm1 p1, Parm2 p2, Parm3 p3) const [inline]`
- 6.62.3.7 `R operator() (Parm1 p1, Parm2 p2) const [inline]`
- 6.62.3.8 `R operator() (Parm1 p1) const [inline]`
- 6.62.3.9 `R operator() () const [inline]`
- 6.62.3.10 `R operator() (ParamsListType const & parms) const [inline]`
- 6.62.3.11 `Functor& operator= (Functor< R, TList, size > const & src) [inline]`
- 6.62.3.12 `bool valid () const [inline]`

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

- `functor.h`

## 6.63 FunctorCall< CallType, R, TYPELIST\_0()> Struct Template Reference

```
#include <inc/dtUtil/funcall.h>
```

### Public Types

- typedef dtUtil::InstantiateH< dtUtil::NullType, dtUtil::TupleHolder > **ParmsListType**

### Static Public Member Functions

- template<class PObj >  
static R **Call** (PObj const &pobj, CallType memfun, **ParmsListType** &)
- template<class Fun >  
static R **Call** (Fun const &fun, **ParmsListType** &)

```
template<typename CallType, typename R> struct dtUtil::FunctorCall< CallType, R, TYPELIST_0()>
```

#### 6.63.1 Member Typedef Documentation

6.63.1.1 typedef dtUtil::InstantiateH<dtUtil::NullType, dtUtil::TupleHolder> **ParmsListType**

#### 6.63.2 Member Function Documentation

6.63.2.1 static R **Call** (PObj const & *pobj*, CallType *memfun*, **ParmsListType** &) [*inline, static*]

6.63.2.2 static R **Call** (Fun const & *fun*, **ParmsListType** &) [*inline, static*]

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

- **funcall.h**

## 6.64 FunctorCall< CallType, R, TYPELIST\_1(P1)> Struct Template Reference

```
#include <inc/dtUtil/funcall.h>
```

### Public Types

- typedef dtUtil::InstantiateH< typename **dtUtil::CreateTL**< P1 >::Type, dtUtil::TupleHolder > **ParmsListType**

### Static Public Member Functions

- template<class PObj >  
static R **Call** (PObj const &pobj, CallType memfun, **ParmsListType** &parms)
- template<class Fun >  
static R **Call** (Fun const &fun, **ParmsListType** &parms)

```
template<typename CallType, typename R, typename P1> struct dtUtil::FunctorCall< CallType, R, TYPELIST_1(P1)>
```

#### 6.64.1 Member Typedef Documentation

6.64.1.1 typedef dtUtil::InstantiateH<typename dtUtil::CreateTL<P1>::Type, dtUtil::TupleHolder> **ParmsListType**

#### 6.64.2 Member Function Documentation

6.64.2.1 static R **Call** (PObj const & *pobj*, CallType *memfun*, **ParmsListType** & *parms*) [*inline*, *static*]

6.64.2.2 static R **Call** (Fun const & *fun*, **ParmsListType** & *parms*) [*inline*, *static*]

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

- **funcall.h**

## 6.65 FunctorCall< CallType, R, TYPELIST\_2(P1, P2)> Struct Template Reference

```
#include <inc/dtUtil/funcall.h>
```

### Public Types

- typedef dtUtil::InstantiateH< typename dtUtil::CreateTL< P1, P2 >::Type, dtUtil::TupleHolder > **ParmsListType**

### Static Public Member Functions

- template<class PObj >  
static R **Call** (PObj const &pobj, CallType memfun, **ParmsListType** &parms)
- template<class Fun >  
static R **Call** (Fun const &fun, **ParmsListType** &parms)

```
template<typename CallType, typename R, typename P1, typename P2> struct dtUtil::FunctorCall< CallType, R, TYPELIST_2(P1, P2)>
```

#### 6.65.1 Member Typedef Documentation

6.65.1.1 typedef dtUtil::InstantiateH<typename dtUtil::CreateTL<P1, P2>::Type, dtUtil::TupleHolder> **ParmsListType**

#### 6.65.2 Member Function Documentation

6.65.2.1 static R **Call** (PObj const & *pobj*, CallType *memfun*, **ParmsListType** & *parms*) [*inline*, *static*]

6.65.2.2 static R **Call** (Fun const & *fun*, **ParmsListType** & *parms*) [*inline*, *static*]

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

- **funcall.h**

## 6.66 FunctorCall< CallType, R, TYPELIST\_3(P1, P2, P3)> Struct Template Reference

```
#include <inc/dtUtil/funcall.h>
```

### Public Types

- typedef dtUtil::InstantiateH< typename **dtUtil::CreateTL**< P1, P2, P3 >::Type, **dtUtil::TupleHolder** > **ParmsListType**

### Static Public Member Functions

- template<class PObj >  
static R **Call** (PObj const &pobj, CallType memfun, **ParmsListType** &parms)
- template<class Fun >  
static R **Call** (Fun const &fun, **ParmsListType** &parms)

```
template<typename CallType, typename R, typename P1, typename P2, typename P3> struct  
dtUtil::FunctorCall< CallType, R, TYPELIST_3(P1, P2, P3)>
```

#### 6.66.1 Member Typedef Documentation

6.66.1.1 typedef dtUtil::InstantiateH<typename dtUtil::CreateTL<P1, P2, P3>::Type,  
dtUtil::TupleHolder> **ParmsListType**

#### 6.66.2 Member Function Documentation

6.66.2.1 static R **Call** (PObj const & *pobj*, CallType *memfun*, **ParmsListType** & *parms*) [*inline*,  
*static*]

6.66.2.2 static R **Call** (Fun const & *fun*, **ParmsListType** & *parms*) [*inline*, *static*]

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

- **funcall.h**

## 6.67 FunctorCall< CallType, R, TYPELIST\_4(P1, P2, P3, P4)> Struct Template Reference

```
#include <inc/dtUtil/funcall.h>
```

### Public Types

- typedef dtUtil::InstantiateH< typename **dtUtil::CreateTL**< P1, P2, P3, P4 >::Type, **dtUtil::TupleHolder** > **ParmsListType**

### Static Public Member Functions

- template<class PObj >  
static R **Call** (PObj const &pobj, CallType memfun, **ParmsListType** &parms)
- template<class Fun >  
static R **Call** (Fun const &fun, **ParmsListType** &parms)

```
template<typename CallType, typename R, typename P1, typename P2, typename P3, typename P4>  
struct dtUtil::FunctorCall< CallType, R, TYPELIST_4(P1, P2, P3, P4)>
```

#### 6.67.1 Member Typedef Documentation

6.67.1.1 typedef dtUtil::InstantiateH<typename dtUtil::CreateTL<P1, P2, P3, P4>::Type, dtUtil::TupleHolder> **ParmsListType**

#### 6.67.2 Member Function Documentation

6.67.2.1 static R **Call** (PObj const & *pobj*, CallType *memfun*, **ParmsListType** & *parms*) [*inline*, *static*]

6.67.2.2 static R **Call** (Fun const & *fun*, **ParmsListType** & *parms*) [*inline*, *static*]

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

- **funcall.h**

## 6.68 FunctorCall< CallType, R, TYPELIST\_5(P1, P2, P3, P4, P5)> Struct Template Reference

```
#include <inc/dtUtil/funcall.h>
```

### Public Types

- typedef dtUtil::InstantiateH< typename **dtUtil::CreateTL**< P1, P2, P3, P4, P5 >::Type, **dtUtil::TupleHolder** > **ParmsListType**

### Static Public Member Functions

- template<class PObj >  
static R **Call** (PObj const &pobj, CallType memfun, **ParmsListType** &parms)
- template<class Fun >  
static R **Call** (Fun const &fun, **ParmsListType** &parms)

```
template<typename CallType, typename R, typename P1, typename P2, typename P3, typename P4, type-  
name P5> struct dtUtil::FunctorCall< CallType, R, TYPELIST_5(P1, P2, P3, P4, P5)>
```

#### 6.68.1 Member Typedef Documentation

6.68.1.1 typedef dtUtil::InstantiateH<typename dtUtil::CreateTL<P1, P2, P3, P4, P5>::Type, dtUtil::TupleHolder> **ParmsListType**

#### 6.68.2 Member Function Documentation

6.68.2.1 static R **Call** (PObj const & *pobj*, CallType *memfun*, **ParmsListType** & *parms*) [*inline*, *static*]

6.68.2.2 static R **Call** (Fun const & *fun*, **ParmsListType** & *parms*) [*inline*, *static*]

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

- **funcall.h**

## 6.69 FunctorCall< CallType, R, TYPELIST\_6(P1, P2, P3, P4, P5, P6)> Struct Template Reference

```
#include <inc/dtUtil/funcall.h>
```

### Public Types

- typedef dtUtil::InstantiateH< typename **dtUtil::CreateTL**< P1, P2, P3, P4, P5, P6 >::Type, **dtUtil::TupleHolder** > **ParmsListType**

### Static Public Member Functions

- template<class PObj >  
static R **Call** (PObj const &pobj, CallType memfun, **ParmsListType** &parms)
- template<class Fun >  
static R **Call** (Fun const &fun, **ParmsListType** &parms)

```
template<typename CallType, typename R, typename P1, typename P2, typename P3, typename P4, typename P5, typename P6> struct dtUtil::FunctorCall< CallType, R, TYPELIST_6(P1, P2, P3, P4, P5, P6)>
```

#### 6.69.1 Member Typedef Documentation

6.69.1.1 typedef dtUtil::InstantiateH<typename dtUtil::CreateTL<P1, P2, P3, P4, P5, P6>::Type, dtUtil::TupleHolder> **ParmsListType**

#### 6.69.2 Member Function Documentation

6.69.2.1 static R **Call** (PObj const & *pobj*, CallType *memfun*, **ParmsListType** & *parms*) [*inline*, *static*]

6.69.2.2 static R **Call** (Fun const & *fun*, **ParmsListType** & *parms*) [*inline*, *static*]

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

- **funcall.h**

## 6.70 FunctorCall< CallType, R, TYPELIST\_7(P1, P2, P3, P4, P5, P6, P7)> Struct Template Reference

```
#include <inc/dtUtil/funcall.h>
```

### Public Types

- typedef dtUtil::InstantiateH< typename **dtUtil::CreateTL**< P1, P2, P3, P4, P5, P6, P7 >::Type, **dtUtil::TupleHolder** > **ParmsListType**

### Static Public Member Functions

- template<class PObj >  
static R **Call** (PObj const &pobj, CallType memfun, **ParmsListType** &parms)
- template<class Fun >  
static R **Call** (Fun const &fun, **ParmsListType** &parms)

```
template<typename CallType, typename R, typename P1, typename P2, typename P3, typename P4, type-  
name P5, typename P6, typename P7> struct dtUtil::FunctorCall< CallType, R, TYPELIST_7(P1, P2, P3,  
P4, P5, P6, P7)>
```

#### 6.70.1 Member Typedef Documentation

6.70.1.1 typedef dtUtil::InstantiateH<typename dtUtil::CreateTL<P1, P2, P3, P4, P5, P6, P7>::Type, dtUtil::TupleHolder> **ParmsListType**

#### 6.70.2 Member Function Documentation

6.70.2.1 static R **Call** (PObj const & *pobj*, CallType *memfun*, **ParmsListType** & *parms*) [*inline*, *static*]

6.70.2.2 static R **Call** (Fun const & *fun*, **ParmsListType** & *parms*) [*inline*, *static*]

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

- **funcall.h**

## 6.71 FunTraits< R(\*)()> Struct Template Reference

```
#include <inc/dtUtil/funtraits.h>
```

### Public Types

- typedef **NullType ObjType**
- typedef R **ResultType**
- typedef **NullType TypeListType**

```
template<typename R> struct dtUtil::FunTraits< R(*)()>
```

### 6.71.1 Member Typedef Documentation

#### 6.71.1.1 typedef NullType ObjType

#### 6.71.1.2 typedef R ResultType

#### 6.71.1.3 typedef NullType TypeListType

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

- **funtraits.h**

## 6.72 FunTraits< R(\*) (P1)> Struct Template Reference

```
#include <inc/dtUtil/funtraits.h>
```

### Public Types

- typedef **NullType ObjType**
- typedef P1 **Parm1**
- typedef R **ResultType**

### Public Member Functions

- typedef **TYPELIST\_1** (P1) **TypeListType**

```
template<typename R, typename P1> struct dtUtil::FunTraits< R(*) (P1)>
```

### 6.72.1 Member Typedef Documentation

6.72.1.1 typedef **NullType ObjType**

6.72.1.2 typedef P1 **Parm1**

6.72.1.3 typedef R **ResultType**

### 6.72.2 Member Function Documentation

6.72.2.1 typedef **TYPELIST\_1** (P1)

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

- **funtraits.h**

## 6.73 FunTraits< R(\*) (P1, P2)> Struct Template Reference

```
#include <inc/dtUtil/funtraits.h>
```

### Public Types

- typedef **NullType** **ObjType**
- typedef P1 **Parm1**
- typedef P2 **Parm2**
- typedef R **ResultType**

### Public Member Functions

- typedef **TYPELIST\_2** (P1, P2) **TypeListType**

```
template<typename R, typename P1, typename P2> struct dtUtil::FunTraits< R(*) (P1, P2)>
```

#### 6.73.1 Member Typedef Documentation

6.73.1.1 typedef **NullType** **ObjType**

6.73.1.2 typedef P1 **Parm1**

6.73.1.3 typedef P2 **Parm2**

6.73.1.4 typedef R **ResultType**

#### 6.73.2 Member Function Documentation

6.73.2.1 typedef **TYPELIST\_2** (P1, P2)

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

- **funtraits.h**

## 6.74 FunTraits< R(\*) (P1, P2, P3)> Struct Template Reference

```
#include <inc/dtUtil/funtraits.h>
```

### Public Types

- typedef **NullType ObjType**
- typedef P1 **Parm1**
- typedef P2 **Parm2**
- typedef P3 **Parm3**
- typedef R **ResultType**

### Public Member Functions

- typedef **TYPELIST\_3** (P1, P2, P3) **TypeListType**

```
template<typename R, typename P1, typename P2, typename P3> struct dtUtil::FunTraits< R(*) (P1, P2, P3)>
```

#### 6.74.1 Member Typedef Documentation

6.74.1.1 typedef **NullType ObjType**

6.74.1.2 typedef **P1 Parm1**

6.74.1.3 typedef **P2 Parm2**

6.74.1.4 typedef **P3 Parm3**

6.74.1.5 typedef **R ResultType**

#### 6.74.2 Member Function Documentation

6.74.2.1 typedef **TYPELIST\_3** (P1, P2, P3)

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

- **funtraits.h**

## 6.75 FunTraits< R(\*) (P1, P2, P3, P4)> Struct Template Reference

```
#include <inc/dtUtil/funtraits.h>
```

### Public Types

- typedef **NullType** **ObjType**
- typedef P1 **Parm1**
- typedef P2 **Parm2**
- typedef P3 **Parm3**
- typedef P4 **Parm4**
- typedef R **ResultType**

### Public Member Functions

- typedef **TYPELIST\_4** (P1, P2, P3, P4) **TypeListType**

```
template<typename R, typename P1, typename P2, typename P3, typename P4> struct dtUtil::FunTraits<R(*) (P1, P2, P3, P4)>
```

#### 6.75.1 Member Typedef Documentation

6.75.1.1 typedef **NullType** **ObjType**

6.75.1.2 typedef P1 **Parm1**

6.75.1.3 typedef P2 **Parm2**

6.75.1.4 typedef P3 **Parm3**

6.75.1.5 typedef P4 **Parm4**

6.75.1.6 typedef R **ResultType**

#### 6.75.2 Member Function Documentation

6.75.2.1 typedef **TYPELIST\_4** (P1, P2, P3, P4)

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

- **funtraits.h**

## 6.76 FunTraits< R(\*) (P1, P2, P3, P4, P5)> Struct Template Reference

```
#include <inc/dtUtil/funtraits.h>
```

### Public Types

- typedef **NullType ObjType**
- typedef P1 **Parm1**
- typedef P2 **Parm2**
- typedef P3 **Parm3**
- typedef P4 **Parm4**
- typedef P5 **Parm5**
- typedef R **ResultType**

### Public Member Functions

- typedef **TYPELIST\_5** (P1, P2, P3, P4, P5) **TypeListType**

```
template<typename R, typename P1, typename P2, typename P3, typename P4, typename P5> struct dtUtil::FunTraits< R(*) (P1, P2, P3, P4, P5)>
```

#### 6.76.1 Member Typedef Documentation

6.76.1.1 typedef **NullType ObjType**

6.76.1.2 typedef P1 **Parm1**

6.76.1.3 typedef P2 **Parm2**

6.76.1.4 typedef P3 **Parm3**

6.76.1.5 typedef P4 **Parm4**

6.76.1.6 typedef P5 **Parm5**

6.76.1.7 typedef R **ResultType**

#### 6.76.2 Member Function Documentation

6.76.2.1 typedef **TYPELIST\_5** (P1, P2, P3, P4, P5)

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

- **funtraits.h**

## 6.77 FunTraits< R(\*) (P1, P2, P3, P4, P5, P6)> Struct Template Reference

```
#include <inc/dtUtil/funtraits.h>
```

### Public Types

- typedef **NullType ObjType**
- typedef P1 **Parm1**
- typedef P2 **Parm2**
- typedef P3 **Parm3**
- typedef P4 **Parm4**
- typedef P5 **Parm5**
- typedef P6 **Parm6**
- typedef R **ResultType**

### Public Member Functions

- typedef **TYPELIST\_6** (P1, P2, P3, P4, P5, P6) **TypeListType**

```
template<typename R, typename P1, typename P2, typename P3, typename P4, typename P5, typename P6> struct dtUtil::FunTraits< R(*) (P1, P2, P3, P4, P5, P6)>
```

#### 6.77.1 Member Typedef Documentation

6.77.1.1 typedef **NullType ObjType**

6.77.1.2 typedef P1 **Parm1**

6.77.1.3 typedef P2 **Parm2**

6.77.1.4 typedef P3 **Parm3**

6.77.1.5 typedef P4 **Parm4**

6.77.1.6 typedef P5 **Parm5**

6.77.1.7 typedef P6 **Parm6**

6.77.1.8 typedef R **ResultType**

#### 6.77.2 Member Function Documentation

6.77.2.1 typedef **TYPELIST\_6** (P1, P2, P3, P4, P5, P6)

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

- **funtraits.h**

## 6.78 FunTraits< R(\*) (P1, P2, P3, P4, P5, P6, P7)> Struct Template Reference

```
#include <inc/dtUtil/funtraits.h>
```

### Public Types

- typedef **NullType** **ObjType**
- typedef P1 **Parm1**
- typedef P2 **Parm2**
- typedef P3 **Parm3**
- typedef P4 **Parm4**
- typedef P5 **Parm5**
- typedef P6 **Parm6**
- typedef P7 **Parm7**
- typedef R **ResultType**

### Public Member Functions

- typedef **TYPELIST\_7** (P1, P2, P3, P4, P5, P6, P6) **TypeListType**

```
template<typename R, typename P1, typename P2, typename P3, typename P4, typename P5, typename P6, typename P7> struct dtUtil::FunTraits< R(*) (P1, P2, P3, P4, P5, P6, P7)>
```

#### 6.78.1 Member Typedef Documentation

##### 6.78.1.1 typedef NullType ObjType

##### 6.78.1.2 typedef P1 Parm1

##### 6.78.1.3 typedef P2 Parm2

##### 6.78.1.4 typedef P3 Parm3

##### 6.78.1.5 typedef P4 Parm4

##### 6.78.1.6 typedef P5 Parm5

##### 6.78.1.7 typedef P6 Parm6

##### 6.78.1.8 typedef P7 Parm7

##### 6.78.1.9 typedef R ResultType

#### 6.78.2 Member Function Documentation

##### 6.78.2.1 typedef TYPELIST\_7 (P1, P2, P3, P4, P5, P6, P6)

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

- **funtraits.h**

## 6.79 FunTraits< R(O::\*)() const > Struct Template Reference

```
#include <inc/dtUtil/funtraits.h>
```

### Public Types

- typedef O **ObjType**
- typedef R **ResultType**
- typedef **NullType** **TypeListType**

```
template<class O, typename R> struct dtUtil::FunTraits< R(O::*)() const >
```

### 6.79.1 Member Typedef Documentation

#### 6.79.1.1 typedef O ObjType

#### 6.79.1.2 typedef R ResultType

#### 6.79.1.3 typedef NullType TypeListType

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

- **funtraits.h**

## 6.80 FunTraits< R(O::\*)()> Struct Template Reference

```
#include <inc/dtUtil/funtraits.h>
```

### Public Types

- typedef O **ObjType**
- typedef R **ResultType**
- typedef **NullType** **TypeListType**

```
template<class O, typename R> struct dtUtil::FunTraits< R(O::*)()>
```

### 6.80.1 Member Typedef Documentation

#### 6.80.1.1 typedef O ObjType

#### 6.80.1.2 typedef R ResultType

#### 6.80.1.3 typedef NullType TypeListType

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

- **funtraits.h**

## 6.81 FunTraits< R(O::\*)(P1) const > Struct Template Reference

```
#include <inc/dtUtil/funtraits.h>
```

### Public Types

- typedef O **ObjType**
- typedef P1 **Parm1**
- typedef R **ResultType**

### Public Member Functions

- typedef **TYPELIST\_1** (P1) **TypeListType**

```
template<class O, typename R, typename P1> struct dtUtil::FunTraits< R(O::*)(P1) const >
```

#### 6.81.1 Member Typedef Documentation

6.81.1.1 typedef O **ObjType**

6.81.1.2 typedef P1 **Parm1**

6.81.1.3 typedef R **ResultType**

#### 6.81.2 Member Function Documentation

6.81.2.1 typedef **TYPELIST\_1** (P1)

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

- **funtraits.h**

## 6.82 FunTraits< R(O::\*)(P1)> Struct Template Reference

```
#include <inc/dtUtil/funtraits.h>
```

### Public Types

- typedef O **ObjType**
- typedef P1 **Parm1**
- typedef R **ResultType**

### Public Member Functions

- typedef TYPELIST\_1 (P1) **TypeListType**

```
template<class O, typename R, typename P1> struct dtUtil::FunTraits< R(O::*)(P1)>
```

### 6.82.1 Member Typedef Documentation

6.82.1.1 typedef O **ObjType**

6.82.1.2 typedef P1 **Parm1**

6.82.1.3 typedef R **ResultType**

### 6.82.2 Member Function Documentation

6.82.2.1 typedef TYPELIST\_1 (P1)

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

- **funtraits.h**

## 6.83 FunTraits< R(O::\*)(P1, P2) const > Struct Template Reference

```
#include <inc/dtUtil/funtraits.h>
```

### Public Types

- typedef O **ObjType**
- typedef P1 **Parm1**
- typedef P2 **Parm2**
- typedef R **ResultType**

### Public Member Functions

- typedef **TYPELIST\_2** (P1, P2) **TypeListType**

```
template<class O, typename R, typename P1, typename P2> struct dtUtil::FunTraits< R(O::*)(P1, P2) const >
```

#### 6.83.1 Member Typedef Documentation

6.83.1.1 typedef O **ObjType**

6.83.1.2 typedef P1 **Parm1**

6.83.1.3 typedef P2 **Parm2**

6.83.1.4 typedef R **ResultType**

#### 6.83.2 Member Function Documentation

6.83.2.1 typedef **TYPELIST\_2** (P1, P2)

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

- **funtraits.h**

## 6.84 FunTraits< R(O::\*)(P1, P2)> Struct Template Reference

```
#include <inc/dtUtil/funtraits.h>
```

### Public Types

- typedef O **ObjType**
- typedef P1 **Parm1**
- typedef P2 **Parm2**
- typedef R **ResultType**

### Public Member Functions

- typedef **TYPELIST\_2** (P1, P2) **TypeListType**

```
template<class O, typename R, typename P1, typename P2> struct dtUtil::FunTraits< R(O::*)(P1, P2)>
```

#### 6.84.1 Member Typedef Documentation

6.84.1.1 typedef O **ObjType**

6.84.1.2 typedef P1 **Parm1**

6.84.1.3 typedef P2 **Parm2**

6.84.1.4 typedef R **ResultType**

#### 6.84.2 Member Function Documentation

6.84.2.1 typedef **TYPELIST\_2** (P1, P2)

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

- **funtraits.h**

## 6.85 FunTraits< R(O::\*)(P1, P2, P3) const > Struct Template Reference

```
#include <inc/dtUtil/funtraits.h>
```

### Public Types

- typedef O **ObjType**
- typedef P1 **Parm1**
- typedef P2 **Parm2**
- typedef P3 **Parm3**
- typedef R **ResultType**

### Public Member Functions

- typedef TYPELIST\_3 (P1, P2, P3) **TypeListType**

```
template<class O, typename R, typename P1, typename P2, typename P3> struct dtUtil::FunTraits<R(O::*)(P1, P2, P3) const >
```

#### 6.85.1 Member Typedef Documentation

6.85.1.1 typedef O **ObjType**

6.85.1.2 typedef P1 **Parm1**

6.85.1.3 typedef P2 **Parm2**

6.85.1.4 typedef P3 **Parm3**

6.85.1.5 typedef R **ResultType**

#### 6.85.2 Member Function Documentation

6.85.2.1 typedef TYPELIST\_3 (P1, P2, P3)

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

- **funtraits.h**

## 6.86 FunTraits< R(O::\*)(P1, P2, P3)> Struct Template Reference

```
#include <inc/dtUtil/funtraits.h>
```

### Public Types

- typedef O **ObjType**
- typedef P1 **Parm1**
- typedef P2 **Parm2**
- typedef P3 **Parm3**
- typedef R **ResultType**

### Public Member Functions

- typedef TYPELIST\_3 (P1, P2, P3) **TypeListType**

```
template<class O, typename R, typename P1, typename P2, typename P3> struct dtUtil::FunTraits<R(O::*)(P1, P2, P3)>
```

#### 6.86.1 Member Typedef Documentation

6.86.1.1 typedef O **ObjType**

6.86.1.2 typedef P1 **Parm1**

6.86.1.3 typedef P2 **Parm2**

6.86.1.4 typedef P3 **Parm3**

6.86.1.5 typedef R **ResultType**

#### 6.86.2 Member Function Documentation

6.86.2.1 typedef TYPELIST\_3 (P1, P2, P3)

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

- **funtraits.h**

## 6.87 FunTraits< R(O::\*)(P1, P2, P3, P4) const > Struct Template Reference

```
#include <inc/dtUtil/funtraits.h>
```

### Public Types

- typedef O **ObjType**
- typedef P1 **Parm1**
- typedef P2 **Parm2**
- typedef P3 **Parm3**
- typedef P4 **Parm4**
- typedef R **ResultType**

### Public Member Functions

- typedef **TYPELIST\_4** (P1, P2, P3, P4) **TypeListType**

```
template<class O, typename R, typename P1, typename P2, typename P3, typename P4> struct dtUtil::FunTraits< R(O::*)(P1, P2, P3, P4) const >
```

#### 6.87.1 Member Typedef Documentation

6.87.1.1 typedef O **ObjType**

6.87.1.2 typedef P1 **Parm1**

6.87.1.3 typedef P2 **Parm2**

6.87.1.4 typedef P3 **Parm3**

6.87.1.5 typedef P4 **Parm4**

6.87.1.6 typedef R **ResultType**

#### 6.87.2 Member Function Documentation

6.87.2.1 typedef **TYPELIST\_4** (P1, P2, P3, P4)

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

- **funtraits.h**

## 6.88 FunTraits< R(O::\*)(P1, P2, P3, P4)> Struct Template Reference

```
#include <inc/dtUtil/funtraits.h>
```

### Public Types

- typedef O **ObjType**
- typedef P1 **Parm1**
- typedef P2 **Parm2**
- typedef P3 **Parm3**
- typedef P4 **Parm4**
- typedef R **ResultType**

### Public Member Functions

- typedef **TYPELIST\_4** (P1, P2, P3, P4) **TypeListType**

```
template<class O, typename R, typename P1, typename P2, typename P3, typename P4> struct dtUtil::FunTraits< R(O::*)(P1, P2, P3, P4)>
```

#### 6.88.1 Member Typedef Documentation

6.88.1.1 typedef O **ObjType**

6.88.1.2 typedef P1 **Parm1**

6.88.1.3 typedef P2 **Parm2**

6.88.1.4 typedef P3 **Parm3**

6.88.1.5 typedef P4 **Parm4**

6.88.1.6 typedef R **ResultType**

#### 6.88.2 Member Function Documentation

6.88.2.1 typedef **TYPELIST\_4** (P1, P2, P3, P4)

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

- **funtraits.h**

## 6.89 FunTraits< R(O::\*)(P1, P2, P3, P4, P5) const > Struct Template Reference

```
#include <inc/dtUtil/funtraits.h>
```

### Public Types

- typedef P1 **Parm1**
- typedef P2 **Parm2**
- typedef P3 **Parm3**
- typedef P4 **Parm4**
- typedef P5 **Parm5**
- typedef R **ResultType**

### Public Member Functions

- typedef **TYPELIST\_5** (P1, P2, P3, P4, P5) **TypeListType**

```
template<class O, typename R, typename P1, typename P2, typename P3, typename P4, typename P5>  
struct dtUtil::FunTraits< R(O::*)(P1, P2, P3, P4, P5) const >
```

#### 6.89.1 Member Typedef Documentation

6.89.1.1 typedef P1 **Parm1**

6.89.1.2 typedef P2 **Parm2**

6.89.1.3 typedef P3 **Parm3**

6.89.1.4 typedef P4 **Parm4**

6.89.1.5 typedef P5 **Parm5**

6.89.1.6 typedef R **ResultType**

#### 6.89.2 Member Function Documentation

6.89.2.1 typedef **TYPELIST\_5** (P1, P2, P3, P4, P5)

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

- **funtraits.h**

## 6.90 FunTraits< R(O::\*)(P1, P2, P3, P4, P5)> Struct Template Reference

```
#include <inc/dtUtil/funtraits.h>
```

### Public Types

- typedef P1 **Parm1**
- typedef P2 **Parm2**
- typedef P3 **Parm3**
- typedef P4 **Parm4**
- typedef P5 **Parm5**
- typedef R **ResultType**

### Public Member Functions

- typedef **TYPELIST\_5** (P1, P2, P3, P4, P5) TypeListType

```
template<class O, typename R, typename P1, typename P2, typename P3, typename P4, typename P5>  
struct dtUtil::FunTraits< R(O::*)(P1, P2, P3, P4, P5)>
```

#### 6.90.1 Member Typedef Documentation

6.90.1.1 typedef P1 **Parm1**

6.90.1.2 typedef P2 **Parm2**

6.90.1.3 typedef P3 **Parm3**

6.90.1.4 typedef P4 **Parm4**

6.90.1.5 typedef P5 **Parm5**

6.90.1.6 typedef R **ResultType**

#### 6.90.2 Member Function Documentation

6.90.2.1 typedef **TYPELIST\_5** (P1, P2, P3, P4, P5)

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

- **funtraits.h**

## 6.91 FunTraits< R(O::\*)(P1, P2, P3, P4, P5, P6) const > Struct Template Reference

```
#include <inc/dtUtil/funtraits.h>
```

### Public Types

- typedef P1 **Parm1**
- typedef P2 **Parm2**
- typedef P3 **Parm3**
- typedef P4 **Parm4**
- typedef P5 **Parm5**
- typedef P6 **Parm6**
- typedef R **ResultType**

### Public Member Functions

- typedef **TYPELIST\_6** (P1, P2, P3, P4, P5, P6) **TypeListType**

```
template<class O, typename R, typename P1, typename P2, typename P3, typename P4, typename P5,
typename P6> struct dtUtil::FunTraits< R(O::*)(P1, P2, P3, P4, P5, P6) const >
```

#### 6.91.1 Member Typedef Documentation

6.91.1.1 typedef P1 **Parm1**

6.91.1.2 typedef P2 **Parm2**

6.91.1.3 typedef P3 **Parm3**

6.91.1.4 typedef P4 **Parm4**

6.91.1.5 typedef P5 **Parm5**

6.91.1.6 typedef P6 **Parm6**

6.91.1.7 typedef R **ResultType**

#### 6.91.2 Member Function Documentation

6.91.2.1 typedef **TYPELIST\_6** (P1, P2, P3, P4, P5, P6)

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

- **funtraits.h**

## 6.92 FunTraits< R(O::\*)(P1, P2, P3, P4, P5, P6)> Struct Template Reference

```
#include <inc/dtUtil/funtraits.h>
```

### Public Types

- typedef P1 **Parm1**
- typedef P2 **Parm2**
- typedef P3 **Parm3**
- typedef P4 **Parm4**
- typedef P5 **Parm5**
- typedef P6 **Parm6**
- typedef R **ResultType**

### Public Member Functions

- typedef **TYPELIST\_6** (P1, P2, P3, P4, P5, P6) **TypeListType**

```
template<class O, typename R, typename P1, typename P2, typename P3, typename P4, typename P5,  
typename P6> struct dtUtil::FunTraits< R(O::*)(P1, P2, P3, P4, P5, P6)>
```

#### 6.92.1 Member Typedef Documentation

6.92.1.1 typedef P1 **Parm1**

6.92.1.2 typedef P2 **Parm2**

6.92.1.3 typedef P3 **Parm3**

6.92.1.4 typedef P4 **Parm4**

6.92.1.5 typedef P5 **Parm5**

6.92.1.6 typedef P6 **Parm6**

6.92.1.7 typedef R **ResultType**

#### 6.92.2 Member Function Documentation

6.92.2.1 typedef **TYPELIST\_6** (P1, P2, P3, P4, P5, P6)

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

- **funtraits.h**

## 6.93 FunTraits< R(O::\*)(P1, P2, P3, P4, P5, P6, P7) const > Struct Template Reference

```
#include <inc/dtUtil/funtraits.h>
```

### Public Types

- typedef P1 **Parm1**
- typedef P2 **Parm2**
- typedef P3 **Parm3**
- typedef P4 **Parm4**
- typedef P5 **Parm5**
- typedef P6 **Parm6**
- typedef P7 **Parm7**
- typedef R **ResultType**

### Public Member Functions

- typedef **TYPELIST\_7** (P1, P2, P3, P4, P5, P6, P7) **TypeListType**

```
template<class O, typename R, typename P1, typename P2, typename P3, typename P4, typename P5,
typename P6, typename P7> struct dtUtil::FunTraits< R(O::*)(P1, P2, P3, P4, P5, P6, P7) const >
```

#### 6.93.1 Member Typedef Documentation

6.93.1.1 typedef P1 **Parm1**

6.93.1.2 typedef P2 **Parm2**

6.93.1.3 typedef P3 **Parm3**

6.93.1.4 typedef P4 **Parm4**

6.93.1.5 typedef P5 **Parm5**

6.93.1.6 typedef P6 **Parm6**

6.93.1.7 typedef P7 **Parm7**

6.93.1.8 typedef R **ResultType**

#### 6.93.2 Member Function Documentation

6.93.2.1 typedef **TYPELIST\_7** (P1, P2, P3, P4, P5, P6, P7)

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

- **funtraits.h**

## 6.94 FunTraits< R(O::\*)(P1, P2, P3, P4, P5, P6, P7)> Struct Template Reference

```
#include <inc/dtUtil/funtraits.h>
```

### Public Types

- typedef P1 **Parm1**
- typedef P2 **Parm2**
- typedef P3 **Parm3**
- typedef P4 **Parm4**
- typedef P5 **Parm5**
- typedef P6 **Parm6**
- typedef P7 **Parm7**
- typedef R **ResultType**

### Public Member Functions

- typedef **TYPELIST\_7** (P1, P2, P3, P4, P5, P6, P7) **TypeListType**

```
template<class O, typename R, typename P1, typename P2, typename P3, typename P4, typename P5,  
typename P6, typename P7> struct dtUtil::FunTraits< R(O::*)(P1, P2, P3, P4, P5, P6, P7)>
```

#### 6.94.1 Member Typedef Documentation

6.94.1.1 typedef P1 **Parm1**

6.94.1.2 typedef P2 **Parm2**

6.94.1.3 typedef P3 **Parm3**

6.94.1.4 typedef P4 **Parm4**

6.94.1.5 typedef P5 **Parm5**

6.94.1.6 typedef P6 **Parm6**

6.94.1.7 typedef P7 **Parm7**

6.94.1.8 typedef R **ResultType**

#### 6.94.2 Member Function Documentation

6.94.2.1 typedef **TYPELIST\_7** (P1, P2, P3, P4, P5, P6, P7)

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

- **funtraits.h**

## 6.95 GeometryCollector Class Reference

```
#include <inc/dtUtil/geometrycollector.h>
```

### Public Member Functions

- **GeometryCollector** ()
- virtual void **apply** (osg::Geode &node)

### Public Attributes

- std::vector< osg::Geometry \* > **mGeomList**

### 6.95.1 Constructor & Destructor Documentation

6.95.1.1 **GeometryCollector** () [inline]

### 6.95.2 Member Function Documentation

6.95.2.1 virtual void **apply** (osg::Geode & *node*) [inline, virtual]

### 6.95.3 Member Data Documentation

6.95.3.1 std::vector<osg::Geometry\*> **mGeomList**

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

- **geometrycollector.h**

## 6.96 GetElementType< T > Struct Template Reference

Template Function that is used to get the RefPtr's Element type from the map.

```
#include <inc/dtUtil/collectorutil.h>
```

### Public Types

- typedef T::allocator\_type::value\_type::second\_type::element\_type **value\_type**

### 6.96.1 Detailed Description

```
template<typename T> struct dtUtil::CollectorUtil::GetElementType< T >
```

Template Function that is used to get the RefPtr's Element type from the map. In other words this function gets the type of the Node or the Object that is stored in the map.

### 6.96.2 Member Typedef Documentation

#### 6.96.2.1 typedef T::allocator\_type::value\_type::second\_type::element\_type value\_type

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

- collectorutil.h

## 6.97 GroupVisitor Class Reference

### Public Member Functions

- **GroupVisitor** (**NodeCollector** \*NewNodeCollector, **NodeCollector::NodeFlag** mask, const std::string &nodeNameIgnored)  
*Constructor for the **GroupVisitor** (p. 166) Class.*
- virtual void **apply** (osg::LOD &pLOD)  
*Function that Checks to see if the LOD Node is a node that we wish to collect and than traverses down the to any of the LOD Nodes children.*
- virtual void **apply** (osgSim::MultiSwitch &pMultiSwitch)  
*Function that Checks to see if the MultiSwitch Node is a node that we wish to collect and than traverses down the to any of the MultiSwitch Nodes children.*
- virtual void **apply** (osg::Switch &pSwitch)  
*Function that Checks to see if the Switch Node is a node that we wish to collect and than traverses down the to any of the Switch Nodes children.*
- virtual void **apply** (osg::MatrixTransform &matrix\_transform)  
*Function that Checks to see if the MatrixTransform Node is a node that we wish to collect and than traverses down the to any of the MatrixTransform Nodes children.*
- virtual void **apply** (osg::Transform &transform)  
*Function that Checks to see if the Transform Node is a node that we wish to collect and than traverses down the to any of the Transform Nodes children.*
- virtual void **apply** (osg::Group &group)  
*Function that Checks to see if the Group Node is a node that we wish to collect and than traverses down the to any of the Group Nodes children.*
- virtual void **apply** (osg::Geode &geode)  
*Function that traverses through a geode and visits all of the Drawable Objects associated with the Geode and the Material Objects that are associated with the Drawable Objects.*

### 6.97.1 Constructor & Destructor Documentation

#### 6.97.1.1 GroupVisitor (NodeCollector \* NewNodeCollector, NodeCollector::NodeFlag mask, const std::string & nodeNameIgnored) [inline]

Constructor for the **GroupVisitor** (p. 166) Class. Parameters

**NewNodeCollector** The **NodeCollector** (p. 218) Object that it is going to manage

**mask** The different node / object types that are going to be looked for

**nodeNameIgnored** A name of a node / object that you don't want to look for

### 6.97.2 Member Function Documentation

#### 6.97.2.1 virtual void apply (osg::LOD & pLOD) [inline, virtual]

Function that Checks to see if the LOD Node is a node that we wish to collect and than traverses down the to any of the LOD Nodes children. Parameters

**LOD** A LOD Node Object that will be checked

**6.97.2.2 virtual void apply (osgSim::MultiSwitch & pMultiSwitch) [inline, virtual]**

Function that Checks to see if the MultiSwitch Node is a node that we wish to collect and than traverses down the to any of the MultiSwitch Nodes children. Parameters

**MultiSwitch** A MultiSwitch Node Object that will be checked

**6.97.2.3 virtual void apply (osg::Switch & pSwitch) [inline, virtual]**

Function that Checks to see if the Switch Node is a node that we wish to collect and than traverses down the to any of the Switch Nodes children. Parameters

**switch** A Switch Node Object that will be checked

**6.97.2.4 virtual void apply (osg::MatrixTransform & matrix\_transform) [inline, virtual]**

Function that Checks to see if the MatrixTransform Node is a node that we wish to collect and than traverses down the to any of the MatrixTransform Nodes children. Parameters

**matrix\_transform** A MatrixTransform Node Object that will be checked

**6.97.2.5 virtual void apply (osg::Transform & transform) [inline, virtual]**

Function that Checks to see if the Transform Node is a node that we wish to collect and than traverses down the to any of the Transform Nodes children. Parameters

**transform** A Transform Node Object that will be checked

**6.97.2.6 virtual void apply (osg::Group & group) [inline, virtual]**

Function that Checks to see if the Group Node is a node that we wish to collect and than traverses down the to any of the Group Nodes children. Parameters

**group** A Group Node Object that will be checked

**6.97.2.7 virtual void apply (osg::Geode & geode) [inline, virtual]**

Function that traverses through a geode and visits all of the Drawable Objects associated with the Geode and the Material Objects that are associated with the Drawable Objects. If it finds an Object that it is looking for than it will pass this on to be added to the Objects respective map. Parameters

**geode** A Geode Object that you wish to visit

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

- **nodecollector.cpp**

## 6.98 HotSpotDefinition Struct Reference

```
#include <inc/dtUtil/hotspotdefinition.h>
```

### Public Attributes

- `osg::Quat mLocalRotation`  
*the desired orientation of the HotSpot in the parent frame.*
- `osg::Vec3 mLocalTranslation`  
*the desired position of the HotSpot in the parent frame.*
- `std::string mName`  
*an identifier for this "hot spot" instance*
- `std::string mParentName`  
*a coordinate frame identifier for the parent system.*

### 6.98.1 Member Data Documentation

#### 6.98.1.1 `osg::Quat mLocalRotation`

the desired orientation of the HotSpot in the parent frame.

#### 6.98.1.2 `osg::Vec3 mLocalTranslation`

the desired position of the HotSpot in the parent frame.

#### 6.98.1.3 `std::string mName`

an identifier for this "hot spot" instance

#### 6.98.1.4 `std::string mParentName`

a coordinate frame identifier for the parent system.

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

- `hotspotdefinition.h`

## 6.99 HotSpotFileHandler Class Reference

```
#include <inc/dtUtil/hotspotxml.h>
```

### Public Types

- typedef std::vector< **HotSpotDefinition** > **HotSpotDefinitionVector**

### Public Member Functions

- **HotSpotFileHandler** ()
- **~HotSpotFileHandler** ()
- void **characters** (const XMLCh \*const chars, const unsigned int length)
- void **endDocument** ()
- void **endElement** (const XMLCh \*const uri, const XMLCh \*const localname, const XMLCh \*const qname)
- void **endPrefixMapping** (const XMLCh \*const prefix)
- **HotSpotDefinitionVector & GetData** ()
- void **ignorableWhitespace** (const XMLCh \*const chars, const unsigned int length)
- void **processingInstruction** (const XMLCh \*const target, const XMLCh \*const data)
- void **setDocumentLocator** (const XERCES\_CPP\_NAMESPACE\_QUALIFIER Locator \*const locator)
- void **skippedEntity** (const XMLCh \*const name)
- void **startDocument** ()
- void **startElement** (const XMLCh \*const uri, const XMLCh \*const localname, const XMLCh \*const qname, const XERCES\_CPP\_NAMESPACE\_QUALIFIER Attributes &attrs)
- void **startPrefixMapping** (const XMLCh \*const prefix, const XMLCh \*const uri)

### Static Public Attributes

- static const char **DEFAULT\_VALUE** [] = { "default\0" }
- static const osg::Vec3 **HEADING\_VEC**
- static const char **HOT\_SPOT\_NODE\_NAME** [] = { "HotSpot\0" }
- static const char **HOT\_SPOT\_PARENT\_NODE\_NAME** [] = { "Parent\0" }
- static const char **LOCAL\_ROTATION\_NODE\_NAME** [] = { "LocalRotation\0" }
- static const char **LOCAL\_TRANSLATION\_NODE\_NAME** [] = { "LocalTranslation\0" }
- static const char **NAME\_ATTRIBUTE\_NAME** [] = { "name\0" }
- static const osg::Vec3 **PITCH\_VEC**
- static const osg::Vec3 **ROLL\_VEC**

## 6.99.1 Member Typedef Documentation

6.99.1.1 `typedef std::vector<HotSpotDefinition> HotSpotDefinitionVector`

## 6.99.2 Constructor & Destructor Documentation

6.99.2.1 `HotSpotFileHandler ()`

6.99.2.2 `~HotSpotFileHandler ()`

## 6.99.3 Member Function Documentation

6.99.3.1 `void characters (const XMLCh *const chars, const unsigned int length)`

6.99.3.2 `void endDocument () [inline]`

6.99.3.3 `void endElement (const XMLCh *const uri, const XMLCh *const localname, const XMLCh *const qname)`

6.99.3.4 `void endPrefixMapping (const XMLCh *const prefix) [inline]`

6.99.3.5 `HotSpotFileHandler::HotSpotDefinitionVector & GetData ()`

6.99.3.6 `void ignorableWhitespace (const XMLCh *const chars, const unsigned int length) [inline]`

6.99.3.7 `void processingInstruction (const XMLCh *const target, const XMLCh *const data) [inline]`

6.99.3.8 `void setDocumentLocator (const XERCES_CPP_NAMESPACE_QUALIFIER Locator *const locator) [inline]`

6.99.3.9 `void skippedEntity (const XMLCh *const name) [inline]`

6.99.3.10 `void startDocument () [inline]`

6.99.3.11 `void startElement (const XMLCh *const uri, const XMLCh *const localname, const XMLCh *const qname, const XERCES_CPP_NAMESPACE_QUALIFIER Attributes & attrs)`

6.99.3.12 `void startPrefixMapping (const XMLCh *const prefix, const XMLCh *const uri) [inline]`

## 6.99.4 Member Data Documentation

6.99.4.1 `const char DEFAULT_VALUE = { "default\0" } [static]`

6.99.4.2 `const osg::Vec3 HEADING_VEC [static]`

6.99.4.3 `const char HOT_SPOT_NODE_NAME = { "HotSpot\0" } [static]`

6.99.4.4 `const char HOT_SPOT_PARENT_NODE_NAME = { "Parent\0" } [static]`

6.99.4.5 `const char LOCAL_ROTATION_NODE_NAME = { "LocalRotation\0" } [static]`

6.99.4.6 `const char LOCAL_TRANSLATION_NODE_NAME = { "LocalTranslation\0" } [static]`

6.99.4.7 `const char NAME_ATTRIBUTE_NAME = { "name\0" } [static]`

6.99.4.8 `const osg::Vec3 PITCH_VEC [static]`

6.99.4.9 `const osg::Vec3 ROLL_VEC [static]`

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

- `hotspotxml.h`
- `hotspotxml.cpp`

## 6.100 IdsFromTL< TL, i > Struct Template Reference

```
#include <inc/dtUtil/generic.h>
```

### Public Types

- `typedef TypeList< Int2Type< i >, typename IdsFromTL< typename TL::Tail, i+1 >::Type > Type`

```
template<class TL, int i = 0> struct dtUtil::IdsFromTL< TL, i >
```

### 6.100.1 Member Typedef Documentation

#### 6.100.1.1 `typedef TypeList<Int2Type<i>, typename IdsFromTL<typename TL::Tail, i+1>::Type> Type`

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

- `generic.h`

## 6.101 IdsFromTL< NullType, i > Struct Template Reference

```
#include <inc/dtUtil/generic.h>
```

### Public Types

- typedef **NullType** Type

```
template<int i> struct dtUtil::IdsFromTL< NullType, i >
```

### 6.101.1 Member Typedef Documentation

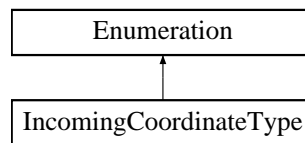
#### 6.101.1.1 typedef **NullType** Type

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

- **generic.h**

## 6.102 IncomingCoordinateType Class Reference

#include <inc/dtUtil/coordinates.h> Inheritance diagram for IncomingCoordinateType::



### Static Public Attributes

- static const **IncomingCoordinateType** GEOCENTRIC
- static const **IncomingCoordinateType** GEODETIC
- static const **IncomingCoordinateType** UTM

### 6.102.1 Member Data Documentation

6.102.1.1 **const IncomingCoordinateType** GEOCENTRIC [static]

6.102.1.2 **const IncomingCoordinateType** GEODETIC [static]

6.102.1.3 **const IncomingCoordinateType** UTM [static]

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

- **coordinates.h**
- **coordinates.cpp**

## 6.103 insert\_back<\_Container, \_InputType > Class Template Reference

```
#include <inc/dtUtil/templateutility.h>
```

### Public Types

- typedef \_Container **container\_type**
- typedef \_InputType **input\_type**

### Public Member Functions

- **insert\_back** (\_Container &\_Cont)
- void **operator()** (input\_type \_Val)

### Protected Attributes

- \_Container \* **container**

```
template<class _Container, class _InputType = typename _Container::const_reference> class dtUtil::insert_back<_Container, _InputType >
```

#### 6.103.1 Member Typedef Documentation

6.103.1.1 typedef \_Container container\_type

6.103.1.2 typedef \_InputType input\_type

#### 6.103.2 Constructor & Destructor Documentation

6.103.2.1 insert\_back (\_Container &\_Cont) [inline, explicit]

#### 6.103.3 Member Function Documentation

6.103.3.1 void operator() (input\_type \_Val) [inline]

#### 6.103.4 Member Data Documentation

6.103.4.1 \_Container\* container [protected]

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

- templateutility.h

## 6.104 insert\_back\_no\_duplicates< \_Container > Class Template Reference

```
#include <inc/dtUtil/templateutility.h>
```

### Public Types

- typedef \_Container **container\_type**
- typedef \_Container::reference **reference**

### Public Member Functions

- **insert\_back\_no\_duplicates** (\_Container &\_Cont)
- bool **operator()** (typename \_Container::const\_reference \_Val)

### Protected Attributes

- \_Container \* **container**

```
template<class _Container> class dtUtil::insert_back_no_duplicates< _Container >
```

#### 6.104.1 Member Typedef Documentation

6.104.1.1 typedef \_Container container\_type

6.104.1.2 typedef \_Container::reference reference

#### 6.104.2 Constructor & Destructor Documentation

6.104.2.1 insert\_back\_no\_duplicates (\_Container &\_Cont) [inline, explicit]

#### 6.104.3 Member Function Documentation

6.104.3.1 bool operator() (typename \_Container::const\_reference \_Val) [inline]

#### 6.104.4 Member Data Documentation

6.104.4.1 \_Container\* container [protected]

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

- templateutility.h

## 6.105 InstantiateH< NullType, Holder, i > Struct Template Reference

```
#include <inc/dtUtil/generic.h>
```

### Public Member Functions

- InstantiateH ()

```
template<template< class, unsigned int > class Holder, unsigned int i> struct dtUtil::InstantiateH< Null-  
Type, Holder, i >
```

### 6.105.1 Constructor & Destructor Documentation

#### 6.105.1.1 InstantiateH () [inline]

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

- generic.h

## 6.106 InstantiateH< TypeList< T, U >, Holder, i > Struct Template Reference

```
#include <inc/dtUtil/generic.h>
```

### Public Types

- enum { **ordern** = i }
- typedef Holder< typename **TypeList**< T, U >::Head, i > **LeftBase**
- typedef InstantiateH< typename **TypeList**< T, U >::Tail, Holder, i+1 > **RightBase**

### Public Member Functions

- **InstantiateH** ()
- **InstantiateH** (typename **TypeList**< T, U >::Head h)
- **InstantiateH** (typename **TypeList**< T, U >::Head h, **NullType**)
- **InstantiateH** (typename **TypeList**< T, U >::Head h, **RightBase** const &t)

```
template<typename T, typename U, template< class, unsigned int > class Holder, unsigned int i> struct dtUtil::InstantiateH< TypeList< T, U >, Holder, i >
```

#### 6.106.1 Member Typedef Documentation

6.106.1.1 typedef Holder<typename **TypeList**<T, U>::Head, i> **LeftBase**

6.106.1.2 typedef InstantiateH<typename **TypeList**<T, U>::Tail, Holder, i+1> **RightBase**

#### 6.106.2 Member Enumeration Documentation

6.106.2.1 anonymous enum

Enumerator:

*ordern*

#### 6.106.3 Constructor & Destructor Documentation

6.106.3.1 **InstantiateH** (typename **TypeList**< T, U >::Head *h*, **RightBase** const & *t*) [inline]

6.106.3.2 **InstantiateH** (typename **TypeList**< T, U >::Head *h*, **NullType**) [inline]

6.106.3.3 **InstantiateH** (typename **TypeList**< T, U >::Head *h*) [inline]

6.106.3.4 **InstantiateH** () [inline]

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

- **generic.h**

## 6.107 InstantiateHAccessor< 0, InstantiateH< TypeList< T, U >, Holder, i >, i > Struct Template Reference

```
#include <inc/dtUtil/generic.h>
```

### Public Types

- typedef InstantiateH< **TypeList**< T, U >, Holder, i > **Instance**
- typedef Instance::LeftBase **TargetHolder**

### Static Public Member Functions

- static **TargetHolder** const & **Get** (**Instance** const &h)
- static **TargetHolder** & **Get** (**Instance** &h)

```
template<typename T, typename U, template< class, unsigned int > class Holder, unsigned int i> struct  
dtUtil::InstantiateHAccessor< 0, InstantiateH< TypeList< T, U >, Holder, i >, i >
```

#### 6.107.1 Member Typedef Documentation

6.107.1.1 typedef InstantiateH<TypeList<T, U>, Holder, i> Instance

6.107.1.2 typedef Instance::LeftBase TargetHolder

#### 6.107.2 Member Function Documentation

6.107.2.1 static TargetHolder const& Get (Instance const & h) [inline, static]

6.107.2.2 static TargetHolder& Get (Instance & h) [inline, static]

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

- **generic.h**

## 6.108 InstantiateHAccessor< j, InstantiateH< TypeList< T, U >, Holder, i >, i > Struct Template Reference

```
#include <inc/dtUtil/generic.h>
```

### Public Types

- typedef InstantiateH< **TypeList**< T, U >, Holder, i > **Instance**
- typedef Instance::RightBase **RightBase**
- typedef Holder< typename TypeAt< **TypeList**< T, U >, j >::Result, j+i > **TargetHolder**

### Static Public Member Functions

- static **TargetHolder** const & **Get** (**Instance** const &h)
- static **TargetHolder** & **Get** (**Instance** &h)

```
template<unsigned int j, typename T, typename U, template< class, unsigned int > class Holder, unsigned int i> struct dtUtil::InstantiateHAccessor< j, InstantiateH< TypeList< T, U >, Holder, i >, i >
```

### 6.108.1 Member Typedef Documentation

6.108.1.1 typedef InstantiateH<TypeList<T, U>, Holder, i> Instance

6.108.1.2 typedef Instance::RightBase RightBase

6.108.1.3 typedef Holder<typename TypeAt<TypeList<T, U>, j>::Result, j+i> TargetHolder

### 6.108.2 Member Function Documentation

6.108.2.1 static TargetHolder const& Get (Instance const & h) [inline, static]

6.108.2.2 static TargetHolder& Get (Instance & h) [inline, static]

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

- generic.h

## 6.109 Int2Type< v > Struct Template Reference

```
#include <inc/dtUtil/generic.h>
```

### Public Types

- enum { **value** = v }

```
template<int v> struct dtUtil::Int2Type< v >
```

### 6.109.1 Member Enumeration Documentation

#### 6.109.1.1 anonymous enum

Enumerator:

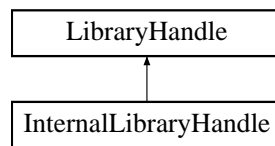
*value*

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

- **generic.h**

## 6.110 InternalLibraryHandle Class Reference

Inheritance diagram for InternalLibraryHandle::



### Public Member Functions

- **InternalLibraryHandle** (const string &libName, **HANDLE** libHandle, bool close)
- virtual **LibrarySharingManager::LibraryHandle::SYMBOL\_ADDRESS FindSymbol** (const string &symbolName) const  
*Looks up a function symbol.*
- virtual **LibrarySharingManager::LibraryHandle::HANDLE GetHandle** () const
- virtual const string & **GetLibName** () const

### Static Public Member Functions

- static dtCore::RefPtr< **LibrarySharingManager::LibraryHandle** > **LoadSharedLibrary** (const string &libraryName)  
*Loads the library and returns a ref pointer to it.*

### Protected Member Functions

- virtual ~**InternalLibraryHandle** ()

#### 6.110.1 Constructor & Destructor Documentation

**6.110.1.1 InternalLibraryHandle** (const string & *libName*, **HANDLE** *libHandle*, bool *close*) [**inline**]

**6.110.1.2 virtual ~InternalLibraryHandle** () [**inline**, **protected**, **virtual**]

#### 6.110.2 Member Function Documentation

**6.110.2.1 virtual LibrarySharingManager::LibraryHandle::SYMBOL\_ADDRESS FindSymbol** (const string & *symbolName*) const [**inline**, **virtual**]

Looks up a function symbol.

Implements **LibraryHandle** (p.200).

**6.110.2.2 virtual LibrarySharingManager::LibraryHandle::HANDLE GetHandle** () const [**inline**, **virtual**]

Implements **LibraryHandle** (p.200).

**6.110.2.3 virtual const string& GetLibName** () const [**inline**, **virtual**]

Returns the system-independent name of the library.

Implements **LibraryHandle** (p.200).

**6.110.2.4 static dtCore::RefPtr<LibrarySharingManager::LibraryHandle> LoadSharedLibrary** (const string & *libraryName*) [**inline**, **static**]

Loads the library and returns a ref pointer to it. This will not reload libraries that are already loaded.

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

- **librarysharingmanager.cpp**

## 6.111 IsConst<\_Type > Struct Template Reference

```
#include <inc/dtUtil/typetraits.h>
```

### Classes

- struct `_IsConst_`
- struct `_IsConst_< const U >`

### Public Types

- enum { `IS_CONST = _IsConst_<_Type>::IS_CONST` }

```
template<typename _Type> struct dtUtil::IsConst<_Type >
```

### 6.111.1 Member Enumeration Documentation

#### 6.111.1.1 anonymous enum

Enumerator:

*IS\_CONST*

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

- `typetraits.h`

## 6.112 IsDelimiter Class Reference

Generic string delimiter check function class.

```
#include <inc/dtUtil/stringutils.h>
```

### Public Member Functions

- **IsDelimiter** (char delim)
- bool **operator()** (char c) const

#### 6.112.1 Detailed Description

Generic string delimiter check function class. Based on the character passed to the constructor, this class will check for that character.

#### 6.112.2 Constructor & Destructor Documentation

6.112.2.1 **IsDelimiter** (char *delim*) [*inline*]

#### 6.112.3 Member Function Documentation

6.112.3.1 **bool operator()** (char *c*) const [*inline*]

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

- **stringutils.h**

## 6.113 IsIntType< T, i > Struct Template Reference

```
#include <inc/dtUtil/generic.h>
```

### Public Types

- enum { **value** = false }

```
template<class T, int i> struct dtUtil::IsIntType< T, i >
```

### 6.113.1 Member Enumeration Documentation

#### 6.113.1.1 anonymous enum

Enumerator:

*value*

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

- **generic.h**

## 6.114 IsIntType< Int2Type< i >, i > Struct Template Reference

```
#include <inc/dtUtil/generic.h>
```

### Public Types

- enum { **value** = true }

```
template<int i> struct dtUtil::IsIntType< Int2Type< i >, i >
```

### 6.114.1 Member Enumeration Documentation

#### 6.114.1.1 anonymous enum

Enumerator:

*value*

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

- **generic.h**

## 6.115 IsPointer<\_Type> Struct Template Reference

```
#include <inc/dtUtil/typetraits.h>
```

### Classes

- struct `_IsPointer_`
- struct `_IsPointer_< U * >`

### Public Types

- enum { `IS_POINTER = _IsPointer_<_Type>::IS_POINTER` }

```
template<typename _Type> struct dtUtil::IsPointer<_Type>
```

### 6.115.1 Member Enumeration Documentation

#### 6.115.1.1 anonymous enum

Enumerator:

*IS\_POINTER*

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

- `typetraits.h`

## 6.116 IsReference<\_Type > Struct Template Reference

```
#include <inc/dtUtil/typetraits.h>
```

### Classes

- struct `_IsReference_`
- struct `_IsReference_< U & >`

### Public Types

- enum { `IS_REFERENCE = _IsReference_<_Type>::IS_REFERENCE` }

```
template<typename _Type> struct dtUtil::IsReference<_Type >
```

### 6.116.1 Member Enumeration Documentation

#### 6.116.1.1 anonymous enum

Enumerator:

***IS\_REFERENCE***

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

- `typetraits.h`

## 6.117 IsSlash Class Reference

Determines if the current character is a forward slash.

```
#include <inc/dtUtil/stringutils.h>
```

### Public Member Functions

- bool **operator()** (char c) const

#### 6.117.1 Detailed Description

Determines if the current character is a forward slash.

#### 6.117.2 Member Function Documentation

##### 6.117.2.1 bool operator() (char c) const [inline]

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

- **stringutils.h**

## 6.118 IsSpace Class Reference

A functor which tests if a character is whitespace.

```
#include <inc/dtUtil/stringutils.h>
```

### Public Member Functions

- **IsSpace** (const std::locale &loc=std::locale(DEFAULT\_LOCALE\_NAME.c\_str()))
- const std::locale & **GetLocale** () const
- bool **operator()** (char c) const

### Static Public Attributes

- static const std::string **DEFAULT\_LOCALE\_NAME**

#### 6.118.1 Detailed Description

A functor which tests if a character is whitespace. This "predicate" needed to have 'state', the locale member.

#### 6.118.2 Constructor & Destructor Documentation

**6.118.2.1 IsSpace (const std::locale & loc = std::locale(DEFAULT\_LOCALE\_NAME.c\_str())) [inline]**

#### 6.118.3 Member Function Documentation

**6.118.3.1 const std::locale& GetLocale () const [inline]**

**6.118.3.2 bool operator() (char c) const [inline]**

#### 6.118.4 Member Data Documentation

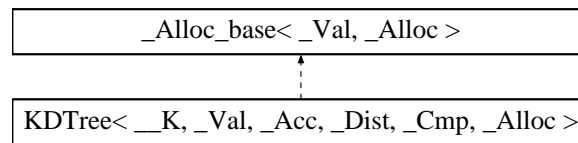
**6.118.4.1 const std::string DEFAULT\_LOCALE\_NAME [static]**

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

- **stringutils.h**
- **stringutils.cpp**

## 6.119 KDTree< \_\_K, \_Val, \_Acc, \_Dist, \_Cmp, \_Alloc > Class Template Reference

#include <inc/dtUtil/kdtree.h> Inheritance diagram for KDTree< \_\_K, \_Val, \_Acc, \_Dist, \_Cmp, \_Alloc >::



### Public Types

- typedef **\_Region**< \_\_K, \_Val, typename \_Acc::result\_type, \_Acc, \_Cmp > **\_Region\_**
- typedef **\_Iterator**< \_Val, **const\_reference**, **const\_pointer** > **const\_iterator**
- typedef **value\_type** const \* **const\_pointer**
- typedef **value\_type** const & **const\_reference**
- typedef std::reverse\_iterator< **const\_iterator** > **const\_reverse\_iterator**
- typedef ptrdiff\_t **difference\_type**
- typedef \_Dist::distance\_type **distance\_type**
- typedef **const\_iterator** **iterator**
- typedef **value\_type** \* **pointer**
- typedef **value\_type** & **reference**
- typedef std::reverse\_iterator< **iterator** > **reverse\_iterator**
- typedef size\_t **size\_type**
- typedef \_Acc::result\_type **subvalue\_type**
- typedef \_Val **value\_type**

### Public Member Functions

- template<typename \_InputIterator >  
**KDTree** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_Acc const &acc=\_Acc(), \_Dist const &\_\_dist=\_Dist(),  
 \_Cmp const &\_\_cmp=\_Cmp(), const **allocator\_type** &\_\_a=**allocator\_type**())
- **KDTree** (const **KDTree** &\_\_x)
- **KDTree** (\_Acc const &\_\_acc=\_Acc(), \_Dist const &\_\_dist=\_Dist(), \_Cmp const &\_\_cmp=\_Cmp(), const  
**allocator\_type** &\_\_a=**allocator\_type**())
- ~**KDTree** ()
- **const\_iterator** **begin** () const
- void **check\_tree** ()
- void **clear** ()
- **size\_type** **count\_within\_range** (\_Region\_ const &\_\_REGION) const
- **size\_type** **count\_within\_range** (**const\_reference** \_\_V, **subvalue\_type** const \_\_R) const
- void **efficient\_replace\_and\_optimize** (std::vector< **value\_type** > &writable\_vector)
- bool **empty** () const
- **const\_iterator** **end** () const
- void **erase** (**const\_iterator** const &\_\_IT)
- void **erase** (**const\_reference** \_\_V)
- void **erase\_exact** (**const\_reference** \_\_V)
- template<class SearchVal >  
**const\_iterator** **find** (SearchVal const &\_\_V) const
- template<class SearchVal >  
**const\_iterator** **find\_exact** (SearchVal const &\_\_V) const
- template<class SearchVal >  
 std::pair< **const\_iterator**, **distance\_type** > **find\_nearest** (SearchVal const &\_\_val, **distance\_type** \_\_max)  
 const
- template<class SearchVal >  
 std::pair< **const\_iterator**, **distance\_type** > **find\_nearest** (SearchVal const &\_\_val) const

- template<class SearchVal , class \_Predicate >  
std::pair< **const\_iterator**, **distance\_type** > **find\_nearest\_if** (SearchVal const &\_\_val, **distance\_type** \_\_max, \_Predicate \_\_p) const
- template<typename \_OutputIterator >  
\_OutputIterator **find\_within\_range** (\_Region\_ const &region, \_OutputIterator out) const
- template<typename SearchVal , typename \_OutputIterator >  
\_OutputIterator **find\_within\_range** (SearchVal const &val, **subvalue\_type** const range, \_OutputIterator out) const
- **const\_iterator** **find\_within\_range\_iterative** (**const\_reference** \_\_a, **const\_reference** \_\_b)
- **allocator\_type** **get\_allocator** () const
- template<typename \_InputIterator >  
void **insert** (**iterator** \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last)
- void **insert** (**iterator** \_\_pos, **size\_type** \_\_n, const **value\_type** &\_\_x)
- template<class \_InputIterator >  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- **iterator** **insert** (**const\_reference** \_\_V)
- **iterator** **insert** (**iterator**, **const\_reference** \_\_V)
- **size\_type** **max\_size** () const
- **KDTree** & **operator=** (const **KDTree** &\_\_x)
- void **optimize** ()
- **const\_reverse\_iterator** **rbegin** () const
- **const\_reverse\_iterator** **rend** () const
- **size\_type** **size** () const
- **\_Acc** **value\_acc** () const  
*Accessor to the value's elements.*
- **\_Cmp** **value\_comp** () const  
*Comparator for the values in the **KDTree** (p. 190).*
- **\_Dist** & **value\_distance** ()
- const **\_Dist** & **value\_distance** () const  
*Distance calculator between 2 value's element.*
- template<class Visitor >  
Visitor **visit\_within\_range** (\_Region\_ const &REGION, Visitor visitor) const
- template<typename SearchVal , class Visitor >  
Visitor **visit\_within\_range** (SearchVal const &V, **subvalue\_type** const R, Visitor visitor) const

## Protected Types

- typedef **\_Alloc\_base**< \_Val, \_Alloc > **\_Base**
- typedef **\_Node\_base** const \* **\_Base\_const\_ptr**
- typedef **\_Node\_base** \* **\_Base\_ptr**
- typedef **\_Node**< \_Val > const \* **\_Link\_const\_type**
- typedef **\_Node**< \_Val > \* **\_Link\_type**
- typedef **\_Node\_compare**< \_Val, \_\_Acc, \_\_Cmp > **\_Node\_compare\_**
- typedef **\_Base::allocator\_type** **allocator\_type**

## Protected Member Functions

- void **\_M\_check\_children** (\_Link\_const\_type child, \_Link\_const\_type parent, **size\_type** const level, bool to\_the\_left)
- void **\_M\_check\_node** (\_Link\_const\_type node, **size\_type** const level)
- **size\_type** **\_M\_count\_within\_range** (\_Link\_const\_type \_\_N, \_Region\_ const &\_\_REGION, \_Region\_ const &\_\_BOUNDS, **size\_type** const \_\_L) const
- void **\_M\_delete\_node** (\_Link\_type \_\_p)
- void **\_M\_empty\_initialise** ()
- **\_Link\_type** **\_M\_erase** (\_Link\_type dead\_dad, **size\_type** const level)

- void **\_M\_erase\_subtree** (**\_Link\_type** \_\_n)
- **const\_iterator** **\_M\_find** (**\_Link\_const\_type** node, **const\_reference** value, **size\_type** const level) const
- **const\_iterator** **\_M\_find\_exact** (**\_Link\_const\_type** node, **const\_reference** value, **size\_type** const level) const
- template<typename **\_OutputIterator** >  
**\_OutputIterator** **\_M\_find\_within\_range** (**\_OutputIterator** out, **\_Link\_const\_type** \_\_N, **\_Region\_const** & **\_REGION**, **\_Region\_const** & **\_\_BOUNDS**, **size\_type** const \_\_L) const
- **\_Link\_type** **\_M\_get\_erase\_replacement** (**\_Link\_type** node, **size\_type** const level)
- std::pair< **\_Link\_type**, **size\_type** > **\_M\_get\_j\_max** (std::pair< **\_Link\_type**, **size\_type** > const node, **size\_type** const level)
- std::pair< **\_Link\_type**, **size\_type** > **\_M\_get\_j\_min** (std::pair< **\_Link\_type**, **size\_type** > const node, **size\_type** const level)
- **\_Link\_const\_type** **\_M\_get\_leftmost** () const
- **\_Link\_const\_type** **\_M\_get\_rightmost** () const
- **\_Link\_type** **\_M\_get\_root** ()
- **\_Link\_const\_type** **\_M\_get\_root** () const
- **iterator** **\_M\_insert** (**\_Link\_type** \_\_N, **const\_reference** \_\_V, **size\_type** const \_\_L)
- **iterator** **\_M\_insert\_left** (**\_Link\_type** \_\_N, **const\_reference** \_\_V)
- **iterator** **\_M\_insert\_right** (**\_Link\_type** \_\_N, **const\_reference** \_\_V)
- bool **\_M\_matches\_node** (**\_Link\_const\_type** \_\_N, **const\_reference** \_\_V, **size\_type** \_\_L=0) const
- bool **\_M\_matches\_node\_in\_d** (**\_Link\_const\_type** \_\_N, **const\_reference** \_\_V, **size\_type** const \_\_L) const
- bool **\_M\_matches\_node\_in\_other\_ds** (**\_Link\_const\_type** \_\_N, **const\_reference** \_\_V, **size\_type** const \_\_L=0) const
- **\_Link\_type** **\_M\_new\_node** (**const\_reference** \_\_V, **\_Base\_ptr** const \_\_PARENT=NULL, **\_Base\_ptr** const \_\_LEFT=NULL, **\_Base\_ptr** const \_\_RIGHT=NULL)
- template<typename **\_Iter** >  
void **\_M\_optimise** (**\_Iter** const & \_\_A, **\_Iter** const & \_\_B, **size\_type** const \_\_L)
- void **\_M\_set\_leftmost** (**\_Node\_base** \*a)
- void **\_M\_set\_rightmost** (**\_Node\_base** \*a)
- void **\_M\_set\_root** (**\_Link\_type** n)
- template<class **Visitor** >  
**Visitor** **\_M\_visit\_within\_range** (**Visitor** visitor, **\_Link\_const\_type** N, **\_Region\_const** & **REGION**, **\_Region\_const** & **BOUNDS**, **size\_type** const L) const

### Static Protected Member Functions

- static bool **\_S\_is\_leaf** (**\_Base\_const\_ptr** N)
- static **\_Link\_const\_type** **\_S\_left** (**\_Base\_const\_ptr** N)
- static **\_Link\_type** **\_S\_left** (**\_Base\_ptr** N)
- static **\_Link\_const\_type** **\_S\_maximum** (**\_Link\_const\_type** \_\_X)
- static **\_Link\_const\_type** **\_S\_minimum** (**\_Link\_const\_type** \_\_X)
- static **\_Link\_const\_type** **\_S\_parent** (**\_Base\_const\_ptr** N)
- static **\_Link\_type** **\_S\_parent** (**\_Base\_ptr** N)
- static **\_Link\_const\_type** **\_S\_right** (**\_Base\_const\_ptr** N)
- static **\_Link\_type** **\_S\_right** (**\_Base\_ptr** N)
- static void **\_S\_set\_left** (**\_Base\_ptr** N, **\_Base\_ptr** l)
- static void **\_S\_set\_parent** (**\_Base\_ptr** N, **\_Base\_ptr** p)
- static void **\_S\_set\_right** (**\_Base\_ptr** N, **\_Base\_ptr** r)
- static **const\_reference** **\_S\_value** (**\_Base\_const\_ptr** N)
- static **const\_reference** **\_S\_value** (**\_Link\_const\_type** N)

### Protected Attributes

- **\_Acc** **\_M\_acc**
- **\_Cmp** **\_M\_cmp**
- **size\_type** **\_M\_count**
- **\_Dist** **\_M\_dist**
- **\_Node\_base** **\_M\_header**
- **\_Link\_type** **\_M\_root**

```
template<size_t const __K, typename _Val, typename _Acc = _Bracket_accessor<_Val>, typename _Dist = squared_difference<typename _Acc::result_type, typename _Acc::result_type>, typename _Cmp = std::less<typename _Acc::result_type>, typename _Alloc = std::allocator<_Node<_Val> >> class dtUtil::KDTree< __K, _Val, _Acc, _Dist, _Cmp, _Alloc >
```

## 6.119.1 Member Typedef Documentation

6.119.1.1 typedef `_Alloc_base<_Val, _Alloc> _Base` [protected]

6.119.1.2 typedef `_Node_base const* _Base_const_ptr` [protected]

6.119.1.3 typedef `_Node_base* _Base_ptr` [protected]

Reimplemented from `_Alloc_base<_Val, _Alloc >` (p. 37).

6.119.1.4 typedef `_Node<_Val> const* _Link_const_type` [protected]

6.119.1.5 typedef `_Node<_Val>* _Link_type` [protected]

6.119.1.6 typedef `_Node_compare<_Val, _Acc, _Cmp> _Node_compare_` [protected]

6.119.1.7 typedef `_Region<__K, _Val, typename _Acc::result_type, _Acc, _Cmp> _Region_`

6.119.1.8 typedef `_Base::allocator_type allocator_type` [protected]

Reimplemented from `_Alloc_base<_Val, _Alloc >` (p. 37).

6.119.1.9 typedef `_Iterator<_Val, const_reference, const_pointer>` `const_iterator`

6.119.1.10 typedef `value_type const*` `const_pointer`

6.119.1.11 typedef `value_type const&` `const_reference`

6.119.1.12 typedef `std::reverse_iterator<const_iterator>` `const_reverse_iterator`

6.119.1.13 typedef `ptrdiff_t` `difference_type`

6.119.1.14 typedef `_Dist::distance_type` `distance_type`

6.119.1.15 typedef `const_iterator` `iterator`

6.119.1.16 typedef `value_type*` `pointer`

6.119.1.17 typedef `value_type&` `reference`

6.119.1.18 typedef `std::reverse_iterator<iterator>` `reverse_iterator`

6.119.1.19 typedef `size_t` `size_type`

6.119.1.20 typedef `_Acc::result_type` `subvalue_type`

6.119.1.21 typedef `_Val` `value_type`

## 6.119.2 Constructor & Destructor Documentation

6.119.2.1 `KDTree (_Acc const & __acc = _Acc(), _Dist const & __dist = _Dist(), _Cmp const & __cmp = _Cmp(), const allocator_type & __a = allocator_type())` [inline]

6.119.2.2 `KDTree (const KDTree< __K, _Val, _Acc, _Dist, _Cmp, _Alloc > & __x)` [inline]

6.119.2.3 `KDTree (_InputIterator __first, _InputIterator __last, _Acc const & acc = _Acc(), _Dist const & __dist = _Dist(), _Cmp const & __cmp = _Cmp(), const allocator_type & __a = allocator_type())` [inline]

6.119.2.4 `~KDTree ()` [inline]

## 6.119.3 Member Function Documentation

6.119.3.1 `void _M_check_children (_Link_const_type child, _Link_const_type parent, size_type const level, bool to_the_left)` [inline, protected]

6.119.3.2 `void _M_check_node (_Link_const_type node, size_type const level)` [inline, protected]

6.119.3.3 `size_type _M_count_within_range (_Link_const_type __N, _Region_const & __REGION, _Region_const & __BOUNDS, size_type const __L) const` [inline, protected]

6.119.3.4 `void _M_delete_node (_Link_type __p)` [inline, protected]

6.119.3.5 `void _M_empty_initialise ()` [inline, protected]

6.119.3.6 `_Link_type _M_erase (_Link_type dead_dad, size_type const level)` [inline, protected]

6.119.3.7 `void _M_erase_subtree (_Link_type __n)` [inline, protected]

6.119.3.8 `const_iterator _M_find (_Link_const_type node, const_reference value, size_type const level) const` [inline, protected]

6.119.3.9 `const_iterator _M_find_exact (_Link_const_type node, const_reference value, size_type const level) const` [inline, protected]

6.119.3.10 `_OutputIterator _M_find_within_range (_OutputIterator out, _Link_const_type __N, _Region_const & __REGION, _Region_const & __BOUNDS, size_type const __L) const` [inline, protected]

6.119.3.11 `_Link_type _M_get_erase_replacement (_Link_type node, size_type const level)` [inline, protected]

6.119.3.12 `std::pair<_Link_type, size_type> _M_get_j_max (std::pair<_Link_type, size_type > const node, size_type const level)` [inline, protected]

6.119.3.13 `std::pair<_Link_type, size_type> _M_get_j_min (std::pair<_Link_type, size_type > const node, size_type const level)` [inline, protected]

6.119.3.14 `_Link_const_type _M_get_leftmost () const` [inline, protected]

6.119.3.15 `_Link_const_type _M_get_rightmost () const` [inline, protected]

6.119.3.16 `_Link_type _M_get_root ()` [inline, protected]

6.119.3.64 void insert (iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last) [inline]

6.119.3.65 void insert (iterator \_\_pos, size\_type \_\_n, const value\_type & \_\_x) [inline]

6.119.3.66 void insert (\_InputIterator \_\_first, \_InputIterator \_\_last) [inline]

6.119.3.67 iterator insert (const\_reference \_\_V) [inline]

6.119.3.68 iterator insert (iterator, const\_reference \_\_V) [inline]

6.119.3.69 size\_type max\_size () const [inline]

6.119.3.70 KDTree& operator= (const KDTree< \_\_K, \_\_Val, \_\_Acc, \_\_Dist, \_\_Cmp, \_\_Alloc > & \_\_x) [inline]

6.119.3.71 void optimize () [inline]

6.119.3.72 const\_reverse\_iterator rbegin () const [inline]

6.119.3.73 const\_reverse\_iterator rend () const [inline]

6.119.3.74 size\_type size () const [inline]

6.119.3.75 \_\_Acc value\_acc () const [inline]

Accessor to the value's elements. This accessor shall not be modified, it could invalidate the tree. Returns a copy of the accessor used by the **KDTree** (p. 190).

6.119.3.76 \_\_Cmp value\_comp () const [inline]

Comparator for the values in the **KDTree** (p. 190). The comparator shall not be modified, it could invalidate the tree. Returns a copy of the comparator used by the **KDTree** (p. 190).

6.119.3.77 \_\_Dist& value\_distance () [inline]

6.119.3.78 const \_\_Dist& value\_distance () const [inline]

Distance calculator between 2 value's element. This functor can be modified. It's modification will only affect the behavior of the find and find\_nearest functions. Returns a reference to the distance calculator used by the **KDTree** (p. 190).

6.119.3.79 Visitor visit\_within\_range (\_Region\_ const & *REGION*, Visitor *visitor*) const [inline]

6.119.3.80 Visitor visit\_within\_range (SearchVal const & *V*, subvalue\_type const *R*, Visitor *visitor*) const [inline]

#### 6.119.4 Member Data Documentation

6.119.4.1 \_\_Acc\_M\_acc [protected]

6.119.4.2 \_\_Cmp\_M\_cmp [protected]

6.119.4.3 size\_type\_M\_count [protected]

6.119.4.4 \_\_Dist\_M\_dist [protected]

6.119.4.5 \_\_Node\_base\_M\_header [protected]

6.119.4.6 \_\_Link\_type\_M\_root [protected]

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

- **kdtree.h**

## 6.120 KeyFrameDecoder< RecordableT, FrameDataT > Class Template Reference

A class that fills key frame data with DOM parsing.

```
#include <inc/dtUtil/keyframedecoder.h>
```

### Public Types

- typedef std::vector< dtCore::RefPtr< **FrameDataType** > > **FrameDataPtrContainer**  
*The data to be saved from the object of interest.*
- typedef FrameDataT **FrameDataType**  
*The type of object of interest. RecordableTypes know how to create, serialize, and deserialize FrameDataTypes.*
- typedef std::pair< double, **FrameDataPtrContainer** > **KeyFrame**  
*A container to hold each source's frame data.*
- typedef std::vector< **KeyFrame** > **KeyFrameContainer**  
*The time stamp applied to the entire container of frame data.*
- typedef std::vector< dtCore::RefPtr< **RecordableType** > > **RecordablePtrContainer**  
*The container of KeyFrame data.*
- typedef RecordableT **RecordableType**

### Public Member Functions

- **KeyFrameDecoder** (const **RecordablePtrContainer** &sources, **KeyFrameContainer** &kfc)  
*The container of sources of frame data.*
- ~**KeyFrameDecoder** ()
- void **Walk** (XERCES\_CPP\_NAMESPACE\_QUALIFIER DOMDocument \*doc)

### Protected Member Functions

- void **DecodeFrameStamp** (XERCES\_CPP\_NAMESPACE\_QUALIFIER DOMDocument \*doc, XERCES\_CPP\_NAMESPACE\_QUALIFIER DOMNode \*fs, **FrameDataPtrContainer** &fdc)  
*Walks the frame stamp tag to find each source's data and asks the source to deserialize the XML tag into a data package.*
- **FrameDataType** \* **DecodeSourceData** (XERCES\_CPP\_NAMESPACE\_QUALIFIER DOMDocument \*doc, XERCES\_CPP\_NAMESPACE\_QUALIFIER DOMElement \*e, unsigned int index)

#### 6.120.1 Detailed Description

```
template<typename RecordableT, typename FrameDataT> class dtUtil::KeyFrameDecoder< RecordableT, FrameDataT >
```

A class that fills key frame data with DOM parsing. Developed for the dtCore::Recorder class, it assumes a DOM tree's structure, and uses existing recordable instances to "decode", or deserialize, the XML hierarchy into a data package, known here as a FrameDataType.

Requires the following layout in XML: Top FrameStamp, Attr="Time" Source

"Time" is required to an attribute name, and its value is assumed to be a double precision floating point number. A 'Source' tag is given to a RecordableType instance and deserialized into usable data.

#### 6.120.2 Member Typedef Documentation

##### 6.120.2.1 typedef std::vector< dtCore::RefPtr<FrameDataType> > FrameDataPtrContainer

The data to be saved from the object of interest.

**6.120.2.2 typedef FrameDataT FrameDataType**

The type of object of interest. RecordableTypes know how to create, serialize, and deserialize FrameDataTypes.

**6.120.2.3 typedef std::pair<double,FrameDataPtrContainer> KeyFrame**

A container to hold each source's frame data.

**6.120.2.4 typedef std::vector<KeyFrame> KeyFrameContainer**

The time stamp applied to the entire container of frame data.

**6.120.2.5 typedef std::vector< dtCore::RefPtr<RecordableType> > RecordablePtrContainer**

The container of KeyFrame data.

**6.120.2.6 typedef RecordableT RecordableType****6.120.3 Constructor & Destructor Documentation****6.120.3.1 KeyFrameDecoder (const RecordablePtrContainer & *sources*, KeyFrameContainer & *kfc*) [inline]**

The container of sources of frame data. The constructor. Parameters

***sources*** The RecordableType instances that are needed for deserializing the XML information.

***kfc*** The KeyFrameContainer provided will be appended during loading. If only loaded data is wished to be contained, then the container should be cleared before executing the Walk.

**6.120.3.2 ~KeyFrameDecoder () [inline]****6.120.4 Member Function Documentation****6.120.4.1 void DecodeFrameStamp (XERCES\_CPP\_NAMESPACE\_QUALIFIER DOMDocument \* *doc*, XERCES\_CPP\_NAMESPACE\_QUALIFIER DOMNode \* *fs*, FrameDataPtrContainer & *fdc*) [inline, protected]**

Walks the frame stamp tag to find each source's data and asks the source to deserialize the XML tag into a data package.

**6.120.4.2 FrameDataType\* DecodeSourceData (XERCES\_CPP\_NAMESPACE\_QUALIFIER DOMDocument \* *doc*, XERCES\_CPP\_NAMESPACE\_QUALIFIER DOMELEMENT \* *e*, unsigned int *index*) [inline, protected]****6.120.4.3 void Walk (XERCES\_CPP\_NAMESPACE\_QUALIFIER DOMDocument \* *doc*) [inline]**

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

- **keyframedecoder.h**

## 6.121 Length< NullType > Struct Template Reference

```
#include <inc/dtUtil/generic.h>
```

### Public Types

- enum { **value** = 0 }

```
template<> struct dtUtil::Length< NullType >
```

### 6.121.1 Member Enumeration Documentation

#### 6.121.1.1 anonymous enum

Enumerator:

*value*

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

- **generic.h**

## 6.122 Length< TypeList< T, U > > Struct Template Reference

```
#include <inc/dtUtil/generic.h>
```

### Public Types

- enum { **value** = 1 + Length<U>::value }

```
template<class T, class U> struct dtUtil::Length< TypeList< T, U > >
```

### 6.122.1 Member Enumeration Documentation

#### 6.122.1.1 anonymous enum

Enumerator:

*value*

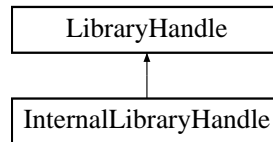
The documentation for this struct was generated from the following file:

- **generic.h**

## 6.123 LibraryHandle Class Reference

pure virtual class abstracting a checked-out handle to a library.

#include <inc/dtUtil/librarysharingmanager.h> Inheritance diagram for LibraryHandle::



### Public Types

- typedef void \* **HANDLE**
- typedef void \* **SYMBOL\_ADDRESS**

### Public Member Functions

- virtual **SYMBOL\_ADDRESS FindSymbol** (const std::string &symbolName) const =0
- virtual **HANDLE GetHandle** () const =0
- virtual const std::string & **GetLibName** () const =0

### Protected Member Functions

- **LibraryHandle** ()
- virtual **~LibraryHandle** ()
- bool **IsShuttingDown** () const
- void **release** ()

#### 6.123.1 Detailed Description

pure virtual class abstracting a checked-out handle to a library. It has both a DynamicLibrary and the system independent name of the library.

#### 6.123.2 Member Typedef Documentation

6.123.2.1 typedef void\* **HANDLE**

6.123.2.2 typedef void\* **SYMBOL\_ADDRESS**

#### 6.123.3 Constructor & Destructor Documentation

6.123.3.1 **LibraryHandle** () [inline, protected]

6.123.3.2 virtual **~LibraryHandle** () [inline, protected, virtual]

#### 6.123.4 Member Function Documentation

6.123.4.1 virtual **SYMBOL\_ADDRESS FindSymbol** (const std::string & *symbolName*) const [pure virtual]

Returns The address of the given symbol or NULL if it was not found.

Implemented in **InternalLibraryHandle** (p. 181).

6.123.4.2 virtual **HANDLE GetHandle** () const [pure virtual]

Implemented in **InternalLibraryHandle** (p. 181).

6.123.4.3 virtual const std::string& **GetLibName** () const [pure virtual]

Returns the system-independent name of the library.

Implemented in **InternalLibraryHandle** (p. 181).

**6.123.4.4 bool IsShuttingDown () const [protected]**

**6.123.4.5 void release () [protected]**

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

- **librarysharingmanager.h**
- **librarysharingmanager.cpp**

## 6.124 LibrarySharingManager Class Reference

Singleton for controlling loading and unloading libraries shared by multiple bodies of code.

```
#include <inc/dtUtil/librarysharingmanager.h>
```

### Classes

- class **ExceptionEnum**
- class **LibraryHandle**  
*pure virtual class abstracting a checked-out handle to a library.*

### Public Member Functions

- void **AddToSearchPath** (const std::string &newPath)  
*Adds a new path to the search path.*
- void **ClearSearchPath** ()  
*Clears the search path completely.*
- const std::string **FindLibraryInSearchPath** (const std::string &libraryFileName) const
- void **GetSearchPath** (std::vector< std::string > &toFill) const  
*This class will search a configured list of paths for a library before letting the OS use its search path.*
- dtCore::RefPtr< **LibraryHandle** > **LoadSharedLibrary** (const std::string &libName, bool assumeModule=false)  
*Loads a library based on a system independent name.*
- void **RemoveFromSearchPath** (const std::string &path)  
*Removes a path from the search path.*

### Static Public Member Functions

- static **LibrarySharingManager &GetInstance** ()
- static std::string **GetPlatformIndependentLibraryName** (const std::string &libName)  
*Strips off the path and platform specific library prefixes and extensions and returns a system independent file name.*
- static std::string **GetPlatformSpecificLibraryName** (const std::string &libBase, bool assumeModule=false)  
*Determines which platform we are running on and returns a platform dependent library name.*

#### 6.124.1 Detailed Description

Singleton for controlling loading and unloading libraries shared by multiple bodies of code. Note This class will search a configured list of paths for a library before letting the OS use its search path. The default list may vary per OS.

#### 6.124.2 Member Function Documentation

##### 6.124.2.1 void AddToSearchPath (const std::string & newPath)

Adds a new path to the search path. It will not convert the path to an absolute path, so be careful when using relative ones. It will also not validate that the path exists so that the app won't blow up if a directory is removed that was being added to the list. Parameters

**newPath** the new path to add.

**6.124.2.2 void ClearSearchPath ()**

Clears the search path completely.

**6.124.2.3 const std::string FindLibraryInSearchPath (const std::string & libraryFileName) const**

Parameters

*the* file name of the library to find.

Note this will not return a path to a library is the OS search path. Returns the path to the library just searching in the search path list

**6.124.2.4 static LibrarySharingManager& GetInstance () [inline, static]****6.124.2.5 string GetPlatformIndependentLibraryName (const std::string & libName) [static]**

Strips off the path and platform specific library prefixes and extensions and returns a system independent file name. Parameters

*libName* The platform specific library name.

Returns A platform independent library name.

**6.124.2.6 string GetPlatformSpecificLibraryName (const std::string & libBase, bool assumeModule = false) [static]**

Determines which platform we are running on and returns a platform dependent library name. Parameters

*libBase* Platform independent library name.

*assumeModule* on some platforms, such as Mac OS X, module library types have a different extension.

Returns A platform dependent library name. Note For example. If the platform independent library name is ExampleActors then on Windows platforms the resulting dependent library name would be ExampleActors.dll, however, on Unix based platforms, the resulting name would be libExampleActors.so.

**6.124.2.7 void GetSearchPath (std::vector< std::string > & toFill) const**

This class will search a configured list of paths for a library before letting the OS use its search path. This method returns the list. The default list may vary per OS. Returns the list of directories this will search for libraries

**6.124.2.8 dtCore::RefPtr< LibrarySharingManager::LibraryHandle > LoadSharedLibrary (const std::string & libName, bool assumeModule = false)**

Loads a library based on a system independent name. The class holds onto already opened libraries, so if the library is being used already, a new handle to it will be returned and no system calls will be made. Parameters

*libName* the system-independent name of the library to load.

*assumeModule* This hint is used for platforms that differentiate between modules and dynamic libraries.

Returns a pointer to a handle representing the library.

Exceptions

*dtUtil::Exception* (p. 116) with key *dtUtil::LibrarySharingManager::ExceptionEnum::LibraryLoadingError* (p. 118) if the library can't be loaded for some reason.

**6.124.2.9 void RemoveFromSearchPath (const std::string & path)**

Removes a path from the search path. Parameters

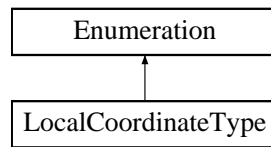
*path* the path to remove.

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

- **librarysharingmanager.h**
- **librarysharingmanager.cpp**

## 6.125 LocalCoordinateType Class Reference

#include <inc/dtUtil/coordinates.h> Inheritance diagram for LocalCoordinateType::



### Static Public Attributes

- static const **LocalCoordinateType** **CARTESIAN**  
*Obsolete, don't use. It will convert to CARTESIAN\_UTM.*
- static const **LocalCoordinateType** **CARTESIAN\_FLAT\_EARTH**  
*the terrain is flattened using a flat earth projection*
- static const **LocalCoordinateType** **CARTESIAN\_UTM**  
*the terrain is flattened using a UTM projection*
- static const **LocalCoordinateType** **GLOBE**  
*The local terrain is a globe or part of a globe so that the XYZ coordinates should be mapped around it.*

### 6.125.1 Member Data Documentation

#### 6.125.1.1 const LocalCoordinateType **CARTESIAN** [static]

Obsolete, don't use. It will convert to CARTESIAN\_UTM.

#### 6.125.1.2 const LocalCoordinateType **CARTESIAN\_FLAT\_EARTH** [static]

the terrain is flattened using a flat earth projection

#### 6.125.1.3 const LocalCoordinateType **CARTESIAN\_UTM** [static]

the terrain is flattened using a UTM projection

#### 6.125.1.4 const LocalCoordinateType **GLOBE** [static]

The local terrain is a globe or part of a globe so that the XYZ coordinates should be mapped around it.

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

- **coordinates.h**
- **coordinates.cpp**

## 6.126 Log Class Reference

**Log** (p. 205) class which the engine uses for all of its logging needs.

```
#include <inc/dtUtil/log.h>
```

### Public Types

- enum **LogMessageType** {  
**LOG\_DEBUG**, **LOG\_INFO**, **LOG\_WARNING**, **LOG\_ERROR**,  
**LOG\_ALWAYS** }  
*The different types of log messages.*
- enum **OutputStreamOptions** { **NO\_OUTPUT** = 0x00000000, **TO\_FILE** = 0x00000001, **TO\_CONSOLE** = 0x00000002, **STANDARD** = **TO\_FILE** | **TO\_CONSOLE** }

### Public Member Functions

- **LogMessageType GetLogLevel** () const
- **LogMessageType GetLogLevelForString** (const std::string &levelString) const
- const std::string **GetLogLevelString** (**LogMessageType** msgType) const
- const std::string & **GetName** () const  
*Returns the name of this logger.*
- unsigned int **GetOutputStreamBit** () const  
*Get the currently defined output stream options.*
- bool **IsLevelEnabled** (**LogMessageType** msgType) const
- void **LogHorizRule** ()  
*Inserts a horizontal rule into the log file.*
- void **LogMessage** (**LogMessageType** msgType, const std::string &source, const char \*msg,...) const  
*Little more sophisticated method for logging messages.*
- void **LogMessage** (**LogMessageType** msgType, const std::string &source, int line, const char \*msg, va\_list list) const  
*Logs a time-stamped message.*
- void **LogMessage** (**LogMessageType** msgType, const std::string &source, int line, const std::string &msg) const  
*Little more sophisticated method for logging messages.*
- void **LogMessage** (**LogMessageType** msgType, const std::string &source, int line, const char \*msg,...) const  
*Little more sophisticated method for logging messages.*
- void **LogMessage** (const std::string &source, int line, const std::string &msg, **LogMessageType** msgType=LOG\_INFO) const  
*Logs a time-stamped message.*
- void **SetLogLevel** (**LogMessageType** msgType)  
*Sets the lowest level of logging that will be logged.*
- void **SetOutputStreamBit** (unsigned int option)  
*Tell the **Log** (p. 205) where to send output messages.*

## Static Public Member Functions

- static **Log & GetInstance** (const std::string &name)
- static **Log & GetInstance** ()

## Protected Member Functions

- **Log** (const std::string &name)  
*Opens the log file and writes the html header information.*
- **~Log** ()  
*Writes any closing html tags and closes the log file.*

### 6.126.1 Detailed Description

**Log** (p. 205) class which the engine uses for all of its logging needs. The log file is formatted using html tags, therefore, any browser should display the log without any problems.

### 6.126.2 Member Enumeration Documentation

#### 6.126.2.1 enum LogMessageType

The different types of log messages.

Enumerator:

**LOG\_DEBUG**  
**LOG\_INFO**  
**LOG\_WARNING**  
**LOG\_ERROR**  
**LOG\_ALWAYS**

#### 6.126.2.2 enum OutputStreamOptions

Enumerator:

**NO\_OUTPUT** **Log** (p. 205) messages don't get written to any device.  
**TO\_FILE** **Log** (p. 205) messages get sent to the output file.  
**TO\_CONSOLE** **Log** (p. 205) messages get sent to the console.  
**STANDARD** The default setting.

### 6.126.3 Constructor & Destructor Documentation

#### 6.126.3.1 **Log** (const std::string & *name*) [protected]

Opens the log file and writes the html header information.

#### 6.126.3.2 **~Log** () [protected]

Writes any closing html tags and closes the log file.

### 6.126.4 Member Function Documentation

#### 6.126.4.1 **Log & GetInstance** (const std::string & *name*) [static]

#### 6.126.4.2 **Log & GetInstance** () [static]

#### 6.126.4.3 **LogMessageType GetLogLevel** () const [inline]

Returns the lowest level of logging that will be logged.

**6.126.4.4 Log::LogLevelType GetLogLevelForString (const std::string & *levelString*) const**

Returns the log level matching a string or WARNING if there is no match.

**6.126.4.5 const std::string GetLogLevelString (Log::LogLevelType *msgType*) const**

Returns a string version of a log level.

**6.126.4.6 const std::string & GetName () const**

Returns the name of this logger.

**6.126.4.7 unsigned int GetOutputStreamBit () const**

Get the currently defined output stream options.

**6.126.4.8 bool IsLevelEnabled (LogLevelType *msgType*) const [inline]**

Returns true if log messages of the given level will be sent to the log output

Parameters

***msgType*** the type of message to query about.

**6.126.4.9 void LogHorizRule ()**

Inserts a horizontal rule into the log file.

**6.126.4.10 void LogMessage (LogLevelType *msgType*, const std::string & *source*, const char \* *msg*, ...) const**

Little more sophisticated method for logging messages. Allows for an unlimited number of parameters in a C-style printf syntax. Parameters

***msgType*** - Type of message being displayed. (error,warning,info)

***source*** - String identifier of the source of the message.

***msg*** - Printf - style format string.

Note Max length of the string to be printed is 2048 characters.

**6.126.4.11 void LogMessage (LogLevelType *msgType*, const std::string & *source*, int *line*, const char \* *msg*, va\_list *list*) const**

Logs a time-stamped message. Takes a variable-argument list (*va\_list*) that was created with *va\_start*. Parameters

***msgType*** - Type of message being displayed. (error,warning,info)

***source*** - String identifier of the source of the message.

***line*** - line number or negative for unknown.

***msg*** - Printf - style format string.

***list*** - *va\_list* created with *va\_start*.

Note Max length of the string to be printed is 2048 characters.

**6.126.4.12 void LogMessage (LogLevelType *msgType*, const std::string & *source*, int *line*, const std::string & *msg*) const**

Little more sophisticated method for logging messages. Allows for an unlimited number of parameters in a C-style printf syntax. Parameters

***msgType*** - Type of message being displayed. (error,warning,info)

***source*** - String identifier of the source of the message.

***line*** - the line number.

***msg*** - std::string that has been formatted.

Note Max length of the string to be printed is 2048 characters.

#### 6.126.4.13 void LogMessage (LogMessageType *msgType*, const std::string & *source*, int *line*, const char \* *msg*, ...) const

Little more sophisticated method for logging messages. Allows for an unlimited number of parameters in a C-style printf syntax. Parameters

***msgType*** - Type of message being displayed. (error,warning,info)

***source*** - String identifier of the source of the message.

***line*** - the line number.

***msg*** - Printf - style format string.

Note Max length of the string to be printed is 2048 characters.

#### 6.126.4.14 void LogMessage (const std::string & *source*, int *line*, const std::string & *msg*, LogMessageType *msgType* = LOG\_INFO) const

Logs a time-stamped message. Parameters

***source*** - String identifier of the source of the message. (\_\_FUNCTION\_\_ is useful here.

***line*** the line number.

***msg*** - Message to display.

***msgType*** - Type of message being displayed. (error,warning,info)

#### 6.126.4.15 void SetLogLevel (LogMessageType *msgType*) [inline]

Sets the lowest level of logging that will be logged. If the level is set to Debug, all messages will be sent. If the level is set error, only errors will be sent. Parameters

***msgType*** the new logging level

#### 6.126.4.16 void SetOutputStreamBit (unsigned int *option*)

Tell the **Log** (p. 205) where to send output messages. The supplied parameter is a bitwise combination of OutputStreamOptions. The default is STANDARD, which directs messages to both the console and the output file. For example, to tell the **Log** (p. 205) to output to the file and console:

```
dtUtil::Log::GetInstance().SetOutputStreamBit(dtUtil::Log::TO_FILE |
dtUtil::Log::TO_CONSOLE);
```

Parameters

***option*** A bitwise combination of options.

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

- **log.h**
- **log.cpp**

## 6.127 LogFile Class Reference

```
#include <inc/dtUtil/log.h>
```

### Static Public Member Functions

- static const std::string **GetFileName** ()  
*Get the current filename of the log file.*
- static const std::string & **GetTitle** ()  
*Get the current HTML title string.*
- static void **SetFileName** (const std::string &name)  
*Change the name of the log file (defaults to "delta3d\_log.html").*
- static void **SetTitle** (const std::string &title)  
*change the title string used in HTML defaults to "Delta 3D Engine Log File" or "Delta 3D Engine Log File (Debug Libs)"*

### 6.127.1 Member Function Documentation

#### 6.127.1.1 const std::string GetFileName () [static]

Get the current filename of the log file.

#### 6.127.1.2 const std::string & GetTitle () [static]

Get the current HTML title string.

#### 6.127.1.3 void SetFileName (const std::string & name) [static]

Change the name of the log file (defaults to "delta3d\_log.html"). This will close the existing file (if opened) and create a new file with the supplied filename.

Parameters

**name** : The name of the new file (will be written using HTML)

#### 6.127.1.4 void SetTitle (const std::string & title) [static]

change the title string used in HTML defaults to "Delta 3D Engine Log File" or "Delta 3D Engine Log File (Debug Libs)"

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

- **log.h**
- **log.cpp**

## 6.128 LogImpl Struct Reference

### Public Member Functions

- **LogImpl** (const std::string &name)

### Public Attributes

- std::string **mName**
- unsigned int **mOutputStreamBit**  
*the current output stream option*

### Static Public Attributes

- static const std::string **mDefaultName**

#### 6.128.1 Constructor & Destructor Documentation

6.128.1.1 **LogImpl** (const std::string & *name*) [*inline*]

#### 6.128.2 Member Data Documentation

6.128.2.1 **const std::string mDefaultName** [*static*]

6.128.2.2 **std::string mName**

6.128.2.3 **unsigned int mOutputStreamBit**

*the current output stream option*

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

- **log.cpp**

## 6.129 LogManager Class Reference

### Public Member Functions

- **LogManager** ()
- **~LogManager** ()
- **bool AddInstance** (const std::string &name, **Log** \*log)
- **void EndFile** ()
- **Log \* GetInstance** (const std::string &name)
- **void OpenFile** ()
- **void TimeTag** (std::string prefix)

### Public Attributes

- std::ofstream **logFile**
- OpenThreads::Mutex **mMutex**

### 6.129.1 Constructor & Destructor Documentation

6.129.1.1 **LogManager** () [inline]

6.129.1.2 **~LogManager** () [inline]

### 6.129.2 Member Function Documentation

6.129.2.1 **bool AddInstance** (const std::string & *name*, **Log** \* *log*) [inline]

6.129.2.2 **void EndFile** () [inline]

6.129.2.3 **Log\*** **GetInstance** (const std::string & *name*) [inline]

6.129.2.4 **void OpenFile** () [inline]

6.129.2.5 **void TimeTag** (std::string *prefix*) [inline]

### 6.129.3 Member Data Documentation

6.129.3.1 std::ofstream **logFile**

6.129.3.2 OpenThreads::Mutex **mMutex**

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

- **log.cpp**

## 6.130 MatrixUtil Class Reference

**MatrixUtil** (p. 212) is a utility class for operating on an `osg::Matrix`.

```
#include <inc/dtUtil/matrixutil.h>
```

### Static Public Member Functions

- static float **ClampUnity** (float x)  
*clamps a float from -1 to 1*
- static `osg::Vec3` **GetRow3** (const `osg::Matrix` &matrix, int row)
- static `osg::Vec4` **GetRow4** (const `osg::Matrix` &matrix, int row)
- static void **HprToMatrix** (`osg::Matrix` &rotation, const `osg::Vec3` &hpr)  
*Translates Euler angles Heading, pitch, and roll to an osg::Matrix.*
- static void **MatrixToHpr** (`osg::Vec3` &hpr, const `osg::Matrix` &rotation)  
*Translates the rotation part of an osg::Matrix to Euler Angles.*
- static void **MatrixToHprAndPosition** (`osg::Vec3` &xyz, `osg::Vec3` &hpr, const `osg::Matrix` &rotation)  
*Translates the rotation part of an osg::Matrix to Euler Angles As well as the translation to a vector.*
- static void **PositionAndHprToMatrix** (`osg::Matrix` &rotation, const `osg::Vec3` &xyz, const `osg::Vec3` &hpr)  
*Translates Euler angles Heading, pitch, and roll to an osg::Matrix in addition fills in the translation of the matrix.*
- static void **Print** (const `osg::Vec4` &vec)  
*Prints a vector.*
- static void **Print** (const `osg::Vec3` &vec)  
*Prints a vector.*
- static void **Print** (const `osg::Matrix` &matrix)  
*Prints a matrix.*
- static void **SetRow** (`osg::Matrix` &matrix, const `osg::Vec4` &vec, int row)
- static void **SetRow** (`osg::Matrix` &matrix, const `osg::Vec3` &vec, int row)
- static void **TransformVec3** (`osg::Vec3` &vec\_in, const `osg::Vec3` &xyz, const `osg::Matrix` &transformMat)  
*This function transforms a point by a 4x4 matrix and stores the result into another vector.*
- static void **TransformVec3** (`osg::Vec3` &xyz, const `osg::Matrix` &transformMat)  
*This function transforms a point by a 4x4 matrix and stores the result back in the point.*
- static void **Transpose** (`osg::Matrix` &dest, const `osg::Matrix` &src)  
*transposes a matrix*

### 6.130.1 Detailed Description

**MatrixUtil** (p. 212) is a utility class for operating on an `osg::Matrix`.

### 6.130.2 Member Function Documentation

#### 6.130.2.1 float ClampUnity (float x) [static]

clamps a float from -1 to 1

**6.130.2.2** `osg::Vec3 GetRow3 (const osg::Matrix & matrix, int row) [static]`

**6.130.2.3** `osg::Vec4 GetRow4 (const osg::Matrix & matrix, int row) [static]`

**6.130.2.4** `void HprToMatrix (osg::Matrix & rotation, const osg::Vec3 & hpr) [static]`

Translates Euler angles Heading, pitch, and roll to an osg::Matrix. Parameters

*rotation*,: the matrix whose rotation will be filled

*hpr*,: the current Heading, pitch, roll to translate to a matrix (values are in degrees)

#### Todo

find a preprocessor way to assign this constant different for the different precision types.

**6.130.2.5** `void MatrixToHpr (osg::Vec3 & hpr, const osg::Matrix & rotation) [static]`

Translates the rotation part of an osg::Matrix to Euler Angles. Parameters

*hpr*,: the vector to fill with the Euler Angles

*rotation*,: the current rotation matrix

**6.130.2.6** `void MatrixToHprAndPosition (osg::Vec3 & xyz, osg::Vec3 & hpr, const osg::Matrix & rotation) [static]`

Translates the rotation part of an osg::Matrix to Euler Angles As well as the translation to a vector. Parameters

*xyz*,: the vector to fill with the translation part of the matrix

*hpr*,: the vector to fill with the Euler Angles

*rotation*,: the current rotation matrix

**6.130.2.7** `void PositionAndHprToMatrix (osg::Matrix & rotation, const osg::Vec3 & xyz, const osg::Vec3 & hpr) [static]`

Translates Euler angles Heading, pitch, and roll to an osg::Matrix in addition fills in the translation of the matrix. Parameters

*rotation*,: the matrix whose rotation will be filled

*xyz*,: the translation to be added to the matrix

*hpr*,: the current Heading, pitch, roll to translate to a matrix

**6.130.2.8** `void Print (const osg::Vec4 & vec) [static]`

Prints a vector.

**6.130.2.9** `void Print (const osg::Vec3 & vec) [static]`

Prints a vector.

**6.130.2.10** `void Print (const osg::Matrix & matrix) [static]`

Prints a matrix.

**6.130.2.11 void SetRow (osg::Matrix & *matrix*, const osg::Vec4 & *vec*, int *row*) [static]**

**6.130.2.12 void SetRow (osg::Matrix & *matrix*, const osg::Vec3 & *vec*, int *row*) [static]**

**6.130.2.13 void TransformVec3 (osg::Vec3 & *vec\_in*, const osg::Vec3 & *xyz*, const osg::Matrix & *transformMat*) [static]**

This function transforms a point by a 4x4 matrix and stores the result into another vector. Parameters

***vec\_in*,**: the vector to store the result of the transform

***xyz*,**: the vector which will be transformed

***transformMat*,**: the 4x4 matrix which will be used to rotate and translate the point

**6.130.2.14 void TransformVec3 (osg::Vec3 & *xyz*, const osg::Matrix & *transformMat*) [static]**

This function transforms a point by a 4x4 matrix and stores the result back in the point. Parameters

***xyz*,**: the vector which will be transformed and have the result stored back in to

***transformMat*,**: the 4x4 matrix which will be used to rotate and translate the point

**6.130.2.15 void Transpose (osg::Matrix & *dest*, const osg::Matrix & *src*) [static]**

transposes a matrix

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

- **matrixutil.h**
- **matrixutil.cpp**

## 6.131 MergeParmsH< i, BoundPTL, UnboundPTL, IdsTL, TL > Struct Template Reference

```
#include <inc/dtUtil/funbind.h>
```

### Classes

- struct **Bound**
- struct **Predicate**
- struct **Predicate**< j, dtUtil::NullType >
- struct **Unbound**

### Public Types

- typedef dtUtil::InstantiateH< TL, dtUtil::TupleHolder, i > **ResultType**

### Static Public Member Functions

- static **ResultType MergeParms** (BoundPTL const &bound, UnboundPTL const &unbound)

```
template<int i, class BoundPTL, class UnboundPTL, class IdsTL, class TL> struct dtUtil::MergeParmsH< i, BoundPTL, UnboundPTL, IdsTL, TL >
```

#### 6.131.1 Member Typedef Documentation

6.131.1.1 typedef dtUtil::InstantiateH<TL, dtUtil::TupleHolder, i> **ResultType**

#### 6.131.2 Member Function Documentation

6.131.2.1 static **ResultType MergeParms** (BoundPTL const & *bound*, UnboundPTL const & *unbound*)  
[inline, static]

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

- **funbind.h**

## 6.132 MergeParmsH< 0, BoundPTL, UnboundPTL, dtUtil::NullType, TL > Struct Template Reference

```
#include <inc/dtUtil/funbind.h>
```

### Static Public Member Functions

- static CallParms< TL >::ParmsListType **MergeParms** (BoundPTL const &, UnboundPTL const &unbound)

```
template<class BoundPTL, class UnboundPTL, class TL> struct dtUtil::MergeParmsH< 0, BoundPTL, UnboundPTL, dtUtil::NullType, TL >
```

### 6.132.1 Member Function Documentation

**6.132.1.1 static CallParms<TL>::ParmsListType MergeParms (BoundPTL const &, UnboundPTL const & *unbound*) [inline, static]**

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

- funbind.h

## 6.133 MergeParamsH< i, BoundPTL, UnboundPTL, IdsTL, dtUtil::NullType > Struct Template Reference

```
#include <inc/dtUtil/funbind.h>
```

### Static Public Member Functions

- static **dtUtil::NullType MergeParams** (BoundPTL const &, UnboundPTL const &)

```
template<int i, class BoundPTL, class UnboundPTL, class IdsTL> struct dtUtil::MergeParamsH< i, BoundPTL, UnboundPTL, IdsTL, dtUtil::NullType >
```

### 6.133.1 Member Function Documentation

6.133.1.1 **static dtUtil::NullType MergeParams** (BoundPTL const &, UnboundPTL const &) [inline, static]

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

- funbind.h

## 6.134 NodeCollector Class Reference

**NodeCollector** (p. 218) is used to gather osg Group nodes, DOFTransform nodes, MatrixTransform nodes, Switch Nodes and Geode nodes (which have Drawable objects and Material objects).

```
#include <inc/dtUtil/nodecollector.h>
```

### Public Types

- typedef std::map< std::string, dtCore::RefPtr< osg::Geode > > **GeodeNodeMap**
- typedef std::map< std::string, dtCore::RefPtr< osg::Group > > **GroupNodeMap**
- typedef std::map< std::string, dtCore::RefPtr< osg::LOD > > **LODNodeMap**
- typedef std::map< std::string, dtCore::RefPtr< osg::MatrixTransform > > **MatrixTransformNodeMap**
- typedef std::map< std::string, dtCore::RefPtr< osgSim::MultiSwitch > > **MultiSwitchNodeMap**
- typedef unsigned **NodeFlag**

*Type Definition that is used to declare flags that allow the user to request searches for different types of nodes or geode nodes.*

- typedef std::map< std::string, dtCore::RefPtr< osg::Switch > > **SwitchNodeMap**
- typedef std::map< std::string, dtCore::RefPtr< osgSim::DOFTransform > > **TransformNodeMap**

### Public Member Functions

- **NodeCollector** (osg::Node \*nodeToLoad, **NodeCollector::NodeFlag** mask, const std::string &nodeNameIgnored="")  
*Constructor that when called will automatically generate the node maps or geode maps that you request.*
- **NodeCollector** ()  
*Blank Constructor that is defined to do nothing.*
- void **AddDOFTransform** (const std::string &name, osgSim::DOFTransform &node)  
*Function that is used to add a DOFTransform Node to the DOFTransform Node map.*
- void **AddGeode** (const std::string &name, osg::Geode &node)  
*Function that is used to add a Geode Node to the Geode Node map.*
- void **AddGroup** (const std::string &name, osg::Group &node)  
*Function that is used to add a Group Node to the Group Node map.*
- void **AddLOD** (const std::string &name, osg::LOD &node)  
*Function that is used to add an LOD Node to the LOD Node map.*
- void **AddMatrixTransform** (const std::string &name, osg::MatrixTransform &node)  
*Function that is used to add a MatrixTransform Node to the MatrixTransform Node map.*
- void **AddMultiSwitch** (const std::string &name, osgSim::MultiSwitch &node)  
*Function that is used to add a MultiSwitch Node to the MultiSwitch Node map.*
- void **AddSwitch** (const std::string &name, osg::Switch &node)  
*Function that is used to add a Switch Node to the Switch Node map.*
- void **ClearAll** ()  
*Function that is defined to clear all the maps of their contents.*
- void **CollectNodes** (osg::Node \*NodeToLoad, **NodeCollector::NodeFlag** mask, const std::string &nodeNameIgnored="")  
*Function that when called will automatically generate the node maps or geode maps that you request.*

- `osgSim::DOFTransform * GetDOFTransform (const std::string &name)`  
*Function that is used to request a pointer to a DOFTransform Node.*
- `const osgSim::DOFTransform * GetDOFTransform (const std::string &name) const`  
*Function that is used to request a CONST pointer to a DOFTransform Node.*
- `osg::Geode * GetGeode (const std::string &name)`  
*Function that is used to request a pointer to a Geode Node.*
- `const osg::Geode * GetGeode (const std::string &name) const`  
*Function that is used to request a CONST pointer to a Geode Node.*
- `NodeCollector::GeodeNodeMap & GetGeodeNodeMap ()`  
*Function that returns a Geode map.*
- `const NodeCollector::GeodeNodeMap & GetGeodeNodeMap () const`  
*Function that returns CONST a Geode map.*
- `osg::Group * GetGroup (const std::string &name)`  
*Function that is used to request a pointer to a Group Node.*
- `const osg::Group * GetGroup (const std::string &name) const`  
*Function that is used to request a CONST pointer to a Group Node.*
- `NodeCollector::GroupNodeMap & GetGroupNodeMap ()`  
*Function that returns a Group map.*
- `const NodeCollector::GroupNodeMap & GetGroupNodeMap () const`  
*Function that returns a CONST Group map.*
- `osg::LOD * GetLOD (const std::string &name)`  
*Function that is used to request a pointer to an LOD Node.*
- `const osg::LOD * GetLOD (const std::string &name) const`  
*Function that is used to request a CONST pointer to an LOD Node.*
- `NodeCollector::LODNodeMap & GetLODNodeMap ()`  
*Function that returns an LOD map.*
- `const NodeCollector::LODNodeMap & GetLODNodeMap () const`  
*Function that returns CONST an LOD map.*
- `osg::MatrixTransform * GetMatrixTransform (const std::string &name)`  
*Function that is used to request a pointer to a MatrixTransform Node.*
- `const osg::MatrixTransform * GetMatrixTransform (const std::string &name) const`  
*Function that is used to request a CONST pointer to a MatrixTransform Node.*
- `NodeCollector::MatrixTransformNodeMap & GetMatrixTransformNodeMap ()`  
*Function that returns a MatrixTransform map.*
- `const NodeCollector::MatrixTransformNodeMap & GetMatrixTransformNodeMap () const`  
*Function that returns CONST a MatrixTransform map.*
- `osgSim::MultiSwitch * GetMultiSwitch (const std::string &name)`  
*Function that is used to request a pointer to a MultiSwitch Node.*

- const osgSim::MultiSwitch \* **GetMultiSwitch** (const std::string &name) const  
*Function that is used to request a CONST pointer to a MultiSwitch Node.*
- **NodeCollector::MultiSwitchNodeMap & GetMultiSwitchNodeMap** ()  
*Function that returns a MultiSwitch map.*
- const **NodeCollector::MultiSwitchNodeMap & GetMultiSwitchNodeMap** () const  
*Function that returns a CONST MultiSwitch map.*
- osg::Switch \* **GetSwitch** (const std::string &name)  
*Function that is used to request a pointer to a Switch Node.*
- const osg::Switch \* **GetSwitch** (const std::string &name) const  
*Function that is used to request a CONST pointer to a Switch Node.*
- **NodeCollector::SwitchNodeMap & GetSwitchNodeMap** ()  
*Function that returns a Switch map.*
- const **NodeCollector::SwitchNodeMap & GetSwitchNodeMap** () const  
*Function that returns a CONST Switch map.*
- **NodeCollector::TransformNodeMap & GetTransformNodeMap** ()  
*Function that returns a Transform map.*
- const **NodeCollector::TransformNodeMap & GetTransformNodeMap** () const  
*Function that returns a CONST Transform map.*
- void **RemoveDOFTransform** (const std::string &name)  
*Function that is used to remove a DOF Transform Node to the DOF Transform Node map.*
- void **RemoveGeode** (const std::string &name)  
*Function that is used to remove a Geode Node to the Geode Node map.*
- void **RemoveGroup** (const std::string &name)  
*Function that is used to remove a Group Node to the Group Node map.*
- void **RemoveLOD** (const std::string &name)  
*Function that is used to remove an LOD Node to the LOD Node map.*
- void **RemoveMatrixTransform** (const std::string &name)  
*Function that is used to remove a Matrix Transform Node to the Matrix Transform Node map.*
- void **RemoveMultiSwitch** (const std::string &name)  
*Function that is used to remove a MultiSwitch Node to the MultiSwitch Node map.*
- void **RemoveSwitch** (const std::string &name)  
*Function that is used to remove a Switch Node to the Switch Node map.*

### Static Public Attributes

- static const **NodeFlag AllNodeTypes**  
*NodeFlag that when defined will represent all kinds of nodes and Geode nodes.*
- static const **NodeFlag DOFTransformFlag** = dtUtil::Bits::Add(0,2)

- static const **NodeFlag GeodeFlag** = dtUtil::Bits::Add(0,32)
- static const **NodeFlag GroupFlag** = dtUtil::Bits::Add(0,1)  
*NodeFlags that represent the four different types of nodes that can be searched for.*
- static const **NodeFlag LODFlag** = dtUtil::Bits::Add(0,64)
- static const **NodeFlag MatrixTransformFlag** = dtUtil::Bits::Add(0,4)
- static const **NodeFlag MultiSwitchFlag** = dtUtil::Bits::Add(0,16)
- static const **NodeFlag SwitchFlag** = dtUtil::Bits::Add(0,8)

### Protected Member Functions

- virtual **~NodeCollector** ()  
*Destructor.*

#### 6.134.1 Detailed Description

**NodeCollector** (p. 218) is used to gather osg Group nodes, DOFTransform nodes, MatrixTransform nodes, Switch Nodes and Geode nodes (which have Drawable objects and Material objects). It stores the different nodes into corresponding maps which may then be retrieved by the user. In the case of Geode nodes it creates maps for Drawable objects and Material Objects which can be retrieved.

For example, to find the osg::Switch named "switch1":

```
dtUtil::NodeCollector *collector = new dtUtil::NodeCollector(modelNode,
    dtUtil::NodeCollector::SwitchFlag);
osg::Switch* sw = collector->GetSwitch("switch1");
```

#### 6.134.2 Member Typedef Documentation

**6.134.2.1** typedef std::map<std::string, dtCore::RefPtr<osg::Geode> > **GeodeNodeMap**

**6.134.2.2** typedef std::map<std::string, dtCore::RefPtr <osg::Group> > **GroupNodeMap**

**6.134.2.3** typedef std::map<std::string, dtCore::RefPtr <osg::LOD> > **LODNodeMap**

**6.134.2.4** typedef std::map<std::string, dtCore::RefPtr <osg::MatrixTransform> > **MatrixTransformNodeMap**

**6.134.2.5** typedef std::map<std::string, dtCore::RefPtr <osgSim::MultiSwitch> > **MultiSwitchNodeMap**

**6.134.2.6** typedef unsigned **NodeFlag**

Type Definition that is used to declare flags that allow the user to request searches for different types of nodes or geode nodes.

**6.134.2.7** typedef std::map<std::string, dtCore::RefPtr <osg::Switch> > **SwitchNodeMap**

**6.134.2.8** typedef std::map<std::string, dtCore::RefPtr <osgSim::DOFTransform> > **TransformNodeMap**

#### 6.134.3 Constructor & Destructor Documentation

**6.134.3.1** **NodeCollector** ()

Blank Constructor that is defined to do nothing. Note If this is used then you must call use the CollectNodes function in order to generate any maps with this class.

**6.134.3.2** **NodeCollector** (osg::Node \* *nodeToLoad*, **NodeCollector::NodeFlag** *mask*, const std::string & *nodeNameIgnored* = "")

Constructor that when called will automatically generate the node maps or geode maps that you request. Parameters

***nodeToLoad*** The starting node who's children you wish to traverse.

***mask*** The different types of nodes you want to collect off of the loaded node.

***nodeNameIgnored*** The name of a node that you do not want to collect.

### 6.134.3.3 ~NodeCollector () [protected, virtual]

Destructor.

## 6.134.4 Member Function Documentation

### 6.134.4.1 void AddDOFTransform (const std::string & name, osgSim::DOFTransform & node)

Function that is used to add a DOFTransform Node to the DOFTransform Node map. Parameters

**name** A String that represents the name of the Node

**node** The DOFTransform Node that you wish to add to the map

### 6.134.4.2 void AddGeode (const std::string & name, osg::Geode & node)

Function that is used to add a Geode Node to the Geode Node map. Parameters

**name** A String that represents the name of the Node

**node** The Geode Node that you wish to add to the map

### 6.134.4.3 void AddGroup (const std::string & name, osg::Group & node)

Function that is used to add a Group Node to the Group Node map. Parameters

**name** A String that represents the name of the Node

**node** The Group Node that you wish to add to the map

### 6.134.4.4 void AddLOD (const std::string & name, osg::LOD & node)

Function that is used to add an LOD Node to the LOD Node map. Parameters

**name** A String that represents the name of the Node

**node** The LOD Node that you wish to add to the map

### 6.134.4.5 void AddMatrixTransform (const std::string & name, osg::MatrixTransform & node)

Function that is used to add a MatrixTransform Node to the MatrixTransform Node map. Parameters

**name** A String that represents the name of the Node

**node** The MatrixTransform Node that you wish to add to the map

### 6.134.4.6 void AddMultiSwitch (const std::string & name, osgSim::MultiSwitch & node)

Function that is used to add a MultiSwitch Node to the MultiSwitch Node map. Parameters

**name** A String that represents the name of the Node

**node** The MultiSwitch Node that you wish to add to the map

### 6.134.4.7 void AddSwitch (const std::string & name, osg::Switch & node)

Function that is used to add a Switch Node to the Switch Node map. Parameters

**name** A String that represents the name of the Node

**node** The Switch Node that you wish to add to the map

### 6.134.4.8 void ClearAll ()

Function that is defined to clear all the maps of their contents.

**6.134.4.9 void CollectNodes (osg::Node \* *NodeToLoad*, NodeCollector::NodeFlag *mask*, const std::string & *nodeNamesIgnored* = "")**

Function that when called will automatically generate the node maps or geode maps that you request. Parameters

***nodeToLoad*** The starting node who's children you wish to traverse.

***mask*** The different types of nodes you want to collect off of the loaded node.

***nodeNameIgnored*** The name of a node that you do not want to collect.

Note This function was originally intended to be used in conjunction with the blank constructor or after a call of the ClearAllNodes function

**6.134.4.10 osgSim::DOFTransform \* GetDOFTransform (const std::string & *name*)**

Function that is used to request a pointer to a DOFTransform Node. Parameters

***name*** A String that represents the name of the node you are looking for

Returns A pointer to the node you were looking for or NULL if the node was not found

**6.134.4.11 const osgSim::DOFTransform \* GetDOFTransform (const std::string & *name*) const**

Function that is used to request a CONST pointer to a DOFTransform Node. Parameters

***name*** A String that represents the name of the node you are looking for

Returns A CONST pointer to the node you were looking for or NULL if the node was not found

**6.134.4.12 osg::Geode \* GetGeode (const std::string & *name*)**

Function that is used to request a pointer to a Geode Node. Parameters

***name*** A String that represents the name of the node you are looking for

Returns A pointer to the node you were looking for or NULL if the node was not found

**6.134.4.13 const osg::Geode \* GetGeode (const std::string & *name*) const**

Function that is used to request a CONST pointer to a Geode Node. Parameters

***name*** A String that represents the name of the node you are looking for

Returns A CONST pointer to the node you were looking for or NULL if the node was not found

**6.134.4.14 NodeCollector::GeodeNodeMap & GetGeodeNodeMap ()**

Function that returns a Geode map. Returns A map with the names of Geode Nodes and osg Geode Nodes

**6.134.4.15 const NodeCollector::GeodeNodeMap & GetGeodeNodeMap () const**

Function that returns CONST a Geode map. Returns A map with the names of Geode Nodes and osg Geode Nodes

**6.134.4.16 osg::Group \* GetGroup (const std::string & *name*)**

Function that is used to request a pointer to a Group Node. Parameters

***name*** A String that represents the name of the node you are looking for

Returns A pointer to the node you were looking for or NULL if the node was not found

**6.134.4.17 const osg::Group \* GetGroup (const std::string & name) const**

Function that is used to request a CONST pointer to a Group Node. Parameters

**name** A String that represents the name of the node you are looking for

Returns A CONST pointer to the node you were looking for or NULL if the node was not found

**6.134.4.18 NodeCollector::GroupNodeMap & GetGroupNodeMap ()**

Function that returns a Group map. Returns A map with the names of Group Nodes and osg Group Nodes

**6.134.4.19 const NodeCollector::GroupNodeMap & GetGroupNodeMap () const**

Function that returns a CONST Group map. Returns A map with the names of Group Nodes and osg Group Nodes

**6.134.4.20 osg::LOD \* GetLOD (const std::string & name)**

Function that is used to request a pointer to an LOD Node. Parameters

**name** A String that represents the name of the node you are looking for

Returns A pointer to the node you were looking for or NULL if the node was not found

**6.134.4.21 const osg::LOD \* GetLOD (const std::string & name) const**

Function that is used to request a CONST pointer to an LOD Node. Parameters

**name** A String that represents the name of the node you are looking for

Returns A CONST pointer to the node you were looking for or NULL if the node was not found

**6.134.4.22 NodeCollector::LODNodeMap & GetLODNodeMap ()**

Function that returns an LOD map. Returns A map with the names of LOD Nodes and osg LOD Nodes

**6.134.4.23 const NodeCollector::LODNodeMap & GetLODNodeMap () const**

Function that returns CONST an LOD map. Returns A map with the names of Geode LOD and osg LOD Nodes

**6.134.4.24 osg::MatrixTransform \* GetMatrixTransform (const std::string & name)**

Function that is used to request a pointer to a MatrixTransform Node. Parameters

**name** A String that represents the name of the node you are looking for

Returns A pointer to the node you were looking for or NULL if the node was not found

**6.134.4.25 const osg::MatrixTransform \* GetMatrixTransform (const std::string & name) const**

Function that is used to request a CONST pointer to a MatrixTransform Node. Parameters

**name** A String that represents the name of the node you are looking for

Returns A CONST pointer to the node you were looking for or NULL if the node was not found

**6.134.4.26 NodeCollector::MatrixTransformNodeMap & GetMatrixTransformNodeMap ()**

Function that returns a MatrixTransform map. Returns A map with the names of MatrixTransform Nodes and osg MatrixTransform Nodes

**6.134.4.27 const NodeCollector::MatrixTransformNodeMap & GetMatrixTransformNodeMap () const**

Function that returns CONST a MatrixTransform map. Returns A map with the names of MatrixTransform Nodes and osg MatrixTransform Nodes

**6.134.4.28** `osgSim::MultiSwitch * GetMultiSwitch (const std::string & name)`

Function that is used to request a pointer to a MultiSwitch Node. Parameters

**name** A String that represents the name of the node you are looking for

Returns A pointer to the node you were looking for or NULL if the node was not found

**6.134.4.29** `const osgSim::MultiSwitch * GetMultiSwitch (const std::string & name) const`

Function that is used to request a CONST pointer to a MultiSwitch Node. Parameters

**name** A String that represents the name of the node you are looking for

Returns A CONST pointer to the node you were looking for or NULL if the node was not found

**6.134.4.30** `NodeCollector::MultiSwitchNodeMap & GetMultiSwitchNodeMap ()`

Function that returns a MultiSwitch map. Returns A map with the names of MultiSwitch Nodes and osg MultiSwitch Nodes

**6.134.4.31** `const NodeCollector::MultiSwitchNodeMap & GetMultiSwitchNodeMap () const`

Function that returns a CONST MultiSwitch map. Returns A map with the names of MultiSwitch Nodes and osg MultiSwitch Nodes

**6.134.4.32** `osg::Switch * GetSwitch (const std::string & name)`

Function that is used to request a pointer to a Switch Node. Parameters

**name** A String that represents the name of the node you are looking for

Returns A pointer to the node you were looking for or NULL if the node was not found

**6.134.4.33** `const osg::Switch * GetSwitch (const std::string & name) const`

Function that is used to request a CONST pointer to a Switch Node. Parameters

**name** A String that represents the name of the node you are looking for

Returns A CONST pointer to the node you were looking for or NULL if the node was not found

**6.134.4.34** `NodeCollector::SwitchNodeMap & GetSwitchNodeMap ()`

Function that returns a Switch map. Returns A map with the names of Switch Nodes and osg Switch Nodes

**6.134.4.35** `const NodeCollector::SwitchNodeMap & GetSwitchNodeMap () const`

Function that returns a CONST Switch map. Returns A map with the names of Switch Nodes and osg Switch Nodes

**6.134.4.36** `NodeCollector::TransformNodeMap & GetTransformNodeMap ()`

Function that returns a Transform map. Returns A map with the names of Transform Nodes and osg Transform Nodes

**6.134.4.37** `const NodeCollector::TransformNodeMap & GetTransformNodeMap () const`

Function that returns a CONST Transform map. Returns A map with the names of Transform Nodes and osg Transform Nodes

**6.134.4.38** `void RemoveDOFTransform (const std::string & name)`

Function that is used to remove a DOF Transform Node to the DOF Transform Node map. Parameters

**name** A String that represents the name of the Node

**6.134.4.39 void RemoveGeode (const std::string & name)**

Function that is used to remove a Geode Node to the Geode Node map. Parameters

**name** A String that represents the name of the Node

**6.134.4.40 void RemoveGroup (const std::string & name)**

Function that is used to remove a Group Node to the Group Node map. Parameters

**name** A String that represents the name of the Node

**6.134.4.41 void RemoveLOD (const std::string & name)**

Function that is used to remove an LOD Node to the LOD Node map. Parameters

**name** A String that represents the name of the Node

**6.134.4.42 void RemoveMatrixTransform (const std::string & name)**

Function that is used to remove a Matrix Transform Node to the Matrix Transform Node map. Parameters

**name** A String that represents the name of the Node

**6.134.4.43 void RemoveMultiSwitch (const std::string & name)**

Function that is used to remove a MultiSwitch Node to the MultiSwitch Node map. Parameters

**name** A String that represents the name of the Node

**6.134.4.44 void RemoveSwitch (const std::string & name)**

Function that is used to remove a Switch Node to the Switch Node map. Parameters

**name** A String that represents the name of the Node

**6.134.5 Member Data Documentation****6.134.5.1 const NodeCollector::NodeFlag AllNodeTypes [static]**

Initial value:

```
(NodeCollector::GroupFlag |
  NodeCollector::DOFTransformFlag |
  NodeCollector::MatrixTransformFlag |
  NodeCollector::SwitchFlag |
  NodeCollector::MultiSwitchFlag |
  NodeCollector::GeodeFlag |
  NodeCollector::LODFlag)
```

NodeFlag that when defined will represent all kinds of nodes and Geode nodes.

**6.134.5.2 const NodeCollector::NodeFlag DOFTransformFlag = dtUtil::Bits::Add(0,2) [static]****6.134.5.3 const NodeCollector::NodeFlag GeodeFlag = dtUtil::Bits::Add(0,32) [static]****6.134.5.4 const NodeCollector::NodeFlag GroupFlag = dtUtil::Bits::Add(0,1) [static]**

NodeFlags that represent the four different types of nodes that can be searched for.

6.134.5.5 `const NodeCollector::NodeFlag LODFlag = dtUtil::Bits::Add(0,64) [static]`

6.134.5.6 `const NodeCollector::NodeFlag MatrixTransformFlag = dtUtil::Bits::Add(0,4) [static]`

6.134.5.7 `const NodeCollector::NodeFlag MultiSwitchFlag = dtUtil::Bits::Add(0,16) [static]`

6.134.5.8 `const NodeCollector::NodeFlag SwitchFlag = dtUtil::Bits::Add(0,8) [static]`

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

- `nodecollector.h`
- `nodecollector.cpp`

## 6.135 NodePrintOut Class Reference

Utility class used to traverse a node and generate a formatted text output.

```
#include <inc/dtUtil/nodeprintout.h>
```

### Public Member Functions

- **NodePrintOut** ()
- std::string **CollectNodeData** (const osg::Node &nodeToPrint, const std::string &outputFilename="", bool printVertData=false, unsigned int nodeMask=0xFFFFFFFF)
 

*Traverse a node's graph and generate a text printout of the hierarchy.*
- std::string **GetFileOutput** () const
 

*Returns the file stream.*
- void **PrintNodeToOSGFile** (const osg::Node &node, std::ostream &oss)
 

*Dumps a node to a stream in osg format.*
- void **PrintNodeToOSGFile** (const osg::Node &node, const std::string &fileName)
 

*Dumps a node to an osg file.*

### Protected Member Functions

- virtual ~**NodePrintOut** ()
- void **Analyze** (const osg::Node &nd, const std::string &indent, unsigned int nodeMask)
 

*Called from printoutnode user should never call.*
- void **AnalyzeGeode** (const osg::Geode &geode, const std::string &indent)
 

*Called from Analyze user should never call.*
- void **AnalyzePrimSet** (const osg::PrimitiveSet &prset, const osg::Vec3Array &verts, const std::string &indent)
 

*Called from AnalyzeGeode user should never call.*

#### 6.135.1 Detailed Description

Utility class used to traverse a node and generate a formatted text output. Example:

```
osg::Node *node = LoadFile("myFile.ive");
RefPtr<NodePrintOut> printout = new NodePrintOut();
std::cout << printout->CollectNodeData(*node) << std::endl;
```

#### 6.135.2 Constructor & Destructor Documentation

##### 6.135.2.1 NodePrintOut ()

##### 6.135.2.2 virtual ~NodePrintOut () [inline, protected, virtual]

#### 6.135.3 Member Function Documentation

##### 6.135.3.1 void Analyze (const osg::Node & nd, const std::string & indent, unsigned int nodeMask) [protected]

Called from printoutnode user should never call. Called from printoutnode user should never call  
 //

##### 6.135.3.2 void AnalyzeGeode (const osg::Geode & geode, const std::string & indent) [protected]

Called from Analyze user should never call.

**6.135.3.3 void AnalyzePrimSet (const osg::PrimitiveSet & *prset*, const osg::Vec3Array & *verts*, const std::string & *indent*) [protected]**

Called from AnalyzeGeode user should never call.

**6.135.3.4 std::string CollectNodeData (const osg::Node & *nodeToPrint*, const std::string & *outputFilename* = "", bool *printVertData* = false, unsigned int *nodeMask* = 0xFFFFFFFF)**

Traverse a node's graph and generate a text printout of the hierarchy. Parameters

***nodeToPrint*** : the node to traverse

***outputFilename*** : an optional filename to save the output to (default = "")

***printVertData*** : optionally print out vertex information (default = false)

***nodeMask*** : will only print out nodes that have a node mask that share at least one bit.

Returns The formatted string output.

**6.135.3.5 std::string GetFileOutput () const**

Returns the file stream.

**6.135.3.6 void PrintNodeToOSGFile (const osg::Node & *node*, std::ostream & *oss*)**

Dumps a node to a stream in osg formate.

**6.135.3.7 void PrintNodeToOSGFile (const osg::Node & *node*, const std::string & *fileName*)**

Dumps a node to an osg file.

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

- **nodeprintout.h**
- **nodeprintout.cpp**

## 6.136 Noise1< Real, Vector > Class Template Reference

An implementation of 1D Gradient Noise.

```
#include <inc/dtUtil/noise1.h>
```

### Public Member Functions

- **Noise1** (unsigned int seed=1023058)
- **~Noise1** ()
- Real **GetNoise** (const Vector vect\_in)
- void **Reseed** (unsigned int seed)

### 6.136.1 Detailed Description

```
template<class Real, class Vector> class dtUtil::Noise1< Real, Vector >
```

An implementation of 1D Gradient Noise.

### 6.136.2 Constructor & Destructor Documentation

6.136.2.1 **Noise1** (unsigned int *seed* = 1023058) [inline]

6.136.2.2 **~Noise1** () [inline]

### 6.136.3 Member Function Documentation

6.136.3.1 **Real GetNoise** (const Vector *vect\_in*) [inline]

6.136.3.2 **void Reseed** (unsigned int *seed*) [inline]

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

- **noise1.h**

## 6.137 Noise2< Real, Vector > Class Template Reference

An implementation of 2D Gradient Noise.

```
#include <inc/dtUtil/noise2.h>
```

### Public Member Functions

- **Noise2** (unsigned int seed=1023058)
- **~Noise2** ()
- Real **GetNoise** (const Vector &vect\_in)
- void **Reseed** (unsigned int seed)

### 6.137.1 Detailed Description

```
template<class Real, class Vector> class dtUtil::Noise2< Real, Vector >
```

An implementation of 2D Gradient Noise.

### 6.137.2 Constructor & Destructor Documentation

6.137.2.1 **Noise2** (unsigned int *seed* = 1023058) [inline]

6.137.2.2 **~Noise2** () [inline]

### 6.137.3 Member Function Documentation

6.137.3.1 **Real GetNoise** (const Vector & *vect\_in*) [inline]

6.137.3.2 **void Reseed** (unsigned int *seed*) [inline]

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

- **noise2.h**

## 6.138 Noise3< Real, Vector > Class Template Reference

An implementation of 3D Gradient Noise.

```
#include <inc/dtUtil/noise3.h>
```

### Public Member Functions

- **Noise3** (unsigned int seed=1023058)
- **~Noise3** ()
- Real **GetNoise** (const Vector &vect\_in)
- void **Reseed** (unsigned int seed)

### 6.138.1 Detailed Description

```
template<class Real, class Vector> class dtUtil::Noise3< Real, Vector >
```

An implementation of 3D Gradient Noise.

### 6.138.2 Constructor & Destructor Documentation

6.138.2.1 **Noise3** (unsigned int *seed* = 1023058) [inline]

6.138.2.2 **~Noise3** () [inline]

### 6.138.3 Member Function Documentation

6.138.3.1 Real **GetNoise** (const Vector & *vect\_in*) [inline]

6.138.3.2 void **Reseed** (unsigned int *seed*) [inline]

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

- **noise3.h**

## 6.139 NoiseTexture Class Reference

Noise Texture is a class that uses **SeamlessNoise** (p. 251) to generate an osg::Image.

```
#include <inc/dtUtil/noisetexture.h>
```

### Public Member Functions

- DT\_UTIL\_EXPORT **NoiseTexture** (int octaves, int frequency, double amp, double persistence, int width, int height, int slices=1)  
*Constructor: For a more detailed listing of these params see dtUtil::Fracal.*
- DT\_UTIL\_EXPORT **NoiseTexture** ()
- DT\_UTIL\_EXPORT **~NoiseTexture** ()
- osg::Image \* **GetNoiseTexture** ()  
*If you need a pointer to the texture you'll find it here don't call this if you haven't generated the texture yet!*
- DT\_UTIL\_EXPORT osg::Image \* **MakeNoiseTexture** (GLenum format)  
*This function creates the texture.*
- void **SetAmplitude** (double a)  
*Sets amplitude.*
- void **SetFrequency** (int f)  
*Sets frequency.*
- void **SetHeight** (int h)  
*Sets height.*
- void **SetOctaves** (int o)  
*Sets octaves.*
- void **SetPersistence** (double p)  
*sets persistence*
- void **SetSlices** (int s)  
*Sets slices.*
- void **SetWidth** (int w)  
*Sets width.*

#### 6.139.1 Detailed Description

Noise Texture is a class that uses **SeamlessNoise** (p. 251) to generate an osg::Image.

#### 6.139.2 Constructor & Destructor Documentation

##### 6.139.2.1 NoiseTexture ()

##### 6.139.2.2 NoiseTexture (int *octaves*, int *frequency*, double *amp*, double *persistence*, int *width*, int *height*, int *slices* = 1)

Constructor: For a more detailed listing of these params see dtUtil::Fracal. Parameters

**octaves** the number of summations of the noise

**frequency** the frequency of the noise

**amp** the amplitude of the noise

***persistence*** the persistence of the noise

***width*** A power of 2, specifying the x resolution

***height*** A power of 2, specifying the y resolution

***slices*** A power of 2, specifying the z resolution, or 1 for 2D textures

See also **dtUtil::Fractal** (p. 128)

### 6.139.2.3 ~NoiseTexture ()

## 6.139.3 Member Function Documentation

### 6.139.3.1 osg::Image\* GetNoiseTexture () [inline]

If you need a pointer to the texture you'll find it here don't call this if you haven't generated the texture yet! Returns returns a pointer to the image created

### 6.139.3.2 osg::Image \* MakeNoiseTexture (GLenum *format*)

This function creates the texture. Parameters

***format*** specifies the format of the texture should be used GL\_ALPHA (for a transparency map), GL\_LUMINANCE, GL\_RGB or GL\_RGBA

Returns returns a pointer to the image

### 6.139.3.3 void SetAmplitude (double *a*) [inline]

Sets amplitude. Parameters

***a*** a double greater than 0

### 6.139.3.4 void SetFrequency (int *f*) [inline]

Sets frequency. Parameters

***f*** an int greater than 0

### 6.139.3.5 void SetHeight (int *h*) [inline]

Sets height. Parameters

***h*** texture resolution of y dimension

### 6.139.3.6 void SetOctaves (int *o*) [inline]

Sets octaves. Parameters

***o*** an int greater than 0

### 6.139.3.7 void SetPersistence (double *p*) [inline]

sets persistence Parameters

***p*** a double from 0-1

### 6.139.3.8 void SetSlices (int *s*) [inline]

Sets slices. Parameters

***s*** the number of slices

**6.139.3.9 void SetWidth (int *w*) [inline]**

Sets width. Parameters

***w*** texture resolution of x dimension

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

- **noisetexture.h**
- **noisetexture.cpp**

## 6.140 NoLeakAlloc Class Reference

```
#include <inc/dtUtil/kdtree.h>
```

### Public Member Functions

- **NoLeakAlloc** (**\_Alloc\_base** \*b)
- **~NoLeakAlloc** ()
- void **disconnect** ()
- **\_Node\_\*** **get** ()

```
template<typename _Tp, typename _Alloc> class dtUtil::_Alloc_base< _Tp, _Alloc >::NoLeakAlloc
```

### 6.140.1 Constructor & Destructor Documentation

6.140.1.1 **NoLeakAlloc** (**\_Alloc\_base** \* *b*) [**inline**]

6.140.1.2 **~NoLeakAlloc** () [**inline**]

### 6.140.2 Member Function Documentation

6.140.2.1 void **disconnect** () [**inline**]

6.140.2.2 **\_Node\_\*** **get** () [**inline**]

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

- **kdtree.h**

## 6.141 NotIntType< T, i > Struct Template Reference

```
#include <inc/dtUtil/generic.h>
```

### Public Types

- enum { **value** = !IsIntType<T, **value** = !IsIntType<T }

```
template<class T, int i> struct dtUtil::NotIntType< T, i >
```

### 6.141.1 Member Enumeration Documentation

#### 6.141.1.1 anonymous enum

Enumerator:

*value*

*value*

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

- **generic.h**

## 6.142 NullType Class Reference

```
#include <inc/dtUtil/generic.h>
```

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

- **generic.h**

## 6.143 `ObjectFactory< UniqueldTypeClass, BaseTypeClass, ItCmpClass >` Class Template Reference

This class is a template object factory.

```
#include <inc/dtUtil/objectfactory.h>
```

### Public Types

- typedef BaseTypeClass **BaseType**
- typedef **BaseType** \*(\* **createObjectFunc** )()
- typedef ItCmpClass **ItCmp**
- typedef std::map< **UniqueldType**, **createObjectFunc**, **ItCmp** > **ObjectMap**  
*Function pointer type for functions creating objects.*
- typedef ObjectMap::iterator **ObjTypeltor**
- typedef ObjectMap::const\_iterator **ObjTypeltorConst**
- typedef UniqueldTypeClass **UniqueldType**

### Public Member Functions

- **ObjectFactory** ()
- **BaseType** \* **CreateObject** (const **UniqueldType** id) const  
*Creates a new object.*
- const **ObjectMap** & **GetMap** () const
- void **GetSupportedTypes** (std::vector< **UniqueldType** > &types) const  
*Gets a list of types that this factory knows how to create.*
- bool **IsTypeSupported** (**UniqueldType** id) const  
*Checks to see if the factory can create objects of the given type.*
- template<typename DerivedType >  
bool **RegisterType** (**UniqueldType** id)  
*Registers a new type of object with the factory.*
- void **RemoveType** (**UniqueldType** id)  
*Removes an existing object type from the factory's known list of object types.*

### Protected Member Functions

- virtual ~**ObjectFactory** ()

#### 6.143.1 Detailed Description

```
template<typename UniqueldTypeClass, typename BaseTypeClass, typename ItCmpClass =
std::less<UniqueldTypeClass>> class dtUtil::ObjectFactory< UniqueldTypeClass, BaseTypeClass,
ItCmpClass >
```

This class is a template object factory. It allows one to create any type of object as long as there is a common base class. The common base class is defined on a per-factory basis using the templated parameter `BaseType`. Note The **ObjectFactory** (p. 239) implementation only supports objects with a default constructor. It will not work with objects that only have named constructors.

## 6.143.2 Member Typedef Documentation

6.143.2.1 `typedef BaseTypeClass BaseType`

6.143.2.2 `typedef BaseType*(* createObjectFunc)()`

6.143.2.3 `typedef ItCmpClass ItCmp`

6.143.2.4 `typedef std::map<UniqueldType,createObjectFunc,ItCmp> ObjectMap`

Function pointer type for functions creating objects.

6.143.2.5 `typedef ObjectMap::iterator ObjTypeltor`

6.143.2.6 `typedef ObjectMap::const_iterator ObjTypeltorConst`

6.143.2.7 `typedef UniqueldTypeClass UniqueldType`

## 6.143.3 Constructor & Destructor Documentation

6.143.3.1 `ObjectFactory () [inline]`

6.143.3.2 `virtual ~ObjectFactory () [inline, protected, virtual]`

## 6.143.4 Member Function Documentation

6.143.4.1 `BaseType* CreateObject (const UniqueldType id) const [inline]`

Creates a new object. Parameters

*id* - Type of object to create.

Returns Returns a pointer to the newly created object or NULL if the given id has not been registered.

Exceptions

**Exception** (p. 116) is thrown if the factory does not know how to create the requested type.

6.143.4.2 `const ObjectMap& GetMap () const [inline]`

6.143.4.3 `void GetSupportedTypes (std::vector< UniqueldType > & types) const [inline]`

Gets a list of types that this factory knows how to create.

6.143.4.4 `bool IsTypeSupported (UniqueldType id) const [inline]`

Checks to see if the factory can create objects of the given type. Parameters

*id* The type of object to check for.

Returns True if the type is supported, false otherwise.

6.143.4.5 `bool RegisterType (UniqueldType id) [inline]`

Registers a new type of object with the factory. Returns false if the type is a duplicate.

6.143.4.6 `void RemoveType (UniqueldType id) [inline]`

Removes an existing object type from the factory's known list of object types.

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

- `objectfactory.h`

## 6.144 Packager Class Reference

The **Packager** (p. 241) is used to package multiple files into a single .dtpkg file.

```
#include <inc/dtUtil/packager.h>
```

### Classes

- struct **PackTreeData**

### Public Member Functions

- **Packager** ()  
*Default constructor.*
- **~Packager** ()  
*Default destructor.*
- bool **AddFile** (const std::string &filepath, const std::string &outDir)  
*Adds a specified file to the package.*
- bool **ClosePackage** ()  
*Closes the currently opened package.*
- **PackTreeData \* FindPackDataForPath** (const std::string &path)  
*This will parse the **PackTreeData** (p. 243) structure and find the node to the given path.*
- const **PackTreeData & GetPackTree** ()  
*Retrieves the file structure tree of all files that are currently bound to be packed into a package.*
- bool **OpenPackage** (const std::string &filename)  
*Opens an existing package file.*
- bool **PackPackage** (const std::string &filename, bool overwrite=true)  
*Packs the currently added files to a package.*
- bool **RemoveFile** (const std::string &filepath)  
*Removes a file that is currently packaged.*
- bool **UnpackPackage** (const std::string &filename, const std::string &outDir)  
*Unpacks a package to a specified location.*

### 6.144.1 Detailed Description

The **Packager** (p. 241) is used to package multiple files into a single .dtpkg file. Using the packager will allow you to view, import, and extract to and from the contents of a package file.

### 6.144.2 Constructor & Destructor Documentation

#### 6.144.2.1 Packager ()

Default constructor.

#### 6.144.2.2 ~Packager ()

Default destructor.

### 6.144.3 Member Function Documentation

#### 6.144.3.1 **bool AddFile (const std::string & *filepath*, const std::string & *outDir*)**

Adds a specified file to the package. Parameters

← ***filepath*** The path of the file to add.

← ***outDir*** The output directory when this file is unpacked. (this is relative to the extraction directory)

#### 6.144.3.2 **bool ClosePackage ()**

Closes the currently opened package. Returns Fails if no package was opened.

#### 6.144.3.3 **Packager::PackTreeData \* FindPackDataForPath (const std::string & *path*)**

This will parse the **PackTreeData** (p. 243) structure and find the node to the given path. If none exists, this will return NULL.

#### 6.144.3.4 **const PackTreeData& GetPackTree () [inline]**

Retrieves the file structure tree of all files that are currently bound to be packed into a package.

#### 6.144.3.5 **bool OpenPackage (const std::string & *filename*)**

Opens an existing package file. You don't need this if you are creating a new package.

Parameters

← ***filename*** The name of the file to open.

Returns This will return false if the package was not found.

#### 6.144.3.6 **bool PackPackage (const std::string & *filename*, bool *overwrite* = true)**

Packs the currently added files to a package. Parameters

← ***filename*** The name of the package file (exclude the ext)

← ***overwrite*** True to overwrite any existing package files.

Returns Returns false if the package was not saved.

#### 6.144.3.7 **bool RemoveFile (const std::string & *filepath*)**

Removes a file that is currently packaged. Parameters

← ***filepath*** The path of the file to remove. Note: this is not the path of the source file, it is the path you used when adding the file to this package.

#### 6.144.3.8 **bool UnpackPackage (const std::string & *filename*, const std::string & *outDir*)**

Unpacks a package to a specified location. Parameters

← ***filename*** The name of the pack file to unpack (don't include extension).

← ***outDir*** The output directory to unpack to.

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

- **packager.h**
- **packager.cpp**

## 6.145 PackTreeData Struct Reference

```
#include <inc/dtUtil/packager.h>
```

### Public Attributes

- `std::vector< PackTreeData > files`
- `std::vector< PackTreeData > folders`
- `bool isFromPack`
- `std::string name`
- `PackTreeData * parent`
- `fpos_t seekPos`
- `std::string source`

### 6.145.1 Member Data Documentation

6.145.1.1 `std::vector<PackTreeData> files`

6.145.1.2 `std::vector<PackTreeData> folders`

6.145.1.3 `bool isFromPack`

6.145.1.4 `std::string name`

6.145.1.5 `PackTreeData* parent`

6.145.1.6 `fpos_t seekPos`

6.145.1.7 `std::string source`

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

- `packager.h`

## 6.146 PolarDecomp Class Reference

**PolarDecomp** (p. 244) is a class that will take a 4x4 Matrix and break it up into the following components Rotation, Scale, and Translation.

```
#include <inc/dtUtil/polardecomp.h>
```

### Static Public Member Functions

- static float **Decompose** (const osg::Matrix &M, osg::Matrix &Q, osg::Matrix &S, osg::Vec3 &T)  
*Find Polar Decomposition of Matrix M: Q=Rotation, S=Scale/Stretch, T=Translation.*

#### 6.146.1 Detailed Description

**PolarDecomp** (p. 244) is a class that will take a 4x4 Matrix and break it up into the following components Rotation, Scale, and Translation.

#### 6.146.2 Member Function Documentation

##### 6.146.2.1 float Decompose (const osg::Matrix & M, osg::Matrix & Q, osg::Matrix & S, osg::Vec3 & T) [static]

Find Polar Decomposition of Matrix M: Q=Rotation, S=Scale/Stretch, T=Translation.

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

- polardecomp.h
- polardecomp.cpp

## 6.147 Predicate< j, IdsTL2 > Struct Template Reference

```
#include <inc/dtUtil/funbind.h>
```

### Public Types

- enum { **value** = IdsTL2::Head::value == j }

```
template<int i, class BoundPTL, class UnboundPTL, class IdsTL, class TL>template<int j, class IdsTL2>  
struct dtUtil::MergeParmsH< i, BoundPTL, UnboundPTL, IdsTL, TL >::Predicate< j, IdsTL2 >
```

### 6.147.1 Member Enumeration Documentation

#### 6.147.1.1 anonymous enum

Enumerator:

*value*

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

- **funbind.h**

## 6.148 Predicate< j, dtUtil::NullType > Struct Template Reference

```
#include <inc/dtUtil/funbind.h>
```

### Public Types

- enum { **value** = false }

```
template<int i, class BoundPTL, class UnboundPTL, class IdsTL, class TL>template<int j> struct  
dtUtil::MergeParmsH< i, BoundPTL, UnboundPTL, IdsTL, TL >::Predicate< j, dtUtil::NullType >
```

### 6.148.1 Member Enumeration Documentation

#### 6.148.1.1 anonymous enum

Enumerator:

*value*

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

- **funbind.h**

## 6.149 RefString Class Reference

A string wrapper that will "intern" all of the strings so that strings with the same value will point to the same memory.

```
#include <inc/dtUtil/refstring.h>
```

### Public Member Functions

- **RefString** (const **RefString** &toCopy)
- **RefString** (const char \*value)
- **RefString** (const std::string &value="")
- **~RefString** ()
- const char \* **c\_str** () const
- const std::string & **Get** () const
- **operator const std::string &** () const
- bool **operator!=** (const std::string &toCompare) const
- bool **operator!=** (const char \*toCompare) const
- bool **operator!=** (const **dtUtil::RefString** &toCompare) const
- **RefString operator+** (const char \*str) const
- **RefString operator+** (const **RefString** &refString) const
- **RefString operator+** (const std::string &string) const
- const std::string \* **operator->** () const
- bool **operator<** (const **dtUtil::RefString** &toCompare) const
- **dtUtil::RefString & operator=** (const **dtUtil::RefString** &value)
- **dtUtil::RefString & operator=** (const std::string &value)
- bool **operator==** (const char \*toCompare) const
- bool **operator==** (const std::string &toCompare) const
- bool **operator==** (const **dtUtil::RefString** &toCompare) const
- std::string::value\_type **operator[]** (int index) const

### Static Public Member Functions

- static size\_t **GetSharedStringCount** ()

#### 6.149.1 Detailed Description

A string wrapper that will "intern" all of the strings so that strings with the same value will point to the same memory. The strings are always only accessible as const, but a new string may be assigned to the refstring

#### 6.149.2 Constructor & Destructor Documentation

**6.149.2.1** **RefString** (const std::string & *value* = "")

**6.149.2.2** **RefString** (const char \* *value*)

**6.149.2.3** **RefString** (const **RefString** & *toCopy*)

**6.149.2.4** **~RefString** ()

#### 6.149.3 Member Function Documentation

**6.149.3.1** const char\* **c\_str** () const [inline]

**6.149.3.2** const std::string& **Get** () const [inline]

**6.149.3.3** size\_t **GetSharedStringCount** () [static]

Returns the number of shared strings.

- 6.149.3.4 `operator const std::string & () const` [inline]
- 6.149.3.5 `bool operator!= (const std::string & toCompare) const` [inline]
- 6.149.3.6 `bool operator!= (const char * toCompare) const` [inline]
- 6.149.3.7 `bool operator!= (const dtUtil::RefString & toCompare) const` [inline]
- 6.149.3.8 `RefString operator+ (const char * str) const`
- 6.149.3.9 `RefString operator+ (const RefString & refString) const`
- 6.149.3.10 `RefString operator+ (const std::string & string) const`
- 6.149.3.11 `const std::string* operator-> () const` [inline]
- 6.149.3.12 `bool operator< (const dtUtil::RefString & toCompare) const` [inline]
- 6.149.3.13 `dtUtil::RefString & operator= (const dtUtil::RefString & value)`
- 6.149.3.14 `dtUtil::RefString & operator= (const std::string & value)`
- 6.149.3.15 `bool operator== (const char * toCompare) const` [inline]
- 6.149.3.16 `bool operator== (const std::string & toCompare) const` [inline]
- 6.149.3.17 `bool operator== (const dtUtil::RefString & toCompare) const` [inline]
- 6.149.3.18 `std::string::value_type operator[] (int index) const` [inline]

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

- `refstring.h`
- `refstring.cpp`

## 6.150 ResourceLoader< ResourceDescriptor, Resource > Class Template Reference

```
#include <inc/dtUtil/resourceloader.h>
```

### Public Member Functions

- virtual void **FreeResource** (Resource \*)=0
- virtual Resource \* **LoadResource** (const ResourceDescriptor &)=0

### Protected Member Functions

- virtual ~**ResourceLoader** ()

```
template<class ResourceDescriptor, class Resource> class dtUtil::ResourceLoader< ResourceDescriptor, Resource >
```

#### 6.150.1 Constructor & Destructor Documentation

6.150.1.1 virtual ~ResourceLoader () [inline, protected, virtual]

#### 6.150.2 Member Function Documentation

6.150.2.1 virtual void FreeResource (Resource \*) [pure virtual]

6.150.2.2 virtual Resource\* LoadResource (const ResourceDescriptor &) [pure virtual]

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

- resourceloader.h

## 6.151 ResourceManager< ResourceKey, Resource > Class Template Reference

```
#include <inc/dtUtil/resourcemanager.h>
```

### Public Types

- typedef ResourceMap::iterator **ResourceConstIterator**
- typedef std::pair< ResourceKey, dtCore::RefPtr< Resource > > **ResourceHandle**
- typedef ResourceMap::iterator **ResourceIterator**
- typedef std::map< ResourceKey, dtCore::RefPtr< Resource > > **ResourceMap**

### Public Member Functions

- **ResourceManager** ()
- virtual void **AddResource** (const ResourceKey &pHandle, Resource \*pResource)
- void **FreeAll** ()
- virtual void **FreeResource** (const ResourceKey &pHandle)
- const Resource \* **GetResource** (const ResourceKey &pHandle) const
- Resource \* **GetResource** (const ResourceKey &pHandle)
- virtual bool **LoadResource** (const ResourceKey &pHandle, const std::string &pFilename)
- void **SetResourceLoader** (**ResourceLoader**< ResourceKey, Resource > \*pLoader)

### Protected Member Functions

- virtual ~**ResourceManager** ()

```
template<class ResourceKey, class Resource> class dtUtil::ResourceManager< ResourceKey, Resource >
```

#### 6.151.1 Member Typedef Documentation

- 6.151.1.1 typedef ResourceMap::iterator **ResourceConstIterator**
- 6.151.1.2 typedef std::pair<ResourceKey, dtCore::RefPtr<Resource> > **ResourceHandle**
- 6.151.1.3 typedef ResourceMap::iterator **ResourceIterator**
- 6.151.1.4 typedef std::map<ResourceKey, dtCore::RefPtr<Resource> > **ResourceMap**

#### 6.151.2 Constructor & Destructor Documentation

- 6.151.2.1 **ResourceManager** () [inline]
- 6.151.2.2 virtual ~**ResourceManager** () [inline, protected, virtual]

#### 6.151.3 Member Function Documentation

- 6.151.3.1 virtual void **AddResource** (const ResourceKey & *pHandle*, Resource \* *pResource*) [inline, virtual]
- 6.151.3.2 void **FreeAll** () [inline]
- 6.151.3.3 virtual void **FreeResource** (const ResourceKey & *pHandle*) [inline, virtual]
- 6.151.3.4 const Resource\* **GetResource** (const ResourceKey & *pHandle*) const [inline]
- 6.151.3.5 Resource\* **GetResource** (const ResourceKey & *pHandle*) [inline]
- 6.151.3.6 virtual bool **LoadResource** (const ResourceKey & *pHandle*, const std::string & *pFilename*) [inline, virtual]
- 6.151.3.7 void **SetResourceLoader** (**ResourceLoader**< ResourceKey, Resource > \* *pLoader*) [inline]

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

- **resourcemanager.h**

## 6.152 SeamlessNoise Class Reference

**SeamlessNoise** (p. 251) is a noise class that creates tileable noise.

```
#include <inc/dtUtil/seamlessnoise.h>
```

### Public Member Functions

- **SeamlessNoise** (unsigned int seed=1023058)
- **~SeamlessNoise** ()
- float **GetNoise** (const osg::Vec3f &vect\_in, int repeat=-1)
- void **Reseed** (unsigned int seed)
  - Regenerates the random numbers using the given seed.*
- void **SetRepeat** (int pRepeat)
  - The SetRepeat function will allow a user to change the frequency the noise function will repeat.*

### 6.152.1 Detailed Description

**SeamlessNoise** (p. 251) is a noise class that creates tileable noise.

### 6.152.2 Constructor & Destructor Documentation

**6.152.2.1 SeamlessNoise (unsigned int seed = 1023058)**

**6.152.2.2 ~SeamlessNoise ()**

### 6.152.3 Member Function Documentation

**6.152.3.1 float GetNoise (const osg::Vec3f & vect\_in, int repeat = -1)**

Parameters

**vect\_in** a 3D Vector to hash the noise to, for 2D noise pass in a constant z value

**repeat** pass in the frequency or the desired resolution to tile the noise in if this param is not passed in the default value will be used

Returns a float from -1 to 1

**6.152.3.2 void Reseed (unsigned int seed)**

Regenerates the random numbers using the given seed. Parameters

**seed** new seed value

**6.152.3.3 void SetRepeat (int pRepeat) [inline]**

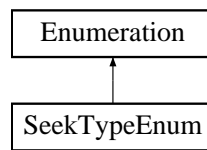
The SetRepeat function will allow a user to change the frequency the noise function will repeat.

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

- **seamlessnoise.h**
- **seamlessnoise.cpp**

## 6.153 SeekTypeEnum Class Reference

#include <inc/dtUtil/datastream.h> Inheritance diagram for SeekTypeEnum::



### Static Public Attributes

- static const **SeekTypeEnum** CURRENT
- static const **SeekTypeEnum** END
- static const **SeekTypeEnum** SET

### 6.153.1 Member Data Documentation

6.153.1.1 **const** **DataStream::SeekTypeEnum** CURRENT [static]

6.153.1.2 **const** **DataStream::SeekTypeEnum** END [static]

6.153.1.3 **const** **DataStream::SeekTypeEnum** SET [static]

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

- **datastream.h**
- **datastream.cpp**

## 6.154 `Select< flag, T, U >` Struct Template Reference

```
#include <inc/dtUtil/generic.h>
```

### Public Types

- `typedef T Result`

```
template<bool flag, typename T, typename U> struct dtUtil::Select< flag, T, U >
```

### 6.154.1 Member Typedef Documentation

#### 6.154.1.1 `typedef T Result`

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

- `generic.h`

## 6.155 **Select**< false, T, U > Struct Template Reference

```
#include <inc/dtUtil/generic.h>
```

### Public Types

- typedef U **Result**

```
template<typename T, typename U> struct dtUtil::Select< false, T, U >
```

### 6.155.1 Member Typedef Documentation

#### 6.155.1.1 typedef U **Result**

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

- **generic.h**

## 6.156 Serializer Struct Reference

A place to implement functions for serialization.

```
#include <inc/dtUtil/serializer.h>
```

### Static Public Member Functions

- static XERCES\_CPP\_NAMESPACE\_QUALIFIER DOMELEMENT \* **ToBool** (const bool value, const char \*name, XERCES\_CPP\_NAMESPACE\_QUALIFIER DOMDocument \*doc)  
*Creates an XML Node.*
- static XERCES\_CPP\_NAMESPACE\_QUALIFIER DOMELEMENT \* **ToDouble** (const double value, const char \*name, XERCES\_CPP\_NAMESPACE\_QUALIFIER DOMDocument \*doc)  
*Creates an XML Node.*
- static XERCES\_CPP\_NAMESPACE\_QUALIFIER DOMELEMENT \* **ToFloat** (const float value, const char \*name, XERCES\_CPP\_NAMESPACE\_QUALIFIER DOMDocument \*doc)  
*Creates an XML Node.*
- static XERCES\_CPP\_NAMESPACE\_QUALIFIER DOMELEMENT \* **ToInt** (const int value, const char \*name, XERCES\_CPP\_NAMESPACE\_QUALIFIER DOMDocument \*doc)  
*Creates an XML Node.*
- static XERCES\_CPP\_NAMESPACE\_QUALIFIER DOMELEMENT \* **ToString** (const std::string &value, const char \*name, XERCES\_CPP\_NAMESPACE\_QUALIFIER DOMDocument \*doc)  
*Creates an XML Node.*

### 6.156.1 Detailed Description

A place to implement functions for serialization.

### 6.156.2 Member Function Documentation

#### 6.156.2.1 DOMELEMENT \* ToBool (const bool value, const char \* name, XERCES\_CPP\_NAMESPACE\_QUALIFIER DOMDocument \* doc) [static]

Creates an XML Node. Parameters

**value** The important value to be stored.

**name** The name of the XML Node to be created.

**doc** The Xerces DOMDocument required for creating new XML Nodes.

#### 6.156.2.2 DOMELEMENT \* ToDouble (const double value, const char \* name, XERCES\_CPP\_NAMESPACE\_QUALIFIER DOMDocument \* doc) [static]

Creates an XML Node. Parameters

**value** The important value to be stored.

**name** The name of the XML Node to be created.

**doc** The Xerces DOMDocument required for creating new XML Nodes.

**6.156.2.3 DOMElement \* ToFloat (const float *value*, const char \* *name*, XERCES\_CPP\_NAMESPACE\_QUALIFIER DOMDocument \* *doc*) [static]**

Creates an XML Node. Parameters

***value*** The important value to be stored.

***name*** The name of the XML Node to be created.

***doc*** The Xerces DOMDocument required for creating new XML Nodes.

**6.156.2.4 DOMElement \* ToInt (const int *value*, const char \* *name*, XERCES\_CPP\_NAMESPACE\_QUALIFIER DOMDocument \* *doc*) [static]**

Creates an XML Node. Parameters

***value*** The important value to be stored.

***name*** The name of the XML Node to be created.

***doc*** The Xerces DOMDocument required for creating new XML Nodes.

**6.156.2.5 DOMElement \* ToString (const std::string & *value*, const char \* *name*, XERCES\_CPP\_NAMESPACE\_QUALIFIER DOMDocument \* *doc*) [static]**

Creates an XML Node. Parameters

***value*** The important value to be stored.

***name*** The name of the XML Node to be created.

***doc*** The Xerces DOMDocument required for creating new XML Nodes.

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

- **serializer.h**
- **serializer.cpp**

## 6.157 squared\_difference< \_Tp, \_Dist > Struct Template Reference

```
#include <inc/dtUtil/kdtree.h>
```

### Public Types

- typedef \_Dist **distance\_type**

### Public Member Functions

- **distance\_type operator()** (const \_Tp &\_\_a, const \_Tp &\_\_b) const

```
template<typename _Tp, typename _Dist> struct dtUtil::squared_difference< _Tp, _Dist >
```

#### 6.157.1 Member Typedef Documentation

6.157.1.1 typedef \_Dist **distance\_type**

#### 6.157.2 Member Function Documentation

6.157.2.1 **distance\_type operator()** (const \_Tp & \_\_a, const \_Tp & \_\_b) const [inline]

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

- **kdtree.h**

## 6.158 squared\_difference\_counted< \_Tp, \_Dist > Struct Template Reference

```
#include <inc/dtUtil/kdtree.h>
```

### Public Types

- typedef \_Dist **distance\_type**

### Public Member Functions

- **squared\_difference\_counted** ()
- long & **count** () const
- **distance\_type operator**() (const \_Tp & \_\_a, const \_Tp & \_\_b) const
- void **reset** ()

```
template<typename _Tp, typename _Dist> struct dtUtil::squared_difference_counted< _Tp, _Dist >
```

#### 6.158.1 Member Typedef Documentation

6.158.1.1 typedef \_Dist **distance\_type**

#### 6.158.2 Constructor & Destructor Documentation

6.158.2.1 **squared\_difference\_counted** () [inline]

#### 6.158.3 Member Function Documentation

6.158.3.1 long& **count** () const [inline]

6.158.3.2 **distance\_type operator**() (const \_Tp & \_\_a, const \_Tp & \_\_b) const [inline]

6.158.3.3 void **reset** () [inline]

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

- **kdtree.h**

## 6.159 StateAttributeCollector Class Reference

```
#include <inc/dtUtil/stateattributecollector.h>
```

### Public Types

- typedef std::map< std::string, dtCore::RefPtr< osg::Material > > **MaterialNodeMap**
- typedef std::map< std::string, dtCore::RefPtr< osg::Program > > **ProgramNodeMap**
- typedef unsigned **StateAttribFlag**
- typedef std::map< std::string, dtCore::RefPtr< osg::Texture > > **TextureNodeMap**

### Public Member Functions

- **StateAttributeCollector** (osg::Node \*nodeToLoad, **StateAttributeCollector::StateAttribFlag** mask, const std::string &nodeNameIgnored="")  
*Constructor that when called will automatically generate the state attribute maps that you request.*
- **StateAttributeCollector** ()  
*Blank Constructor that is defined to do nothing.*
- void **AddMaterial** (const std::string &name, osg::Material &materialObject)  
*Function that is used to add a Material StateAttribute to the Material map.*
- void **AddProgram** (const std::string &name, osg::Program &programObject)  
*Function that is used to add a Program StateAttribute to the Program map.*
- void **AddTexture** (const std::string &name, osg::Texture &textureObject)  
*Function that is used to add a Texture StateAttribute to the Texture map.*
- void **ClearAll** ()  
*Function that is defined to clear all the maps of their contents.*
- void **CollectStateAttributes** (osg::Node \*nodeToLoad, **StateAttributeCollector::StateAttribFlag** mask, const std::string &nodeNameIgnored="")  
*Function that when called will automatically generate the state attribute maps that you request.*
- osg::Material \* **GetMaterial** (const std::string &name)  
*Function that is used to request a pointer to a Materials State Attribute.*
- const osg::Material \* **GetMaterial** (const std::string &name) const  
*Function that is used to request a CONST pointer to a Materials State Attribute.*
- **StateAttributeCollector::MaterialNodeMap & GetMaterialMap** ()  
*Function that returns a map populated with Material StateAttributes.*
- const **StateAttributeCollector::MaterialNodeMap & GetMaterialMap** () const  
*Function that returns a CONST map populated with Material StateAttributes.*
- osg::Program \* **GetProgram** (const std::string &name)  
*Function that is used to request a pointer to a Program State Attribute.*
- const osg::Program \* **GetProgram** (const std::string &name) const  
*Function that is used to request a CONST pointer to a Program State Attribute.*
- **StateAttributeCollector::ProgramNodeMap & GetProgramMap** ()  
*Function that returns a map populated with Program StateAttributes.*

- const **StateAttributeCollector::ProgramNodeMap & GetProgramMap ()** const  
*Function that returns a CONST map populated with Program StateAttributes.*
- osg::Texture \* **GetTexture** (const std::string &name)  
*Function that is used to request a pointer to a Texture State Attribute.*
- const osg::Texture \* **GetTexture** (const std::string &name) const  
*Function that is used to request a CONST pointer to a Texture State Attribute.*
- **StateAttributeCollector::TextureNodeMap & GetTextureMap ()**  
*Function that returns a map populated with Texture StateAttributes.*
- const **StateAttributeCollector::TextureNodeMap & GetTextureMap ()** const  
*Function that returns a CONST map populated with Texture StateAttributes.*

## Static Public Attributes

- static const **StateAttribFlag AllAttributes**
- static const **StateAttribFlag MaterialFlag** = dtUtil::Bits::Add(0,1)
- static const **StateAttribFlag ProgramFlag** = dtUtil::Bits::Add(0,2)
- static const **StateAttribFlag TextureFlag** = dtUtil::Bits::Add(0,4)

## Protected Member Functions

- virtual **~StateAttributeCollector ()**  
*Destructor.*

### 6.159.1 Member Typedef Documentation

6.159.1.1 **typedef std::map<std::string, dtCore::RefPtr<osg::Material> > MaterialNodeMap**

6.159.1.2 **typedef std::map<std::string, dtCore::RefPtr<osg::Program> > ProgramNodeMap**

6.159.1.3 **typedef unsigned StateAttribFlag**

6.159.1.4 **typedef std::map<std::string, dtCore::RefPtr<osg::Texture> > TextureNodeMap**

### 6.159.2 Constructor & Destructor Documentation

#### 6.159.2.1 StateAttributeCollector ()

Blank Constructor that is defined to do nothing. Note If this is used then you must call use the CollectAttributes function in order to generate any maps with this class.

#### 6.159.2.2 StateAttributeCollector (osg::Node \* *nodeToLoad*, StateAttributeCollector::StateAttribFlag *mask*, const std::string & *nodeNameIgnored* = "")

Constructor that when called will automatically generate the state attribute maps that you request. Parameters

***nodeToLoad*** The starting node who's state attributes you would like to explore.

***mask*** The different types of state attributes you want to collect off of the loaded node.

***nodeNameIgnored*** The name of a state attribute that you do not want to collect.

#### 6.159.2.3 ~StateAttributeCollector () [protected, virtual]

Destructor.

### 6.159.3 Member Function Documentation

#### 6.159.3.1 void AddMaterial (const std::string & *name*, osg::Material & *materialObject*)

Function that is used to add a Material StateAttribute to the Material map. Parameters

***name*** A String that represents the name of the StateAttribute

***materialObject*** The Material StateAttribute that you wish to add to the map

#### 6.159.3.2 void AddProgram (const std::string & *name*, osg::Program & *programObject*)

Function that is used to add a Program StateAttribute to the Program map. Parameters

***name*** A String that represents the name of the StateAttribute

***materialObject*** The Program StateAttribute that you wish to add to the map

#### 6.159.3.3 void AddTexture (const std::string & *name*, osg::Texture & *textureObject*)

Function that is used to add a Texture StateAttribute to the Texture map. Parameters

***name*** A String that represents the name of the StateAttribute

***materialObject*** The Texture StateAttribute that you wish to add to the map

#### 6.159.3.4 void ClearAll ()

Function that is defined to clear all the maps of their contents.

#### 6.159.3.5 void CollectStateAttributes (osg::Node \* *nodeToLoad*, StateAttributeCollector::StateAttribFlag *mask*, const std::string & *nodeNameIgnored* = "")

Function that when called will automatically generate the state attribute maps that you request. Parameters

***nodeToLoad*** The starting node who's state attributes you would like to explore.

***mask*** The different types of state attributes you want to collect off of the loaded node.

***nodeNameIgnored*** The name of a state attribute that you do not want to collect.

Note This function was originally intended to be used in conjunction with the blank constructor or after a call of the ClearAllMaps function

#### 6.159.3.6 osg::Material \* GetMaterial (const std::string & *name*)

Function that is used to request a pointer to a Materials State Attribute. Parameters

***name*** A String that represents the name of the Materials State Attribute you are looking for

Returns A pointer to the Materials State Attribute you were looking for or NULL if the Materials State Attribute was not found

#### 6.159.3.7 const osg::Material \* GetMaterial (const std::string & *name*) const

Function that is used to request a CONST pointer to a Materials State Attribute. Parameters

***name*** A String that represents the name of the Materials State Attribute you are looking for

Returns A CONST pointer to the Materials State Attribute you were looking for or NULL if the Materials State Attribute was not found

#### 6.159.3.8 StateAttributeCollector::MaterialNodeMap & GetMaterialMap ()

Function that returns a map populated with Material StateAttributes. Returns A map with the names of Material StateAttributes and osg Material StateAttributes

**6.159.3.9 const StateAttributeCollector::MaterialNodeMap & GetMaterialMap () const**

Function that returns a CONST map populated with Material StateAttributes. Returns A CONST map with the names of Material StateAttributes and osg Material StateAttributes

**6.159.3.10 osg::Program \* GetProgram (const std::string & name)**

Function that is used to request a pointer to a Program State Attribute. Parameters

**name** A String that represents the name of the Program State Attribute you are looking for

Returns A pointer to the Program State Attribute you were looking for or NULL if the Program State Attribute was not found

**6.159.3.11 const osg::Program \* GetProgram (const std::string & name) const**

Function that is used to request a CONST pointer to a Program State Attribute. Parameters

**name** A String that represents the name of the Program State Attribute you are looking for

Returns A CONST pointer to the Program State Attribute you were looking for or NULL if the Program State Attribute was not found

**6.159.3.12 StateAttributeCollector::ProgramNodeMap & GetProgramMap ()**

Function that returns a map populated with Program StateAttributes. Returns A map with the names of Program StateAttributes and osg Program StateAttributes

**6.159.3.13 const StateAttributeCollector::ProgramNodeMap & GetProgramMap () const**

Function that returns a CONST map populated with Program StateAttributes. Returns A CONST map with the names of Program StateAttributes and osg Program StateAttributes

**6.159.3.14 osg::Texture \* GetTexture (const std::string & name)**

Function that is used to request a pointer to a Texture State Attribute. Parameters

**name** A String that represents the name of the Texture State Attribute you are looking for

Returns A pointer to the Texture State Attribute you were looking for or NULL if the Texture State Attribute was not found

**6.159.3.15 const osg::Texture \* GetTexture (const std::string & name) const**

Function that is used to request a CONST pointer to a Texture State Attribute. Parameters

**name** A String that represents the name of the Texture State Attribute you are looking for

Returns A CONST pointer to the Texture State Attribute you were looking for or NULL if the Texture State Attribute was not found

**6.159.3.16 StateAttributeCollector::TextureNodeMap & GetTextureMap ()**

Function that returns a map populated with Texture StateAttributes. Returns A map with the names of Texture StateAttributes and osg Texture StateAttributes

**6.159.3.17 const StateAttributeCollector::TextureNodeMap & GetTextureMap () const**

Function that returns a CONST map populated with Texture StateAttributes. Returns A CONST map with the names of Texture StateAttributes and osg Texture StateAttributes

## 6.159.4 Member Data Documentation

**6.159.4.1** `const StateAttributeCollector::StateAttribFlag AllAttributes` [static]

Initial value:

```
(StateAttributeCollector::MaterialFlag |  
    StateAttributeCollector::ProgramFlag |  
    StateAttributeCollector::TextureFlag)
```

**6.159.4.2** `const StateAttributeCollector::StateAttribFlag MaterialFlag = dtUtil::Bits::Add(0,1)` [static]

**6.159.4.3** `const StateAttributeCollector::StateAttribFlag ProgramFlag = dtUtil::Bits::Add(0,2)` [static]

**6.159.4.4** `const StateAttributeCollector::StateAttribFlag TextureFlag = dtUtil::Bits::Add(0,4)` [static]

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

- `stateattributecollector.h`
- `stateattributecollector.cpp`

## 6.160 StateVisitor Class Reference

### Public Member Functions

- **StateVisitor** (**StateAttributeCollector** \*NewStateAttributeCollector, **StateAttributeCollector::StateAttribFlag** mask, const std::string &nodeNamesIgnored)  
*Constructor for the **StateVisitor** (p. 264) Class.*
- virtual void **apply** (osg::Node &group)  
*Function that visits any none Geode node and stores any StateAttributes that are associated with that Node.*
- virtual void **apply** (osg::Geode &geode)  
*Function that touches a geode and stores any StateAttributes found on it.*

### Protected Member Functions

- void **stateSetParser** (osg::StateSet \*ss)  
*Supporter function that adds StateAttributes to their corresponding map.*

### 6.160.1 Constructor & Destructor Documentation

#### 6.160.1.1 StateVisitor (StateAttributeCollector \* NewStateAttributeCollector, StateAttributeCollector::StateAttribFlag mask, const std::string & nodeNamesIgnored) [inline]

Constructor for the **StateVisitor** (p. 264) Class. Parameters

**NewStateAttributeCollector** The **StateAttributeCollector** (p. 259) Object that wants to look for different StateAttributes

**mask** The different StateAttributes you wish to look for

**nodeNamesIgnored** A name of a StateAttribute that you don't want to look for

### 6.160.2 Member Function Documentation

#### 6.160.2.1 virtual void apply (osg::Node & group) [inline, virtual]

Function that visits any none Geode node and stores any StateAttributes that are associated with that Node. Parameters

**group** A Group Node Object that will be checked

#### 6.160.2.2 virtual void apply (osg::Geode & geode) [inline, virtual]

Function that touches a geode and stores any StateAttributes found on it. It also stores the StateAttributes of all the Drawable's that may be attached to the geode. It than traverses the geode's children. Parameters

**geode** A Geode Object that you wish to visit

#### 6.160.2.3 void stateSetParser (osg::StateSet \* ss) [inline, protected]

Supporter function that adds StateAttributes to their corresponding map. Parameters

**ss** A StateSet that will be searched for StateAttributes

Note This is a supporter function that is used by all of the StateVistor's apply functions

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

- **stateattributecollector.cpp**

## 6.161 StringTokenizer< Pred > Class Template Reference

```
#include <inc/dtUtil/stringutils.h>
```

### Static Public Member Functions

- static void **tokenize** (std::vector< std::string > &tokens, const std::string &stringToParse, const Pred &predFxn=Pred())

```
template<class Pred = IsSpace> class dtUtil::StringTokenizer< Pred >
```

### 6.161.1 Member Function Documentation

**6.161.1.1 void tokenize (std::vector< std::string > & *tokens*, const std::string & *stringToParse*, const Pred & *predFxn* = Pred())** [*inline*, *static*]

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

- **stringutils.h**

## 6.162 StringToXMLConverter Class Reference

Utility methods for using strings, often for XML purposes.

```
#include <inc/dtUtil/xercesutils.h>
```

### Public Member Functions

- **StringToXMLConverter** (const std::string &charData)
- **~StringToXMLConverter** ()
- const XMLCh \* **ToXmlString** ()

#### 6.162.1 Detailed Description

Utility methods for using strings, often for XML purposes. This is a simple class that lets us do easy (though not terribly efficient) transcoding of string data to XMLCh.

#### 6.162.2 Constructor & Destructor Documentation

**6.162.2.1 StringToXMLConverter (const std::string & charData) [inline]**

**6.162.2.2 ~StringToXMLConverter () [inline]**

#### 6.162.3 Member Function Documentation

**6.162.3.1 const XMLCh\* ToXmlString () [inline]**

Returns the XMLCh string as a char\*

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

- **xercesutils.h**

## 6.163 TailAt< InstantiateH< TypeList< T, U >, Holder, i >, 0, i > Struct Template Reference

```
#include <inc/dtUtil/generic.h>
```

### Public Types

- typedef InstantiateH< typename **TypeList**< T, U >::Tail, Holder, i+1 > **Result**

```
template<typename T, typename U, template< class, unsigned int > class Holder, unsigned int i> struct dtUtil::TailAt< InstantiateH< TypeList< T, U >, Holder, i >, 0, i >
```

### 6.163.1 Member Typedef Documentation

#### 6.163.1.1 typedef InstantiateH<typename TypeList<T, U>::Tail, Holder, i+1> Result

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

- **generic.h**

## 6.164 TailAt< InstantiateH< TypeList< T, U >, Holder, i >, j, i > Struct Template Reference

```
#include <inc/dtUtil/generic.h>
```

### Public Types

- typedef TailAt< InstantiateH< typename **TypeList**< T, U >::Tail, Holder, i+1 >, j-1, i+1 >::Result **Result**

```
template<typename T, typename U, template< class, unsigned int > class Holder, unsigned int j, unsigned int i> struct dtUtil::TailAt< InstantiateH< TypeList< T, U >, Holder, i >, j, i >
```

### 6.164.1 Member Typedef Documentation

#### 6.164.1.1 typedef TailAt<InstantiateH<typename TypeList<T, U>::Tail, Holder, i+1>, j-1, i+1>::Result Result

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

- **generic.h**

## 6.165 TangentSpaceVisitor Class Reference

This visitor is used to generate tangents for your node.

```
#include <inc/dtUtil/tangentspacevisitor.h>
```

### Public Member Functions

- **TangentSpaceVisitor** (const std::string &vertexAttributeName, osg::Program \*shaderProgram=NULL, int tangentVertexAttribNumber=6)
- void **apply** (osg::Geode &geode)

#### 6.165.1 Detailed Description

This visitor is used to generate tangents for your node. It will stick the tangents on a Vertex Attribute Array. In Open GL Shader Language, each vertex can have a number of attributes (minimum supported is 16 = 0 to 15). You can use these attributes to put a single value for each vertex. This visitor generates tangents for each vertex and puts them in the specified array number (0 to 15). If you pass in the shader program, it will also bind the array number to a name, so you can use it in your Vertex Shader like this: attribute vec4 vTangent; Typically, tangents are put in index 6. An example usage looks like this: dtCore::RefPtr<TangentSpaceVisitor> visitor = new **TangentSpaceVisitor** (p.269) (defaultShader, 6, "vTangent"); mMyCoolNode->accept(\*visitor.get()); This visitor was based on osgFx::BumpMapping.cpp

#### 6.165.2 Constructor & Destructor Documentation

**6.165.2.1 TangentSpaceVisitor** (const std::string & *vertexAttributeName*, osg::Program \* *shaderProgram* = NULL, int *tangentVertexAttribNumber* = 6)

#### 6.165.3 Member Function Documentation

**6.165.3.1 void apply** (osg::Geode & *geode*)

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

- **tangentspacevisitor.h**
- **tangentspacevisitor.cpp**

## 6.166 Temp< TL2, i, IdsTL2, Predicate > Struct Template Reference

```
#include <inc/dtUtil/funbind.h>
```

### Public Types

- typedef **dtUtil::Select**< Predicate< typename IdsTL2::Head, i >::value, typename **dtUtil::TypeList**< typename TL2::Head, typename **Temp**< typename TL2::Tail, i+1, typename IdsTL2::Tail, Predicate >::Result >, typename **Temp**< typename TL2::Tail, i+1, IdsTL2, Predicate >::Result >::Result **Result**

```
template<class TL, class IdsTL, bool include>template<class TL2, int i, class IdsTL2, template< class, int > class Predicate> struct dtUtil::Filter2< TL, IdsTL, include >::Temp< TL2, i, IdsTL2, Predicate >
```

### 6.166.1 Member Typedef Documentation

6.166.1.1 typedef **dtUtil::Select**< Predicate<typename IdsTL2::Head, i>::value, typename **dtUtil::TypeList**<typename TL2::Head, typename **Temp**<typename TL2::Tail, i+1, typename IdsTL2::Tail, Predicate>::Result>, typename **Temp**<typename TL2::Tail, i+1, IdsTL2, Predicate>::Result >::Result **Result**

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

- **funbind.h**

## 6.167 Temp< dtUtil::NullType, i, dtUtil::NullType, Predicate > Struct Template Reference

```
#include <inc/dtUtil/funbind.h>
```

### Public Types

- typedef dtUtil::NullType Result

```
template<class TL, class IdsTL, bool include>template<int i, template< class, int > class Predicate>  
struct dtUtil::Filter2< TL, IdsTL, include >::Temp< dtUtil::NullType, i, dtUtil::NullType, Predicate >
```

### 6.167.1 Member Typedef Documentation

#### 6.167.1.1 typedef dtUtil::NullType Result

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

- funbind.h

## 6.168 Temp< dtUtil::NullType, i, IdsTL2, Predicate > Struct Template Reference

```
#include <inc/dtUtil/funbind.h>
```

### Public Types

- typedef dtUtil::NullType Result

```
template<class TL, class IdsTL, bool include>template<int i, class IdsTL2, template< class, int > class  
Predicate> struct dtUtil::Filter2< TL, IdsTL, include >::Temp< dtUtil::NullType, i, IdsTL2, Predicate >
```

### 6.168.1 Member Typedef Documentation

#### 6.168.1.1 typedef dtUtil::NullType Result

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

- funbind.h

## 6.169 Temp< TL2, 0, dtUtil::NullType, dtUtil::IsIntType > Struct Template Reference

```
#include <inc/dtUtil/funbind.h>
```

### Public Types

- typedef dtUtil::NullType Result

```
template<class TL, class IdsTL, bool include>template<class TL2> struct dtUtil::Filter2< TL, IdsTL, include >::Temp< TL2, 0, dtUtil::NullType, dtUtil::IsIntType >
```

### 6.169.1 Member Typedef Documentation

#### 6.169.1.1 typedef dtUtil::NullType Result

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

- funbind.h

## 6.170 Temp< TL2, 0, dtUtil::NullType, dtUtil::NotIntType > Struct Template Reference

```
#include <inc/dtUtil/funbind.h>
```

### Public Types

- typedef TL2 Result

```
template<class TL, class IdsTL, bool include>template<class TL2> struct dtUtil::Filter2< TL, IdsTL, include >::Temp< TL2, 0, dtUtil::NullType, dtUtil::NotIntType >
```

### 6.170.1 Member Typedef Documentation

#### 6.170.1.1 typedef TL2 Result

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

- funbind.h

## 6.171 Temp< TL2, 0, typename dtUtil::ldsFromTL< TL2 >::Type, dtUtil::lsIntType > Struct Template Reference

```
#include <inc/dtUtil/funbind.h>
```

### Public Types

- typedef TL2 Result

```
template<class TL, class ldsTL, bool include>template<class TL2> struct dtUtil::Filter2< TL, ldsTL, include >::Temp< TL2, 0, typename dtUtil::ldsFromTL< TL2 >::Type, dtUtil::lsIntType >
```

### 6.171.1 Member Typedef Documentation

#### 6.171.1.1 typedef TL2 Result

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

- funbind.h

## 6.172 Temp< TL2, 0, typename dtUtil::IdsFromTL< TL2 >::Type, dtUtil::NotIntType > Struct Template Reference

```
#include <inc/dtUtil/funbind.h>
```

### Public Types

- typedef dtUtil::NullType Result

```
template<class TL, class IdsTL, bool include>template<class TL2> struct dtUtil::Filter2< TL, IdsTL, include >::Temp< TL2, 0, typename dtUtil::IdsFromTL< TL2 >::Type, dtUtil::NotIntType >
```

### 6.172.1 Member Typedef Documentation

#### 6.172.1.1 typedef dtUtil::NullType Result

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

- funbind.h

## 6.173 Temp< TL2, i, dtUtil::NullType, Predicate > Struct Template Reference

```
#include <inc/dtUtil/funbind.h>
```

### Public Types

- typedef dtUtil::Select< include, dtUtil::NullType, TL2 >::Result Result

```
template<class TL, class IdsTL, bool include>template<class TL2, int i, template< class, int > class  
Predicate> struct dtUtil::Filter2< TL, IdsTL, include >::Temp< TL2, i, dtUtil::NullType, Predicate >
```

### 6.173.1 Member Typedef Documentation

#### 6.173.1.1 typedef dtUtil::Select<include, dtUtil::NullType, TL2>::Result Result

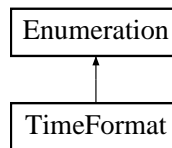
The documentation for this struct was generated from the following file:

- funbind.h

## 6.174 TimeFormat Class Reference

The **TimeFormat** (p. 278) enumeration is used to specify how to map a **DateTime** (p. 96) object to a string.

#include <inc/dtUtil/datetime.h>Inheritance diagram for TimeFormat::



### Static Public Attributes

- static const **TimeFormat** CALENDAR\_DATE\_AND\_TIME\_FORMAT
- static const **TimeFormat** CALENDAR\_DATE\_FORMAT
- static const **TimeFormat** CLOCK\_TIME\_12\_HOUR\_FORMAT
- static const **TimeFormat** CLOCK\_TIME\_24\_HOUR\_FORMAT
- static const **TimeFormat** LEXICAL\_DATE\_FORMAT
- static const **TimeFormat** LOCAL\_DATE\_AND\_TIME\_FORMAT
- static const **TimeFormat** LOCAL\_DATE\_FORMAT
- static const **TimeFormat** ORDINAL\_DATE\_FORMAT
- static const **TimeFormat** WEEK\_DATE\_FORMAT

### Protected Member Functions

- ~**TimeFormat** ()

#### 6.174.1 Detailed Description

The **TimeFormat** (p. 278) enumeration is used to specify how to map a **DateTime** (p. 96) object to a string. This enumeration is used in conjunction with **ToString()** (p. 102).

#### 6.174.2 Constructor & Destructor Documentation

6.174.2.1 ~**TimeFormat** () [inline, protected]

#### 6.174.3 Member Data Documentation

6.174.3.1 const **DateTime::TimeFormat** CALENDAR\_DATE\_AND\_TIME\_FORMAT [static]

6.174.3.2 const **DateTime::TimeFormat** CALENDAR\_DATE\_FORMAT [static]

6.174.3.3 const **DateTime::TimeFormat** CLOCK\_TIME\_12\_HOUR\_FORMAT [static]

6.174.3.4 const **DateTime::TimeFormat** CLOCK\_TIME\_24\_HOUR\_FORMAT [static]

6.174.3.5 const **DateTime::TimeFormat** LEXICAL\_DATE\_FORMAT [static]

6.174.3.6 const **DateTime::TimeFormat** LOCAL\_DATE\_AND\_TIME\_FORMAT [static]

6.174.3.7 const **DateTime::TimeFormat** LOCAL\_DATE\_FORMAT [static]

6.174.3.8 const **DateTime::TimeFormat** ORDINAL\_DATE\_FORMAT [static]

6.174.3.9 const **DateTime::TimeFormat** WEEK\_DATE\_FORMAT [static]

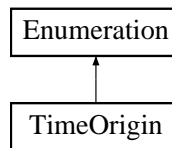
The documentation for this class was generated from the following files:

- **datetime.h**
- **datetime.cpp**

## 6.175 TimeOrigin Class Reference

The **TimeOrigin** (p. 279) enumeration determines how the instance of **DateTime** (p. 96) should be interpreted.

#include <inc/dtUtil/datetime.h>Inheritance diagram for TimeOrigin::



### Static Public Attributes

- static const **TimeOrigin** GMT\_TIME
- static const **TimeOrigin** LOCAL\_TIME

### Protected Member Functions

- ~**TimeOrigin** ()

#### 6.175.1 Detailed Description

The **TimeOrigin** (p. 279) enumeration determines how the instance of **DateTime** (p. 96) should be interpreted. When passing a **TimeOrigin** (p. 279) into the constructor of **DateTime** (p. 96) it will automatically set the clock to that specific time.

#### 6.175.2 Constructor & Destructor Documentation

6.175.2.1 ~**TimeOrigin** () [inline, protected]

#### 6.175.3 Member Data Documentation

6.175.3.1 const **DateTime::TimeOrigin** GMT\_TIME [static]

6.175.3.2 const **DateTime::TimeOrigin** LOCAL\_TIME [static]

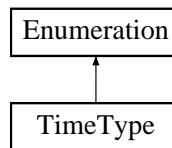
The documentation for this class was generated from the following files:

- **datetime.h**
- **datetime.cpp**

## 6.176 TimeType Class Reference

The **TimeType** (p. 280) enumeration is used to identify the purpose of a **DateTime** (p. 96) instance.

#include <inc/dtUtil/datetime.h>Inheritance diagram for TimeType::



### Static Public Attributes

- static const **TimeType** **CLOCK\_TIME**
- static const **TimeType** **SCENARIO\_TIME**
- static const **TimeType** **SIMULATION\_TIME**
- static const **TimeType** **TIME\_STAMP**
- static const **TimeType** **TIME\_TYPE\_OTHER**
- static const **TimeType** **TRIP\_TIME**

### Protected Member Functions

- ~**TimeType** ()

#### 6.176.1 Detailed Description

The **TimeType** (p. 280) enumeration is used to identify the purpose of a **DateTime** (p. 96) instance.

#### 6.176.2 Constructor & Destructor Documentation

6.176.2.1 ~**TimeType** () [inline, protected]

#### 6.176.3 Member Data Documentation

6.176.3.1 const **DateTime::TimeType** **CLOCK\_TIME** [static]

6.176.3.2 const **DateTime::TimeType** **SCENARIO\_TIME** [static]

6.176.3.3 const **DateTime::TimeType** **SIMULATION\_TIME** [static]

6.176.3.4 const **DateTime::TimeType** **TIME\_STAMP** [static]

6.176.3.5 const **DateTime::TimeType** **TIME\_TYPE\_OTHER** [static]

6.176.3.6 const **DateTime::TimeType** **TRIP\_TIME** [static]

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

- **datetime.h**
- **datetime.cpp**

## 6.177 ToLowerClass Class Reference

### Public Member Functions

- **ToLowerClass** ()
- char **operator**() (char &elem) const

### 6.177.1 Constructor & Destructor Documentation

6.177.1.1 **ToLowerClass** () [inline]

### 6.177.2 Member Function Documentation

6.177.2.1 char **operator**() (char & *elem*) const [inline]

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

- **fileutils.cpp**

## 6.178 Transformation< T > Class Template Reference

An interface class to use for "things that transform" i.e., things that return a value given an input value.

```
#include <inc/dtUtil/transformation.h>
```

### Public Member Functions

- virtual T **operator()** (T) const =0

#### 6.178.1 Detailed Description

```
template<typename T> class dtUtil::Transformation< T >
```

An interface class to use for "things that transform" i.e., things that return a value given an input value.

#### 6.178.2 Member Function Documentation

**6.178.2.1 virtual T operator() (T) const [pure virtual]**

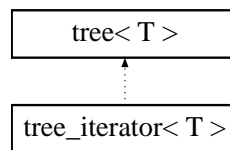
Implemented in **EdgeStepFilter** (p. 108).

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

- **transformation.h**

## 6.179 tree< T > Class Template Reference

#include <inc/dtUtil/tree.h> Inheritance diagram for tree< T >::



### Public Types

- typedef const **tree\_iterator**< T > **const\_iterator**
- typedef **tree\_iterator**< T > **iterator**

### Public Member Functions

- **tree** (const **tree** &in)
- **tree** (const T &inT)
- **tree** ()
- virtual ~**tree** ()
- **iterator begin** ()
- **const\_iterator begin** () const
- void **clear** ()
- void **copy\_tree** (const **tree** &in)
- const T & **data** (const T &inData)
- const T & **data** () const
- T & **data** ()
- **iterator & end** () const
- bool **erase** (const **iterator** &i)
- **iterator find** (const T &inT, const **iterator** &iter, bool(\*obj)(const T &, const T &)) const
- **iterator find** (const T &inT, const **iterator** &iter) const
- **iterator find** (const T &inT, bool(\*obj)(const T &, const T &)) const
- **iterator find** (const T &inT) const
- **iterator get\_tree\_iterator** ()
- **iterator get\_tree\_iterator** () const
- **iterator in** ()
- **const\_iterator in** () const
- **iterator insert** (const T &inT)
- **iterator insert** (const **iterator** &i)
- **iterator insert** (const T &inT, bool(\*pObj)(const T &, const T &))
- size\_t **level** () const
- const T & **operator\*** () const
- T & **operator\*** ()
- const **tree** & **operator=** (const **tree** &in)
- const bool **operator==** (const **tree** &inTree) const
- **iterator operator[]** (size\_t loc)
- **iterator operator[]** (size\_t loc) const
- **iterator out** ()
- **const\_iterator out** () const
- **iterator push\_back** (const T &inT)
- **iterator push\_front** (const T &inT)
- **iterator reinsert** (**tree** \*in)
- **iterator reinsert** (**tree** \*in, bool(\*pObj)(const T &, const T &))
- bool **remove** (const T &inData)
- size\_t **size** () const

- **iterator tree\_find\_breadth** (const T &inT, const **iterator** &iter, bool(\*obj)(const T &, const T &)) const
- **iterator tree\_find\_breadth** (const T &inT, const **iterator** &iter) const
- **iterator tree\_find\_breadth** (const T &inT, bool(\*obj)(const T &, const T &)) const
- **iterator tree\_find\_breadth** (const T &inT) const
- **iterator tree\_find\_depth** (const T &inT, const **iterator** &iter, bool(\*obj)(const T &, const T &)) const
- **iterator tree\_find\_depth** (const T &inT, const **iterator** &iter) const
- **iterator tree\_find\_depth** (const T &inT, bool(\*obj)(const T &, const T &)) const
- **iterator tree\_find\_depth** (const T &inT) const

## Protected Member Functions

- **iterator begin** (const **tree** &in) const
- **iterator in** (const **tree** &in) const
- **size\_t level** (const **tree** &in) const
- **const tree \* next** (const **tree** &in) const
- **iterator out** (const **tree** &in) const
- **const tree \* prev** (const **tree** &in) const
- **size\_t size** (const **tree** &in) const

**template**<typename T> **class** dtUtil::tree< T >

### 6.179.1 Member Typedef Documentation

#### 6.179.1.1 typedef const tree\_iterator<T> const\_iterator

Reimplemented in `tree_iterator< T >` (p. 288).

#### 6.179.1.2 typedef tree\_iterator<T> iterator

Reimplemented in `tree_iterator< T >` (p. 288).

### 6.179.2 Constructor & Destructor Documentation

#### 6.179.2.1 tree () [inline]

#### 6.179.2.2 tree (const T & inT) [inline]

#### 6.179.2.3 tree (const tree< T > & in) [inline]

#### 6.179.2.4 virtual ~tree () [inline, virtual]

### 6.179.3 Member Function Documentation

#### 6.179.3.1 iterator begin () [inline]

Reimplemented in `tree_iterator< T >` (p. 288).

#### 6.179.3.2 const\_iterator begin () const [inline]

Reimplemented in `tree_iterator< T >` (p. 288).

#### 6.179.3.3 iterator begin (const tree< T > & in) const [inline, protected]

#### 6.179.3.4 void clear () [inline]

#### 6.179.3.5 void copy\_tree (const tree< T > & in) [inline]

#### 6.179.3.6 const T& data (const T & inData) [inline]

#### 6.179.3.7 const T& data () const [inline]

Reimplemented in `tree_iterator< T >` (p. 288).

#### 6.179.3.8 T& data () [inline]

Reimplemented in `tree_iterator< T >` (p. 288).

**6.179.3.9** iterator& end () const [inline]

Reimplemented in tree\_iterator&lt; T &gt; (p. 289).

**6.179.3.10** bool erase (const iterator & *i*) [inline]**6.179.3.11** iterator find (const T & *inT*, const iterator & *iter*, bool (\*)(const T &, const T &) *obj*) const [inline]

Reimplemented in tree\_iterator&lt; T &gt; (p. 289).

**6.179.3.12** iterator find (const T & *inT*, const iterator & *iter*) const [inline]

Reimplemented in tree\_iterator&lt; T &gt; (p. 289).

**6.179.3.13** iterator find (const T & *inT*, bool (\*)(const T &, const T &) *obj*) const [inline]

Reimplemented in tree\_iterator&lt; T &gt; (p. 289).

**6.179.3.14** iterator find (const T & *inT*) const [inline]

Reimplemented in tree\_iterator&lt; T &gt; (p. 289).

**6.179.3.15** iterator get\_tree\_iterator () [inline]**6.179.3.16** iterator get\_tree\_iterator () const [inline]**6.179.3.17** iterator in () [inline]

Reimplemented in tree\_iterator&lt; T &gt; (p. 289).

**6.179.3.18** const\_iterator in () const [inline]

Reimplemented in tree\_iterator&lt; T &gt; (p. 289).

**6.179.3.19** iterator in (const tree< T > & *in*) const [inline, protected]**6.179.3.20** iterator insert (const T & *inT*) [inline]

Reimplemented in tree\_iterator&lt; T &gt; (p. 289).

**6.179.3.21** iterator insert (const iterator & *i*) [inline]**6.179.3.22** iterator insert (const T & *inT*, bool (\*)(const T &, const T &) *pObj*) [inline]

Reimplemented in tree\_iterator&lt; T &gt; (p. 289).

**6.179.3.23** size\_t level () const [inline]

Reimplemented in tree\_iterator&lt; T &gt; (p. 289).

**6.179.3.24** size\_t level (const tree< T > & *in*) const [inline, protected]**6.179.3.25** const tree\* next (const tree< T > & *in*) const [inline, protected]**6.179.3.26** const T& operator\* () const [inline]

Reimplemented in tree\_iterator&lt; T &gt; (p. 289).

**6.179.3.27** T& operator\* () [inline]

Reimplemented in tree\_iterator&lt; T &gt; (p. 289).

**6.179.3.28** const tree& operator= (const tree< T > & *in*) [inline]**6.179.3.29** const bool operator== (const tree< T > & *inTree*) const [inline]**6.179.3.30** iterator operator[] (size\_t *loc*) [inline]

Reimplemented in tree\_iterator&lt; T &gt; (p. 289).

**6.179.3.31** iterator operator[] (size\_t *loc*) const [inline]

Reimplemented in tree\_iterator&lt; T &gt; (p. 290).

**6.179.3.32** `iterator out () [inline]`

Reimplemented in `tree_iterator< T >` (p. 290).

**6.179.3.33** `const_iterator out () const [inline]`

Reimplemented in `tree_iterator< T >` (p. 290).

**6.179.3.34** `iterator out (const tree< T > & in) const [inline, protected]`**6.179.3.35** `const tree* prev (const tree< T > & in) const [inline, protected]`**6.179.3.36** `iterator push_back (const T & inT) [inline]`

Reimplemented in `tree_iterator< T >` (p. 290).

**6.179.3.37** `iterator push_front (const T & inT) [inline]`

Reimplemented in `tree_iterator< T >` (p. 290).

**6.179.3.38** `iterator reinsert (tree< T > * in) [inline]`**6.179.3.39** `iterator reinsert (tree< T > * in, bool(*)(const T &, const T &) pObj) [inline]`**6.179.3.40** `bool remove (const T & inData) [inline]`

Reimplemented in `tree_iterator< T >` (p. 290).

**6.179.3.41** `size_t size () const [inline]`

Reimplemented in `tree_iterator< T >` (p. 290).

**6.179.3.42** `size_t size (const tree< T > & in) const [inline, protected]`**6.179.3.43** `iterator tree_find_breadth (const T & inT, const iterator & iter, bool(*)(const T &, const T &) obj) const [inline]`

Reimplemented in `tree_iterator< T >` (p. 290).

**6.179.3.44** `iterator tree_find_breadth (const T & inT, const iterator & iter) const [inline]`

Reimplemented in `tree_iterator< T >` (p. 290).

**6.179.3.45** `iterator tree_find_breadth (const T & inT, bool(*)(const T &, const T &) obj) const [inline]`

Reimplemented in `tree_iterator< T >` (p. 290).

**6.179.3.46** `iterator tree_find_breadth (const T & inT) const [inline]`

Reimplemented in `tree_iterator< T >` (p. 290).

**6.179.3.47** `iterator tree_find_depth (const T & inT, const iterator & iter, bool(*)(const T &, const T &) obj) const [inline]`

Reimplemented in `tree_iterator< T >` (p. 290).

**6.179.3.48** `iterator tree_find_depth (const T & inT, const iterator & iter) const [inline]`

Reimplemented in `tree_iterator< T >` (p. 290).

**6.179.3.49** `iterator tree_find_depth (const T & inT, bool(*)(const T &, const T &) obj) const [inline]`

Reimplemented in `tree_iterator< T >` (p. 290).

**6.179.3.50** `iterator tree_find_depth (const T & inT) const [inline]`

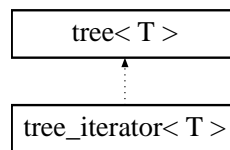
Reimplemented in `tree_iterator< T >` (p. 290).

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

- `tree.h`

## 6.180 tree\_iterator< T > Class Template Reference

#include <inc/dtUtil/tree.h> Inheritance diagram for tree\_iterator< T >::



### Public Types

- typedef **tree< T >::const\_iterator** **const\_iterator**
- typedef **tree< T >::iterator** **iterator**

### Public Member Functions

- **tree\_iterator** (const **TreeType** &tree\_ref)
- **tree\_iterator** (**TreeType** &tree\_ref)
- **tree\_iterator** (const **tree\_iterator** &i)
- **tree\_iterator** ()
- ~**tree\_iterator** ()
- **iterator begin** ()
- **iterator begin** () const
- void **clear\_children** ()
- void **clear\_tree** ()
- const T & **data** (const T &inData) const
- const T & **data** () const
- T & **data** ()
- const **iterator** & **end** () const
- **iterator find** (const T &inT, const **iterator** &iter, bool(\*obj)(const T &, const T &)) const
- **iterator find** (const T &inT, const **iterator** &iter) const
- **iterator find** (const T &inT, bool(\*obj)(const T &, const T &)) const
- **iterator find** (const T &inT) const
- **iterator in** ()
- **iterator in** () const
- **iterator insert** (const T &t, bool(\*obj)(const T &, const T &))
- **iterator insert** (const T &t)
- **size\_t level** () const
- **iterator next** () const
- bool **operator!=** (const **tree\_iterator** &rhs) const
- const T & **operator\*** () const
- T & **operator\*** ()
- **iterator operator++** (int) const
- const **iterator** & **operator++** () const
- const **iterator** & **operator--** () const
- const T \* **operator->** () const
- T \* **operator->** ()
- const **iterator** & **operator=** (const **TreeType** &rhs) const
- const **iterator** & **operator=** (const **tree\_iterator** &iter) const
- **iterator** & **operator=** (const **tree\_iterator** &iter)
- bool **operator==** (const **tree\_iterator** &rhs) const
- **iterator operator[]** (size\_t loc)
- **iterator operator[]** (size\_t loc) const
- **iterator out** ()
- **iterator out** () const

- **iterator push\_back** (const T &t)
- **iterator push\_front** (const T &t)
- **iterator reinsert** (const iterator &in)
- **iterator reinsert** (const iterator &in, bool(\*obj)(const T &, const T &))
- **bool remove** (const T &inT)
- **size\_t size** () const
- **iterator tree\_find\_breadth** (const T &inT, const iterator &iter, bool(\*obj)(const T &, const T &)) const
- **iterator tree\_find\_breadth** (const T &inT, const iterator &iter) const
- **iterator tree\_find\_breadth** (const T &inT, bool(\*obj)(const T &, const T &)) const
- **iterator tree\_find\_breadth** (const T &inT) const
- **iterator tree\_find\_depth** (const T &inT, const iterator &iter, bool(\*obj)(const T &, const T &)) const
- **iterator tree\_find\_depth** (const T &inT, const iterator &iter) const
- **iterator tree\_find\_depth** (const T &inT, bool(\*obj)(const T &, const T &)) const
- **iterator tree\_find\_depth** (const T &inT) const
- **TreeType \* tree\_ptr** () const
- **TreeType & tree\_ref** () const

### Static Public Member Functions

- static **iterator & end\_iterator** ()

template<typename T> class dtUtil::tree\_iterator< T >

#### 6.180.1 Member Typedef Documentation

##### 6.180.1.1 typedef tree<T>::const\_iterator const\_iterator

Reimplemented from `tree< T >` (p. 284).

##### 6.180.1.2 typedef tree<T>::iterator iterator

Reimplemented from `tree< T >` (p. 284).

#### 6.180.2 Constructor & Destructor Documentation

##### 6.180.2.1 tree\_iterator () [inline]

##### 6.180.2.2 tree\_iterator (const tree\_iterator< T > & i) [inline]

##### 6.180.2.3 tree\_iterator (TreeType & tree\_ref) [inline]

##### 6.180.2.4 tree\_iterator (const TreeType & tree\_ref) [inline]

##### 6.180.2.5 ~tree\_iterator () [inline]

#### 6.180.3 Member Function Documentation

##### 6.180.3.1 iterator begin () [inline]

Reimplemented from `tree< T >` (p. 284).

##### 6.180.3.2 iterator begin () const [inline]

Reimplemented from `tree< T >` (p. 284).

##### 6.180.3.3 void clear\_children () [inline]

##### 6.180.3.4 void clear\_tree () [inline]

##### 6.180.3.5 const T& data (const T & inData) const [inline]

##### 6.180.3.6 const T& data () const [inline]

Reimplemented from `tree< T >` (p. 284).

##### 6.180.3.7 T& data () [inline]

Reimplemented from `tree< T >` (p. 284).

**6.180.3.8** const iterator& end () const [inline]

Reimplemented from tree&lt; T &gt; (p. 285).

**6.180.3.9** static iterator& end\_iterator () [inline, static]**6.180.3.10** iterator find (const T & *inT*, const iterator & *iter*, bool(\*)(const T &, const T &) *obj*) const [inline]

Reimplemented from tree&lt; T &gt; (p. 285).

**6.180.3.11** iterator find (const T & *inT*, const iterator & *iter*) const [inline]

Reimplemented from tree&lt; T &gt; (p. 285).

**6.180.3.12** iterator find (const T & *inT*, bool(\*)(const T &, const T &) *obj*) const [inline]

Reimplemented from tree&lt; T &gt; (p. 285).

**6.180.3.13** iterator find (const T & *inT*) const [inline]

Reimplemented from tree&lt; T &gt; (p. 285).

**6.180.3.14** iterator in () [inline]

Reimplemented from tree&lt; T &gt; (p. 285).

**6.180.3.15** iterator in () const [inline]

Reimplemented from tree&lt; T &gt; (p. 285).

**6.180.3.16** iterator insert (const T & *t*, bool(\*)(const T &, const T &) *obj*) [inline]

Reimplemented from tree&lt; T &gt; (p. 285).

**6.180.3.17** iterator insert (const T & *f*) [inline]

Reimplemented from tree&lt; T &gt; (p. 285).

**6.180.3.18** size\_t level () const [inline]

Reimplemented from tree&lt; T &gt; (p. 285).

**6.180.3.19** iterator next () const [inline]**6.180.3.20** bool operator!= (const tree\_iterator< T > & *rhs*) const [inline]**6.180.3.21** const T& operator\* () const [inline]

Reimplemented from tree&lt; T &gt; (p. 285).

**6.180.3.22** T& operator\* () [inline]

Reimplemented from tree&lt; T &gt; (p. 285).

**6.180.3.23** iterator operator++ (int) const [inline]**6.180.3.24** const iterator& operator++ () const [inline]**6.180.3.25** const iterator& operator-- () const [inline]**6.180.3.26** const T\* operator-> () const [inline]**6.180.3.27** T\* operator-> () [inline]**6.180.3.28** const iterator& operator= (const TreeType & *rhs*) const [inline]**6.180.3.29** const iterator& operator= (const tree\_iterator< T > & *iter*) const [inline]**6.180.3.30** iterator& operator= (const tree\_iterator< T > & *iter*) [inline]**6.180.3.31** bool operator== (const tree\_iterator< T > & *rhs*) const [inline]**6.180.3.32** iterator operator[] (size\_t *loc*) [inline]

Reimplemented from tree&lt; T &gt; (p. 285).

**6.180.3.33 iterator operator[] (size\_t loc) const [inline]**

Reimplemented from `tree< T >` (p. 285).

**6.180.3.34 iterator out () [inline]**

Reimplemented from `tree< T >` (p. 286).

**6.180.3.35 iterator out () const [inline]**

Reimplemented from `tree< T >` (p. 286).

**6.180.3.36 iterator push\_back (const T & t) [inline]**

Reimplemented from `tree< T >` (p. 286).

**6.180.3.37 iterator push\_front (const T & t) [inline]**

Reimplemented from `tree< T >` (p. 286).

**6.180.3.38 iterator reinsert (const iterator & in) [inline]****6.180.3.39 iterator reinsert (const iterator & in, bool (\*)(const T &, const T &) obj) [inline]****6.180.3.40 bool remove (const T & inT) [inline]**

Reimplemented from `tree< T >` (p. 286).

**6.180.3.41 size\_t size () const [inline]**

Reimplemented from `tree< T >` (p. 286).

**6.180.3.42 iterator tree\_find\_breadth (const T & inT, const iterator & iter, bool (\*)(const T &, const T &) obj) const [inline]**

Reimplemented from `tree< T >` (p. 286).

**6.180.3.43 iterator tree\_find\_breadth (const T & inT, const iterator & iter) const [inline]**

Reimplemented from `tree< T >` (p. 286).

**6.180.3.44 iterator tree\_find\_breadth (const T & inT, bool (\*)(const T &, const T &) obj) const [inline]**

Reimplemented from `tree< T >` (p. 286).

**6.180.3.45 iterator tree\_find\_breadth (const T & inT) const [inline]**

Reimplemented from `tree< T >` (p. 286).

**6.180.3.46 iterator tree\_find\_depth (const T & inT, const iterator & iter, bool (\*)(const T &, const T &) obj) const [inline]**

Reimplemented from `tree< T >` (p. 286).

**6.180.3.47 iterator tree\_find\_depth (const T & inT, const iterator & iter) const [inline]**

Reimplemented from `tree< T >` (p. 286).

**6.180.3.48 iterator tree\_find\_depth (const T & inT, bool (\*)(const T &, const T &) obj) const [inline]**

Reimplemented from `tree< T >` (p. 286).

**6.180.3.49 iterator tree\_find\_depth (const T & inT) const [inline]**

Reimplemented from `tree< T >` (p. 286).

**6.180.3.50 TreeType\* tree\_ptr () const [inline]****6.180.3.51 TreeType& tree\_ref () const [inline]**

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

- `tree.h`

## 6.181 TupleHolder< T, i > Struct Template Reference

```
#include <inc/dtUtil/generic.h>
```

### Public Types

- typedef T **StoredType**
- typedef T **Type**

### Public Member Functions

- **TupleHolder** (Type t)
- **TupleHolder** ()
- **TupleHolder** & **operator=** (TupleHolder const &v)

### Public Attributes

- **StoredType** value

```
template<typename T, unsigned int i = 0> struct dtUtil::TupleHolder< T, i >
```

#### 6.181.1 Member Typedef Documentation

6.181.1.1 typedef T **StoredType**

6.181.1.2 typedef T **Type**

#### 6.181.2 Constructor & Destructor Documentation

6.181.2.1 **TupleHolder** () [inline]

6.181.2.2 **TupleHolder** (Type t) [inline]

#### 6.181.3 Member Function Documentation

6.181.3.1 **TupleHolder**& **operator=** (TupleHolder< T, i > const & v) [inline]

#### 6.181.4 Member Data Documentation

6.181.4.1 **StoredType** value

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

- **generic.h**

## 6.182 TypeAt< TypeList< T, U >, 0 > Struct Template Reference

```
#include <inc/dtUtil/generic.h>
```

### Public Types

- typedef T Result

```
template<class T, class U> struct dtUtil::TypeAt< TypeList< T, U >, 0 >
```

### 6.182.1 Member Typedef Documentation

#### 6.182.1.1 typedef T Result

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

- **generic.h**

## 6.183 TypeAt< TypeList< T, U >, i > Struct Template Reference

```
#include <inc/dtUtil/generic.h>
```

### Public Types

- `typedef TypeAt< U, i-1 >::Result Result`

```
template<class T, class U, unsigned int i> struct dtUtil::TypeAt< TypeList< T, U >, i >
```

### 6.183.1 Member Typedef Documentation

#### 6.183.1.1 `typedef TypeAt<U, i - 1>::Result Result`

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

- `generic.h`

## 6.184 TypeAtNonStrict< TList, i, DefType > Struct Template Reference

```
#include <inc/dtUtil/generic.h>
```

### Public Types

- typedef DefType **Result**

```
template<class TList, unsigned int i, typename DefType = NullType> struct dtUtil::TypeAtNonStrict<TList, i, DefType >
```

### 6.184.1 Member Typedef Documentation

#### 6.184.1.1 typedef DefType Result

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

- **generic.h**

## 6.185 TypeAtNonStrict< TypeList< T, U >, 0, DefType > Struct Template Reference

```
#include <inc/dtUtil/generic.h>
```

### Public Types

- typedef T Result

```
template<class T, class U, typename DefType> struct dtUtil::TypeAtNonStrict< TypeList< T, U >, 0, DefType >
```

### 6.185.1 Member Typedef Documentation

#### 6.185.1.1 typedef T Result

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

- generic.h

## 6.186 TypeAtNonStrict< TypeList< T, U >, i, DefType > Struct Template Reference

```
#include <inc/dtUtil/generic.h>
```

### Public Types

- typedef **TypeAtNonStrict**< U, i-1, DefType >::**Result Result**

```
template<class T, class U, unsigned int i, typename DefType> struct dtUtil::TypeAtNonStrict< TypeList< T, U >, i, DefType >
```

### 6.186.1 Member Typedef Documentation

#### 6.186.1.1 typedef **TypeAtNonStrict**<U, i - 1, DefType>::**Result Result**

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

- **generic.h**

## 6.187 TypeList< T, U > Struct Template Reference

```
#include <inc/dtUtil/generic.h>
```

### Public Types

- typedef T **Head**
- typedef U **Tail**

```
template<class T, class U> struct dtUtil::TypeList< T, U >
```

### 6.187.1 Member Typedef Documentation

#### 6.187.1.1 typedef T Head

#### 6.187.1.2 typedef U Tail

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

- **generic.h**

## 6.188 TypeTraits< U > Struct Template Reference

```
#include <inc/dtUtil/command.h>
```

### Classes

- struct **\_Traits**
- struct **\_Traits< const T & >**
- struct **\_Traits< T & >**

### Public Types

- typedef **\_Traits< U >::ConstRef ConstRef**
- typedef **\_Traits< U >::NonConstNoRef NonConstNoRef**
- typedef **\_Traits< U >::NonConstRef NonConstRef**

```
template<typename U> struct dtUtil::details::TypeTraits< U >
```

#### 6.188.1 Member Typedef Documentation

6.188.1.1 typedef **\_Traits<U>::ConstRef ConstRef**

6.188.1.2 typedef **\_Traits<U>::NonConstNoRef NonConstNoRef**

6.188.1.3 typedef **\_Traits<U>::NonConstRef NonConstRef**

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

- **command.h**

## 6.189 TypeTraits<\_Type> Struct Template Reference

```
#include <inc/dtUtil/typetraits.h>
```

### Classes

- struct `_TypeTraits_`
- struct `_TypeTraits_< const U & >`
- struct `_TypeTraits_< const U * >`
- struct `_TypeTraits_< const U *const >`
- struct `_TypeTraits_< U & >`
- struct `_TypeTraits_< U * >`
- struct `_TypeTraits_< U *const >`

### Public Types

- typedef `_TypeTraits_<_Type>::const_return_type const_param_type`
- typedef `_TypeTraits_<_Type>::const_reference const_reference`
- typedef `_TypeTraits_<_Type>::const_return_type const_return_type`
- typedef `_TypeTraits_<_Type>::return_type param_type`
- typedef `_TypeTraits_<_Type>::pointer_type pointer_type`
- typedef `_TypeTraits_<_Type>::reference reference`
- typedef `_TypeTraits_<_Type>::return_type return_type`
- typedef `_TypeTraits_<_Type>::value_type value_type`

```
template<typename _Type> struct dtUtil::TypeTraits<_Type>
```

#### 6.189.1 Member Typedef Documentation

**6.189.1.1** typedef `_TypeTraits_<_Type>::const_return_type const_param_type`

**6.189.1.2** typedef `_TypeTraits_<_Type>::const_reference const_reference`

**6.189.1.3** typedef `_TypeTraits_<_Type>::const_return_type const_return_type`

**6.189.1.4** typedef `_TypeTraits_<_Type>::return_type param_type`

**6.189.1.5** typedef `_TypeTraits_<_Type>::pointer_type pointer_type`

**6.189.1.6** typedef `_TypeTraits_<_Type>::reference reference`

**6.189.1.7** typedef `_TypeTraits_<_Type>::return_type return_type`

**6.189.1.8** typedef `_TypeTraits_<_Type>::value_type value_type`

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

- `typetraits.h`

## 6.190 Unbound Struct Reference

```
#include <inc/dtUtil/funbind.h>
```

### Static Public Member Functions

- static **ResultType MergeParms** (BoundPTL const &bound, UnboundPTL const &unbound)

```
template<int i, class BoundPTL, class UnboundPTL, class IdsTL, class TL> struct dtUtil::MergeParmsH<i, BoundPTL, UnboundPTL, IdsTL, TL >::Unbound
```

### 6.190.1 Member Function Documentation

**6.190.1.1 static ResultType MergeParms (BoundPTL const & *bound*, UnboundPTL const & *unbound*)**  
[inline, static]

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

- **funbind.h**

## 6.191 UnboundHelper< Incoming, IdsTL > Struct Template Reference

```
#include <inc/dtUtil/funbind.h>
```

### Public Types

- typedef Incoming::TypeListType IncomingTL
- typedef dtUtil::TypeAtNonStrict< UnboundTL, 0, dtUtil::NullType >::Result Parm1
- typedef dtUtil::TypeAtNonStrict< UnboundTL, 1, dtUtil::NullType >::Result Parm2
- typedef dtUtil::TypeAtNonStrict< UnboundTL, 2, dtUtil::NullType >::Result Parm3
- typedef dtUtil::TypeAtNonStrict< UnboundTL, 3, dtUtil::NullType >::Result Parm4
- typedef dtUtil::TypeAtNonStrict< UnboundTL, 4, dtUtil::NullType >::Result Parm5
- typedef UnboundTL2< IncomingTL, IdsTL >::Result UnboundTL

```
template<class Incoming, class IdsTL> struct dtUtil::UnboundHelper< Incoming, IdsTL >
```

### 6.191.1 Member Typedef Documentation

6.191.1.1 typedef Incoming::TypeListType IncomingTL

6.191.1.2 typedef dtUtil::TypeAtNonStrict<UnboundTL, 0, dtUtil::NullType>::Result Parm1

6.191.1.3 typedef dtUtil::TypeAtNonStrict<UnboundTL, 1, dtUtil::NullType>::Result Parm2

6.191.1.4 typedef dtUtil::TypeAtNonStrict<UnboundTL, 2, dtUtil::NullType>::Result Parm3

6.191.1.5 typedef dtUtil::TypeAtNonStrict<UnboundTL, 3, dtUtil::NullType>::Result Parm4

6.191.1.6 typedef dtUtil::TypeAtNonStrict<UnboundTL, 4, dtUtil::NullType>::Result Parm5

6.191.1.7 typedef UnboundTL2<IncomingTL, IdsTL>::Result UnboundTL

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

- funbind.h

## 6.192 UnboundTL2< TL, IdsTL > Struct Template Reference

```
#include <inc/dtUtil/funbind.h>
```

### Public Types

- typedef **Filter2**< TL, IdsTL, false >::Result Result

```
template<class TL, class IdsTL> struct dtUtil::UnboundTL2< TL, IdsTL >
```

### 6.192.1 Member Typedef Documentation

#### 6.192.1.1 typedef **Filter2**<TL, IdsTL, false>::Result Result

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

- **funbind.h**

## 6.193 UTMParameters Struct Reference

```
#include <inc/dtUtil/coordinates.h>
```

### Public Member Functions

- **UTMParameters** ()
- void **CalcTransverseMercatorParameters** (double a, double f, double Origin\_Latitude, double Central\_Meridian, double False\_Easting, double False\_Northing, double Scale\_Factor)
- double **DENOM** (double Latitude) const
- double **SPHSN** (double Latitude) const
- double **SPHSR** (double Latitude) const
- double **SPHTMD** (double Latitude) const

### Public Attributes

- double **TranMerc\_a**
- double **TranMerc\_ap**
- double **TranMerc\_bp**
- double **TranMerc\_cp**
- double **TranMerc\_Delta\_Easting**
- double **TranMerc\_Delta\_Northing**
- double **TranMerc\_dp**
- double **TranMerc\_ebs**
- double **TranMerc\_ep**
- double **TranMerc\_es**
- double **TranMerc\_f**
- double **TranMerc\_False\_Easting**
- double **TranMerc\_False\_Northing**
- double **TranMerc\_Origin\_Lat**
- double **TranMerc\_Origin\_Long**
- double **TranMerc\_Scale\_Factor**

## 6.193.1 Constructor & Destructor Documentation

### 6.193.1.1 UTMParameters ()

## 6.193.2 Member Function Documentation

6.193.2.1 void CalcTransverseMercatorParameters (double *a*, double *f*, double *Origin\_Latitude*, double *Central\_Meridian*, double *False\_Easting*, double *False\_Northing*, double *Scale\_Factor*)

6.193.2.2 double DENOM (double *Latitude*) const

6.193.2.3 double SPHSN (double *Latitude*) const

6.193.2.4 double SPHSR (double *Latitude*) const

6.193.2.5 double SPHTMD (double *Latitude*) const

## 6.193.3 Member Data Documentation

6.193.3.1 double TranMerc\_a

6.193.3.2 double TranMerc\_ap

6.193.3.3 double TranMerc\_bp

6.193.3.4 double TranMerc\_cp

6.193.3.5 double TranMerc\_Delta\_Easting

6.193.3.6 double TranMerc\_Delta\_Northing

6.193.3.7 double TranMerc\_dp

6.193.3.8 double TranMerc\_ebs

6.193.3.9 double TranMerc\_ep

6.193.3.10 double TranMerc\_es

6.193.3.11 double TranMerc\_f

6.193.3.12 double TranMerc\_False\_Easting

6.193.3.13 double TranMerc\_False\_Northing

6.193.3.14 double TranMerc\_Origin\_Lat

6.193.3.15 double TranMerc\_Origin\_Long

6.193.3.16 double TranMerc\_Scale\_Factor

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

- `coordinates.h`
- `coordinates.cpp`

## 6.194 VTable Struct Reference

```
#include <inc/dtUtil/functor.h>
```

### Public Attributes

- `R(* call_)(Functor const &, ParamsListType)`
- `VTable>(* clone_)(Functor const &, Functor &)`
- `void(* destroy_)(Functor const &)`

```
template<typename R, class TList, unsigned int size = 4 * sizeof(void*)> struct dtUtil::Functor< R, TList, size >::FunImplBase::VTable
```

### 6.194.1 Member Data Documentation

6.194.1.1 `R(* call_)(Functor const &, ParamsListType)`

6.194.1.2 `VTable>(* clone_)(Functor const &, Functor &)`

6.194.1.3 `void(* destroy_)(Functor const &)`

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

- `functor.h`

## 6.195 XercesErrorHandler Class Reference

Logs Xerces parsing errors.

```
#include <inc/dtUtil/xerceserrorhandler.h>
```

### Public Member Functions

- **XercesErrorHandler** ()
- **~XercesErrorHandler** ()
- virtual void **error** (const XERCES\_CPP\_NAMESPACE\_QUALIFIER SAXParseException &e)
- virtual void **fatalError** (const XERCES\_CPP\_NAMESPACE\_QUALIFIER SAXParseException &e)
- virtual void **resetErrors** ()
- virtual void **warning** (const XERCES\_CPP\_NAMESPACE\_QUALIFIER SAXParseException &e)

### 6.195.1 Detailed Description

Logs Xerces parsing errors. Used with the Xerces SAX2XMLReader. Create an instance of this class when creating SAX2XMLReader and set it as the error handler with the parser's 'setErrorHandler' method. This class is used by classes that parse XML files. It is not needed by client code.

### 6.195.2 Constructor & Destructor Documentation

6.195.2.1 **XERCES\_CPP\_NAMESPACE\_USE XercesErrorHandler** ()

6.195.2.2 **~XercesErrorHandler** ()

### 6.195.3 Member Function Documentation

6.195.3.1 **void error** (const XERCES\_CPP\_NAMESPACE\_QUALIFIER SAXParseException & e)  
[virtual]

Exceptions

**SAXParseException** Upon errors

6.195.3.2 **void fatalError** (const XERCES\_CPP\_NAMESPACE\_QUALIFIER SAXParseException & e)  
[virtual]

Exceptions

**SAXParseException** Upon fatal errors

6.195.3.3 **void resetErrors** () [virtual]

6.195.3.4 **void warning** (const XERCES\_CPP\_NAMESPACE\_QUALIFIER SAXParseException & e)  
[virtual]

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

- **xerceserrorhandler.h**
- **xerceserrorhandler.cpp**

## 6.196 XercesParser Class Reference

A class to unify the usage of Xerces parsing.

```
#include <inc/dtUtil/xercesparser.h>
```

### Public Member Functions

- **XercesParser** ()
- **~XercesParser** ()
- bool **Parse** (const std::string &data, XERCES\_CPP\_NAMESPACE\_QUALIFIER ContentHandler &handler, const std::string &schema="")

*The function that parses the file.*

### 6.196.1 Detailed Description

A class to unify the usage of Xerces parsing. This class uses the Xerces SAX2 parser. Client code must provide a Xerces ContentHandler instance, and the XML data file to be parsed.

### 6.196.2 Constructor & Destructor Documentation

6.196.2.1 **XERCES\_CPP\_NAMESPACE\_USE XercesParser** ()

6.196.2.2 **~XercesParser** ()

### 6.196.3 Member Function Documentation

6.196.3.1 **bool Parse** (const std::string & *data*, XERCES\_CPP\_NAMESPACE\_QUALIFIER ContentHandler & *handler*, const std::string & *schema* = "")

The function that parses the file. This function uses the Xerces SAX2Reader to parse a XML document. Parameters

***data*** the file to be parsed.

***handler*** the object to handle content within the file to be parsed.

***schema*** the file that defines the schema requirements.

Returns if parsing the file threw an exception, return is false. Otherwise, true.

Exceptions

***SAXParseException*** if a parsing error occurs.

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

- **xercesparser.h**
- **xercesparser.cpp**

## 6.197 XercesWriter Class Reference

A class that manages one XML DOM document.

```
#include <inc/dtUtil/xerceswriter.h>
```

### Public Member Functions

- **XercesWriter** ()  
*Initializes the xerces system.*
- void **CreateDocument** (const std::string &rootname)  
*Create a new document for the instance to use.*
- const XERCES\_CPP\_NAMESPACE\_QUALIFIER DOMDocument \* **GetDocument** () const
- XERCES\_CPP\_NAMESPACE\_QUALIFIER DOMDocument \* **GetDocument** ()
- void **WriteFile** (const std::string &file)  
*This is the call that serializes the XML Tree.*

### Protected Member Functions

- **~XercesWriter** ()  
*Does NOT destroy the xerces system.*

#### 6.197.1 Detailed Description

A class that manages one XML DOM document.

#### 6.197.2 Constructor & Destructor Documentation

##### 6.197.2.1 XercesWriter ()

Initializes the xerces system.

##### 6.197.2.2 ~XercesWriter () [protected]

Does **NOT** destroy the xerces system.

#### 6.197.3 Member Function Documentation

##### 6.197.3.1 void CreateDocument (const std::string & rootname)

Create a new document for the instance to use. Parameters

**rootname** the name of the root XML node.

##### 6.197.3.2 const XERCES\_CPP\_NAMESPACE\_QUALIFIER DOMDocument\* GetDocument () const [inline]

##### 6.197.3.3 XERCES\_CPP\_NAMESPACE\_QUALIFIER DOMDocument\* GetDocument () [inline]

##### 6.197.3.4 void WriteFile (const std::string & file)

This is the call that serializes the XML Tree. Parameters

**the** filename

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

- **xerceswriter.h**
- **xerceswriter.cpp**

## 6.198 XMLStringConverter Class Reference

Utility methods for using strings, often for XML purposes.

```
#include <inc/dtUtil/xercesutils.h>
```

### Public Member Functions

- **XMLStringConverter** (const XMLCh \*const charData)
- **~XMLStringConverter** ()
- const char \* **c\_str** () const
- const std::string **ToString** () const

### 6.198.1 Detailed Description

Utility methods for using strings, often for XML purposes. This is a simple class that lets us do easy (though not terribly efficient) transcoding of XMLCh data to local code page for display. This code was take from the xerces-c 2.6 samples

It's main reason for existing is to allow short and quick translations for printing out debugging info.

### 6.198.2 Constructor & Destructor Documentation

**6.198.2.1 XMLStringConverter (const XMLCh \*const *charData*) [inline]**

**6.198.2.2 ~XMLStringConverter () [inline]**

### 6.198.3 Member Function Documentation

**6.198.3.1 const char\* c\_str () const [inline]**

Returns the XMLCh string as a char\*

**6.198.3.2 const std::string ToString () const [inline]**

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

- **xercesutils.h**



## File Documentation

---

### 7.1 barycentric.h File Reference

#### Classes

- class **BarycentricSpace**< **VecT** >  
*Transforms Cartesian data points into Barycentric coordinate systems.*

#### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*

#### Functions

- template<class V >  
V **CalculateBarycentricCenter** (const V &a, const V &b, const V &c)

## 7.2 bits.h File Reference

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*
- namespace **dtUtil::Bits**  
*Contains bit-wise operation functionality which makes using bits a little "bit" easier.*

### Functions

- template<class N , class B >  
**N Add** (N number, B bits)  
*Add the "bits" to "number".*
- template<class N , class B >  
**bool Has** (N number, B bits)  
*See if the "bits" are in "number".*
- template<class N , class B >  
**N Remove** (N number, B bits)  
*Remove the "bits" from "number".*
- template<class N , class B >  
**N Toggle** (N number, B bits)  
*Toggle the "bits" in "number".*

## 7.3 boundingshapeutils.h File Reference

```
#include <osg/NodeVisitor>
#include <osg/BoundingBox>
#include <osg/Geode>
#include <osg/MatrixTransform>
#include <osg/Version>
```

### Classes

- class **BoundingBoxVisitor**

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

## 7.4 breakoverride.h File Reference

### Classes

- struct **BreakOverride**  
*This macro should be used when deprecating (or removing) virtual functions.*

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*

### Defines

- #define **BREAK\_OVERRIDE(f)**

#### 7.4.1 Define Documentation

##### 7.4.1.1 #define BREAK\_OVERRIDE(f)

###### Value:

```
virtual dtUtil::BreakOverride f\  
{\  
    return dtUtil::BreakOverride();\  
}
```

## 7.5 collectorutil.h File Reference

```
#include <string>
```

### Classes

- struct **GetElementType**< T >  
*Template Function that is used to get the RefPtr's Element type from the map.*

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*
- namespace **dtUtil::CollectorUtil**

### Functions

- template<class mapType >  
bool **AddNode** (const std::string &nodeName, typename GetElementType< mapType >::value\_type \*nodeType, mapType &nodeMap)  
*Template function that is used to Add a unique node or object to its respective map.*
- template<class mapType >  
GetElementType< mapType >::value\_type \* **FindNodePointer** (const std::string &nodeName, mapType &nodeMap)  
*Template function that is used to find a node or object that is inside a map.*
- template<class mapType >  
const GetElementType< mapType >::value\_type \* **FindNodePointer** (const std::string &nodeName, const mapType &nodeMap)  
*Template function that is used to find a node or object that is inside a map.*
- template<class mapType >  
bool **RemoveNode** (const std::string &nodeName, mapType &nodeMap)  
*Template function that is used to Remove a unique node or object from its respective map.*

## 7.6 command.h File Reference

```
#include <osg/Referenced>
#include <dtUtil/functor.h>
#include <dtUtil/generic.h>
```

### Classes

- struct **\_Traits**< **T** >
- struct **\_Traits**< **const T &** >
- struct **\_Traits**< **T &** >
- class **Command**< **RetT** >  
*An abstract class for all types which provide a uniform interface for executing Functors.*
- class **Command0**< **RetT** >  
*A **Command** (p. 70) which does not need arguments and has no return value.*
- class **Command1**< **RetT, ArgT** >  
*A **Command** (p. 70) which uses one argument and has no return value.*
- class **Command2**< **RetT, ArgTMember1, ArgTMember2** >  
*A **Command** (p. 70) which uses two arguments and has no return value.*
- struct **TypeTraits**< **U** >

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*
- namespace **dtUtil::details**

## 7.7 configproperties.h File Reference

```
#include <string>
```

### Classes

- class **ConfigProperties**  
*Interface for having configuration properties.*

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*

## 7.8 coordinates.cpp File Reference

```
#include <prefix/dtutilprefix-src.h>
#include <cmath>
#include <cstdio>
#include <float>
#include <dtUtil/macros.h>
#include <dtUtil/matrixutil.h>
#include <dtUtil/coordinates.h>
#include <dtUtil/log.h>
#include <dtUtil/stringutils.h>
#include <dtUtil/mathdefines.h>
```

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

## 7.9 coordinates.h File Reference

```
#include <map>
#include <set>
#include <string>
#include <osg/Matrix>
#include <osg/Math>
#include <osg/Vec3>
#include <osg/Vec3d>
#include <osg/Vec3f>
#include <dtUtil/export.h>
#include <dtUtil/enumeration.h>
#include <dtUtil/exception.h>
#include <dtUtil/mathdefines.h>
```

### Classes

- class **CoordinateConversionExceptionEnum**
- class **Coordinates**
- class **IncomingCoordinateType**
- class **LocalCoordinateType**
- struct **UTMParameters**

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

### Functions

- template<typename T >  
T **safeASIN** (T x)

### Variables

- const double **AD\_C** = 1.0026000
- const double **CentralMeridianScale** = 0.9996  
*Scale used in UTM calculations.*
- const double **COS\_67P5** = 0.38268343236508977
- const double **flatteningReciprocal** = 298.257223563  
*The reciprocal of the flattening parameter (WGS 84).*
- const double **Geocent\_a** = semiMajorAxis
- const double **Geocent\_e2** = (2.0 - Geocent\_f) \* Geocent\_f
- const double **Geocent\_e2\_2** = Geocent\_e2 \* Geocent\_e2
- const double **Geocent\_e2\_3** = Geocent\_e2\_2 \* Geocent\_e2
- const double **Geocent\_ef** = Geocent\_f / (2.0 - Geocent\_f)
- const double **Geocent\_ef\_3** = Geocent\_ef \* Geocent\_ef \* Geocent\_ef
- const double **Geocent\_ef\_4** = Geocent\_ef\_3 \* Geocent\_ef
- const double **Geocent\_ep2** = Geocent\_e2 / (1.0 - Geocent\_e2)
- const double **Geocent\_f** = 1 / flatteningReciprocal
- const double **MagneticNorthLatitude** = 82.116

- const double **MagneticNorthLongitude** = 114.0666
- const double **MAX\_DELTA\_LONG** = ((osg::PI \* 90)/180.0)
- const double **MAX\_EASTING** = 900000
- const double **MAX\_LAT** = ( (84.5 \* osg::PI) / 180.0 )
- const double **MAX\_NORTHING** = 10000000
- const double **MAX\_SCALE\_FACTOR** = 3.0
- const double **METERS\_PER\_DEGREE** = 1852.0\*60
- const double **MIN\_EASTING** = 100000
- const double **MIN\_LAT** = ( (-80.5 \* osg::PI) / 180.0 )
- const double **MIN\_NORTHING** = 0
- const double **MIN\_SCALE\_FACTOR** = 0.3
- const double **semiMajorAxis** = 6378137.0

*The length of the semi-major axis, in meters (WGS 84).*

## 7.10 datastream.cpp File Reference

```
#include <prefix/dtutilprefix-src.h>
#include <climits>
#include <osg/Endian>
#include <dtUtil/exception.h>
#include <dtUtil/datastream.h>
#include <cstring>
```

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

### Variables

- static const char \* **LOGNAME** = "DataStream"

## 7.11 datastream.h File Reference

```
#include <string>
#include <osg/Vec2>
#include <osg/Vec3>
#include <osg/Vec4>
#include <osg/Vec2f>
#include <osg/Vec3f>
#include <osg/Vec4f>
#include <osg/Vec2d>
#include <osg/Vec3d>
#include <osg/Vec4d>
#include <dtUtil/enumeration.h>
#include <dtUtil/export.h>
```

### Classes

- class **DataStream**
- class **DataStreamException**
- class **SeekTypeEnum**

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*

## 7.12 datetime.cpp File Reference

```
#include <prefix/dtutilprefix-src.h>
#include <dtUtil/datetime.h>
#include <dtUtil/macros.h>
#include <cmath>
#include <cstring>
```

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

### Functions

- **IMPLEMENT\_ENUM** (DateTime::TimeFormat)
- **IMPLEMENT\_ENUM** (DateTime::TimeType)
- **IMPLEMENT\_ENUM** (DateTime::TimeOrigin)

## 7.13 datetime.h File Reference

```
#include <dtUtil/export.h>
#include <dtUtil/enumeration.h>
#include <osg/Referenced>
#include <string>
#include <ctime>
```

### Classes

- class **DateTime**
- class **TimeFormat**  
*The **TimeFormat** (p. 278) enumeration is used to specify how to map a **DateTime** (p. 96) object to a string.*
- class **TimeOrigin**  
*The **TimeOrigin** (p. 279) enumeration determines how the instance of **DateTime** (p. 96) should be interpreted.*
- class **TimeType**  
*The **TimeType** (p. 280) enumeration is used to identify the purpose of a **DateTime** (p. 96) instance.*

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*

## 7.14 deprecationmgr.cpp File Reference

```
#include <dtUtil/deprecationmgr.h>
#include <csignal>
#include <iostream>
#include <sstream>
#include <map>
#include <set>
```

### Classes

- struct **DeprecatedFunction**
- class **DeprecationMgrImpl**

## 7.15 deprecationmgr.h File Reference

```
#include <dtUtil/macros.h>
```

```
#include <dtUtil/export.h>
```

### Classes

- class **DeprecationMgr**  
*Used to deprecate a method.*

### Defines

- #define **DEPRECATE**(a, b)

#### 7.15.1 Define Documentation

##### 7.15.1.1 #define DEPRECATE(a, b)

## 7.16 dtutil.h File Reference

```
#include "dtUtil/deprecationmgr.h"
#include "dtUtil/enumeration.h"
#include "dtUtil/fractal.h"
#include "dtUtil/fileutils.h"
#include "dtUtil/keyframedecoder.h"
#include "dtUtil/log.h"
#include "dtUtil/mathdefines.h"
#include "dtUtil/matrixutil.h"
#include "dtUtil/noise1.h"
#include "dtUtil/noise2.h"
#include "dtUtil/noise3.h"
#include "dtUtil/noisetexture.h"
#include "dtUtil/noiseutility.h"
#include "dtUtil/objectfactory.h"
#include "dtUtil/polardecomp.h"
#include "dtUtil/seamlessnoise.h"
#include "dtUtil/serializer.h"
#include "dtUtil/stringutils.h"
#include "dtUtil/xerceserrorhandler.h"
#include "dtUtil/xerceswriter.h"
#include "dtUtil/xercesparser.h"
#include "dtUtil/xercesutils.h"
```

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

## 7.17 enumeration.cpp File Reference

```
#include <dtUtil/enumeration.h>
```

```
#include <ostream>
```

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

### Functions

- `std::ostream & operator<< (std::ostream &os, const Enumeration &e)`

*Helper method to print enumerations to an output stream.*

## 7.18 enumeration.h File Reference

```
#include <string>
#include <vector>
#include <iosfwd>
#include <dtUtil/export.h>
```

### Classes

- class **Enumeration**  
*This class represents a type-safe enumeration pattern.*

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*

### Defines

- #define **DECLARE\_ENUM**(EnumType)  
*Helper macros used to create the static data and methods needed to enumerate an enumeration.*
- #define **IMPLEMENT\_ENUM**(EnumType)

### Functions

- `std::ostream & operator<<` (`std::ostream &os`, `const Enumeration &e`)  
*Helper method to print enumerations to an output stream.*

#### 7.18.1 Define Documentation

##### 7.18.1.1 #define DECLARE\_ENUM(EnumType)

###### Value:

```
private:
    static std::vector<EnumType*> mInstances;
    static std::vector<dtUtil::Enumeration*> mGenericInstances;
    static void AddInstance(EnumType* instance)
    {
        EnumType::mInstances.push_back(instance);
        EnumType::mGenericInstances.push_back(instance);
    }
public:
    static const std::vector<EnumType*>& EnumerateType()
    {
        return EnumType::mInstances;
    }

    static const std::vector<dtUtil::Enumeration*>& Enumerate()
    {
        return EnumType::mGenericInstances;
    }

    static EnumType* GetValueForName(const std::string& name)
    {
        for (unsigned i = 0; i < mInstances.size(); ++i)
        {
            if (name == mInstances[i]->GetName())
            {
                return mInstances[i];
            }
        }
    }
```

```
    }                                     \
    return NULL;                          \
}
```

Helper macros used to create the static data and methods needed to enumerate an enumeration.

#### 7.18.1.2 #define IMPLEMENT\_ENUM(EnumType)

##### Value:

```
std::vector<EnumType*> EnumType::mInstances;      \
std::vector<dtUtil::Enumeration*> EnumType::mGenericInstances;
```

## 7.19 exception.cpp File Reference

```
#include <prefix/dtutilprefix-src.h>
#include <sstream>
#include <dtUtil/exception.h>
#include <dtUtil/log.h>
#include <iostream>
```

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

### Functions

- **IMPLEMENT\_ENUM** (BaseExceptionType)
- **DT\_UTIL\_EXPORT** std::ostream & **operator**<< (std::ostream &o, const Exception &ex)

## 7.20 exception.h File Reference

```
#include <string>
#include <iosfwd>
#include <dtUtil/export.h>
#include <dtUtil/enumeration.h>
#include <dtUtil/log.h>
```

### Classes

- class **BaseExceptionType**
- class **Exception**

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*

### Functions

- DT\_UTIL\_EXPORT std::ostream & **operator**<< (std::ostream &o, const Exception &ex)

## 7.21 export.h File Reference

### Defines

- #define DT\_UTIL\_EXPORT

#### 7.21.1 Define Documentation

##### 7.21.1.1 #define DT\_UTIL\_EXPORT

## 7.22 fileutils.cpp File Reference

```
#include <prefix/dtutilprefix-src.h>
#include <dtUtil/macros.h>
#include <osg/Version>
#include <algorithm>
#include <sys/param.h>
#include <errno.h>
#include <osg/Notify>
#include <stack>
#include <dtUtil/fileutils.h>
#include <dtUtil/exception.h>
#include <dtUtil/stringutils.h>
#include <dtUtil/log.h>
#include <dtCore/globals.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <iostream>
#include <osgDB/FileUtils>
#include <osgDB/FileNameUtils>
#include <queue>
```

### Classes

- class **ToLowerClass**

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

### Defines

- #define **MAX\_PATH** 1024
- #define **S\_ISDIR(x)** (((x) & S\_IFMT) == S\_IFDIR)
- #define **S\_ISREG(x)** (((x) & S\_IFMT) == S\_IFREG)
- #define **stat64** stat

### Functions

- std::string **getFileExtensionIncludingDot** (const std::string &fileName)
- bool **iMakeDirectory** (const std::string &path)

#### 7.22.1 Define Documentation

**7.22.1.1 #define MAX\_PATH 1024**

**7.22.1.2 #define S\_ISDIR(x) (((x) & S\_IFMT) == S\_IFDIR)**

**7.22.1.3 #define S\_ISREG(x) (((x) & S\_IFMT) == S\_IFREG)**

**7.22.1.4 #define stat64 stat**

## 7.23 fileutils.h File Reference

```
#include <string>
#include <vector>
#include <dtCore/refptr.h>
#include <osg/Referenced>
#include <dtUtil/enumeration.h>
#include <dtUtil/export.h>
```

### Classes

- class **FileExceptionEnum**  
*An enumeration of exception types that will be thrown by fileutils.*
- struct **FileInfo**
- class **FileUtils**

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*

### Typedefs

- typedef std::vector< std::string > **DirectoryContents**
- typedef std::vector< std::string > **FileExtensionList**

### Enumerations

- enum **FileType** { **FILE\_NOT\_FOUND**, **REGULAR\_FILE**, **DIRECTORY** }

## 7.24 fractal.h File Reference

### Classes

- class **Fractal**< **Real**, **Vector**, **Noise** >

***Fractal** (p. 128): This class is made to complement the noise classes where as the noise class hashes vectors to floats between -1 to 1 this class can be used to make summations of the noise to use this class I recommend including **NoiseUtility.h** (p. 372) and using the appropriate typedefs.*

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

## 7.25 funbind.h File Reference

```
#include <dtUtil/functor.h>
```

### Classes

- class **Binder**< Incoming, BoundIdsTL >
- struct **Bound**
- struct **BoundHelper**< Incoming, IdsTL >
- struct **BoundTL2**< TL, IdsTL >
- struct **Filter2**< TL, IdsTL, include >
- struct **MergeParmsH**< i, BoundPTL, UnboundPTL, IdsTL, TL >
- struct **MergeParmsH**< 0, BoundPTL, UnboundPTL, dtUtil::NullType, TL >
- struct **MergeParmsH**< i, BoundPTL, UnboundPTL, IdsTL, dtUtil::NullType >
- struct **Predicate**< j, IdsTL2 >
- struct **Predicate**< j, dtUtil::NullType >
- struct **Temp**< TL2, i, IdsTL2, Predicate >
- struct **Temp**< dtUtil::NullType, i, dtUtil::NullType, Predicate >
- struct **Temp**< dtUtil::NullType, i, IdsTL2, Predicate >
- struct **Temp**< TL2, 0, dtUtil::NullType, dtUtil::IsIntType >
- struct **Temp**< TL2, 0, dtUtil::NullType, dtUtil::NotIntType >
- struct **Temp**< TL2, 0, typename dtUtil::IdsFromTL< TL2 >::Type, dtUtil::IsIntType >
- struct **Temp**< TL2, 0, typename dtUtil::IdsFromTL< TL2 >::Type, dtUtil::NotIntType >
- struct **Temp**< TL2, i, dtUtil::NullType, Predicate >
- struct **Unbound**
- struct **UnboundHelper**< Incoming, IdsTL >
- struct **UnboundTL2**< TL, IdsTL >

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

### Functions

- template<int i1, int i2, int i3, int i4, int i5, int i6, int i7, class Incoming >  
Binder< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5, i6, i7 >::Type >::Outgoing **Bind** (Incoming const &fun, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5, i6, i7 >::Type >::Parm1 p1, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5, i6, i7 >::Type >::Parm2 p2, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5, i6, i7 >::Type >::Parm3 p3, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5, i6, i7 >::Type >::Parm4 p4, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5, i6, i7 >::Type >::Parm5 p5, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5, i6, i7 >::Type >::Parm5 p6, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5, i6, i7 >::Type >::Parm5 p7)
- template<int i1, int i2, int i3, int i4, int i5, int i6, class Incoming >  
Binder< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5, i6 >::Type >::Outgoing **Bind** (Incoming const &fun, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5, i6 >::Type >::Parm1 p1, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5, i6 >::Type >::Parm2 p2, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5, i6 >::Type >::Parm3 p3, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5, i6 >::Type >::Parm4 p4, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5, i6 >::Type >::Parm5 p5, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5, i6 >::Type >::Parm5 p6)

- `template<int i1, int i2, int i3, int i4, int i5, class Incoming >`  
`Binder< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5 >::Type >::Outgoing Bind` (Incoming const &fun, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5 >::Type >::Parm1 p1, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5 >::Type >::Parm2 p2, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5 >::Type >::Parm3 p3, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5 >::Type >::Parm4 p4, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4, i5 >::Type >::Parm5 p5)
- `template<int i1, int i2, int i3, int i4, class Incoming >`  
`Binder< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4 >::Type >::Outgoing Bind` (Incoming const &fun, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4 >::Type >::Parm1 p1, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4 >::Type >::Parm2 p2, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4 >::Type >::Parm3 p3, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3, i4 >::Type >::Parm4 p4)
- `template<int i1, int i2, int i3, class Incoming >`  
`Binder< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3 >::Type >::Outgoing Bind` (Incoming const &fun, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3 >::Type >::Parm1 p1, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3 >::Type >::Parm2 p2, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2, i3 >::Type >::Parm3 p3)
- `template<int i1, int i2, class Incoming >`  
`Binder< Incoming, typename dtUtil::CreateldsTL< i1, i2 >::Type >::Outgoing Bind` (Incoming const &fun, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2 >::Type >::Parm1 p1, typename BoundHelper< Incoming, typename dtUtil::CreateldsTL< i1, i2 >::Type >::Parm2 p2)
- `template<int i1, class Incoming >`  
`Binder< Incoming, typename dtUtil::CreateldsTL< i1 >::Type >::Outgoing Bind` (Incoming const &fun, typename BoundHelper< Incoming, typename dtUtil::CreateTL< dtUtil::Int2Type< i1 > >::Type >::Parm1 p1)
- `template<class Incoming >`  
`Binder< Incoming, typename dtUtil::CreateldsTL<>::Type >::Outgoing Bind` (Incoming const &fun)
- `template<class TL , class IdsTL , class BoundPTL , class UnboundPTL >`  
`CallParms< TL >::ParmsListType MergeParms` (BoundPTL const &bound, UnboundPTL const &unbound)

## 7.26 funcall.h File Reference

```
#include <dtUtil/generic.h>
```

### Classes

- struct **CallParms**< TYPELIST\_0(>
- struct **CallParms**< TYPELIST\_1(P1)>
- struct **CallParms**< TYPELIST\_2(P1, P2)>
- struct **CallParms**< TYPELIST\_3(P1, P2, P3)>
- struct **CallParms**< TYPELIST\_4(P1, P2, P3, P4)>
- struct **CallParms**< TYPELIST\_5(P1, P2, P3, P4, P5)>
- struct **CallParms**< TYPELIST\_6(P1, P2, P3, P4, P5, P6)>
- struct **CallParms**< TYPELIST\_7(P1, P2, P3, P4, P5, P6, P7)>
- struct **FunctorCall**< CallType, R, TYPELIST\_0(>
- struct **FunctorCall**< CallType, R, TYPELIST\_1(P1)>
- struct **FunctorCall**< CallType, R, TYPELIST\_2(P1, P2)>
- struct **FunctorCall**< CallType, R, TYPELIST\_3(P1, P2, P3)>
- struct **FunctorCall**< CallType, R, TYPELIST\_4(P1, P2, P3, P4)>
- struct **FunctorCall**< CallType, R, TYPELIST\_5(P1, P2, P3, P4, P5)>
- struct **FunctorCall**< CallType, R, TYPELIST\_6(P1, P2, P3, P4, P5, P6)>
- struct **FunctorCall**< CallType, R, TYPELIST\_7(P1, P2, P3, P4, P5, P6, P7)>

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*

## 7.27 functor.h File Reference

```
#include <dtUtil/generic.h>
#include <dtUtil/funtraits.h>
#include <dtUtil/funcall.h>
#include <cstddef>
#include <utility>
```

### Classes

- struct **ByValue**< T >
- class **Functor**< R, TList, size >
- struct **FunctorImpl**< T >
- struct **FunImplBase**
- struct **FunStorageImpl**< V, Derived >
- struct **MemberFnImpl**< P, T >
- struct **NewAlloc**< T >
- struct **SelectStored**< T >
- struct **Stored**
- struct **Typeless**
- struct **VTable**

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*

### Defines

- #define **DoCall**(parms) return vptr\_->call\_(\*this, parms);

### Functions

- template<typename CallType , class Fun >  
 Functor< typename dtUtil::FunTraits< CallType >::ResultType, typename dtUtil::FunTraits< CallType >::TypeListType > **MakeFunctor** (Fun const &fun)
- template<typename CallType , class PObj >  
 Functor< typename dtUtil::FunTraits< CallType >::ResultType, typename dtUtil::FunTraits< CallType >::TypeListType > **MakeFunctor** (CallType memfun, PObj const &pobj)
- template<typename CallType >  
 Functor< typename dtUtil::FunTraits< CallType >::ResultType, typename dtUtil::FunTraits< CallType >::TypeListType > **MakeFunctor** (CallType fun)

#### 7.27.1 Define Documentation

##### 7.27.1.1 #define DoCall(parms) return vptr\_->call\_(\*this, parms);

## 7.28 funtraits.h File Reference

```
#include <dtUtil/generic.h>
```

### Classes

- struct **FunTraits**< R(\*)()>
- struct **FunTraits**< R(\*)(**P1**)>
- struct **FunTraits**< R(\*)(**P1**, **P2**)>
- struct **FunTraits**< R(\*)(**P1**, **P2**, **P3**)>
- struct **FunTraits**< R(\*)(**P1**, **P2**, **P3**, **P4**)>
- struct **FunTraits**< R(\*)(**P1**, **P2**, **P3**, **P4**, **P5**)>
- struct **FunTraits**< R(\*)(**P1**, **P2**, **P3**, **P4**, **P5**, **P6**)>
- struct **FunTraits**< R(\*)(**P1**, **P2**, **P3**, **P4**, **P5**, **P6**, **P7**)>
- struct **FunTraits**< R(O::\*)(**P1**) const >
- struct **FunTraits**< R(O::\*)(**P1**)>
- struct **FunTraits**< R(O::\*)(**P1**) const >
- struct **FunTraits**< R(O::\*)(**P1**)>
- struct **FunTraits**< R(O::\*)(**P1**, **P2**) const >
- struct **FunTraits**< R(O::\*)(**P1**, **P2**)>
- struct **FunTraits**< R(O::\*)(**P1**, **P2**, **P3**) const >
- struct **FunTraits**< R(O::\*)(**P1**, **P2**, **P3**)>
- struct **FunTraits**< R(O::\*)(**P1**, **P2**, **P3**, **P4**) const >
- struct **FunTraits**< R(O::\*)(**P1**, **P2**, **P3**, **P4**)>
- struct **FunTraits**< R(O::\*)(**P1**, **P2**, **P3**, **P4**, **P5**) const >
- struct **FunTraits**< R(O::\*)(**P1**, **P2**, **P3**, **P4**, **P5**)>
- struct **FunTraits**< R(O::\*)(**P1**, **P2**, **P3**, **P4**, **P5**, **P6**) const >
- struct **FunTraits**< R(O::\*)(**P1**, **P2**, **P3**, **P4**, **P5**, **P6**)>
- struct **FunTraits**< R(O::\*)(**P1**, **P2**, **P3**, **P4**, **P5**, **P6**, **P7**) const >
- struct **FunTraits**< R(O::\*)(**P1**, **P2**, **P3**, **P4**, **P5**, **P6**, **P7**)>

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

## 7.29 generic.h File Reference

### Classes

- struct **AppendTL**< TList, T >
- struct **AppendTL**< NullType, T >
- struct **CreatesTL**< i1, i2, i3, i4, i5, i6, i7, i8 >
- struct **CreatesTL**<-1,-1,-1,-1,-1,-1,-1,-1 >
- struct **CreateTL**< T1, T2, T3, T4, T5, T6, T7, T8 >
- struct **CreateTL**< NullType, NullType, NullType, NullType, NullType, NullType, NullType, NullType >
- struct **EmptyType**
- struct **IdsFromTL**< TL, i >
- struct **IdsFromTL**< NullType, i >
- struct **InstantiateH**< NullType, Holder, i >
- struct **InstantiateH**< TypeList< T, U >, Holder, i >
- struct **InstantiateHAccessor**< 0, InstantiateH< TypeList< T, U >, Holder, i >, i >
- struct **InstantiateHAccessor**< j, InstantiateH< TypeList< T, U >, Holder, i >, i >
- struct **Int2Type**< v >
- struct **IsIntType**< T, i >
- struct **IsIntType**< Int2Type< i >, i >
- struct **Length**< NullType >
- struct **Length**< TypeList< T, U > >
- struct **NotIntType**< T, i >
- class **NullType**
- struct **Select**< flag, T, U >
- struct **Select**< false, T, U >
- struct **TailAt**< InstantiateH< TypeList< T, U >, Holder, i >, 0, i >
- struct **TailAt**< InstantiateH< TypeList< T, U >, Holder, i >, j, i >
- struct **TupleHolder**< T, i >
- struct **TypeAt**< TypeList< T, U >, 0 >
- struct **TypeAt**< TypeList< T, U >, i >
- struct **TypeAtNonStrict**< TList, i, DefType >
- struct **TypeAtNonStrict**< TypeList< T, U >, 0, DefType >
- struct **TypeAtNonStrict**< TypeList< T, U >, i, DefType >
- struct **TypeList**< T, U >

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

### Defines

- #define **TYPELIST\_0**() dtUtil::NullType
- #define **TYPELIST\_1**(T1) dtUtil::TypeList<T1, dtUtil::NullType>
- #define **TYPELIST\_2**(T1, T2) dtUtil::TypeList<T1, TYPELIST\_1(T2) >
- #define **TYPELIST\_3**(T1, T2, T3) dtUtil::TypeList<T1, TYPELIST\_2(T2, T3) >
- #define **TYPELIST\_4**(T1, T2, T3, T4) dtUtil::TypeList<T1, TYPELIST\_3(T2, T3, T4) >
- #define **TYPELIST\_5**(T1, T2, T3, T4, T5) dtUtil::TypeList<T1, TYPELIST\_4(T2, T3, T4, T5) >
- #define **TYPELIST\_6**(T1, T2, T3, T4, T5, T6) dtUtil::TypeList<T1, TYPELIST\_5(T2, T3, T4, T5, T6) >
- #define **TYPELIST\_7**(T1, T2, T3, T4, T5, T6, T7) dtUtil::TypeList<T1, TYPELIST\_6(T2, T3, T4, T5, T6, T7) >
- #define **TYPELIST\_8**(T1, T2, T3, T4, T5, T6, T7, T8) dtUtil::TypeList<T1, TYPELIST\_7(T2, T3, T4, T5, T6, T7, T8) >

## Functions

- `template<unsigned int j, class Instantiated >`  
`InstantiateHAccessor<j, Instantiated, Instantiated::ordern >::TargetHolder const & GetH (Instantiated const &h)`
- `template<unsigned int j, class Instantiated >`  
`InstantiateHAccessor<j, Instantiated, Instantiated::ordern >::TargetHolder & GetH (Instantiated &h)`

### 7.29.1 Define Documentation

7.29.1.1 `#define TYPELIST_0() dtUtil::NullType`

7.29.1.2 `#define TYPELIST_1(T1) dtUtil::TypeList<T1, dtUtil::NullType>`

7.29.1.3 `#define TYPELIST_2(T1, T2) dtUtil::TypeList<T1, TYPELIST_1(T2) >`

7.29.1.4 `#define TYPELIST_3(T1, T2, T3) dtUtil::TypeList<T1, TYPELIST_2(T2, T3) >`

7.29.1.5 `#define TYPELIST_4(T1, T2, T3, T4) dtUtil::TypeList<T1, TYPELIST_3(T2, T3, T4) >`

7.29.1.6 `#define TYPELIST_5(T1, T2, T3, T4, T5) dtUtil::TypeList<T1, TYPELIST_4(T2, T3, T4, T5) >`

7.29.1.7 `#define TYPELIST_6(T1, T2, T3, T4, T5, T6) dtUtil::TypeList<T1, TYPELIST_5(T2, T3, T4, T5, T6) >`

7.29.1.8 `#define TYPELIST_7(T1, T2, T3, T4, T5, T6, T7) dtUtil::TypeList<T1, TYPELIST_6(T2, T3, T4, T5, T6, T7) >`

7.29.1.9 `#define TYPELIST_8(T1, T2, T3, T4, T5, T6, T7, T8) dtUtil::TypeList<T1, TYPELIST_7(T2, T3, T4, T5, T6, T7, T8) >`

## 7.30 geometrycollector.h File Reference

```
#include <osg/NodeVisitor>
```

```
#include <osg/Geode>
```

### Classes

- class **GeometryCollector**

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

## 7.31 hotspotdefinition.h File Reference

```
#include <string>
#include <osg/Vec3>
#include <osg/Quat>
#include <dtUtil/export.h>
```

### Classes

- struct **HotSpotDefinition**

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

## 7.32 hotspotxml.cpp File Reference

```
#include <prefix/dtutilprefix-src.h>
#include <dtUtil/hotspotxml.h>
#include <dtUtil/xercesutils.h>
#include <cstdlib>
#include <sstream>
```

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

## 7.33 hotspotxml.h File Reference

```
#include <xercesc/sax2/ContentHandler.hpp>
#include <vector>
#include <stack>
#include <dtUtil/hotspotdefinition.h>
#include <dtUtil/export.h>
#include <osg/Vec3>
```

### Classes

- class **HotSpotFileHandler**

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

## 7.34 kdtree.h File Reference

Defines the interface for the KDTree class. #include <vector>

```
#include <algorithm>
```

```
#include <cmath>
```

```
#include <cstddef>
```

```
#include <cassert>
```

### Classes

- class **\_Alloc\_base**< \_Tp, \_Alloc >
- class **\_Base\_iterator**
- struct **\_Bracket\_accessor**< \_Val >
- class **\_Iterator**< \_Val, \_Ref, \_Ptr >
- struct **\_Node**< \_Val >
- struct **\_Node\_base**
- class **\_Node\_compare**< \_Val, \_Acc, \_Cmp >
- struct **\_Region**< \_\_K, \_Val, \_SubVal, \_Acc, \_Cmp >
- struct **always\_true**< \_Tp >
- class **KDTree**< \_\_K, \_Val, \_Acc, \_Dist, \_Cmp, \_Alloc >
- class **NoLeakAlloc**
- struct **squared\_difference**< \_Tp, \_Dist >
- struct **squared\_difference\_counted**< \_Tp, \_Dist >

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

### Defines

- #define **KDTREE\_LIB\_VERSION** "0\_7\_1"
- #define **KDTREE\_VERSION** 701

### Functions

- template<typename \_ValA , typename \_ValB , typename \_Dist , typename \_Acc >  
**\_Dist::distance\_type \_S\_accumulate\_node\_distance** (const size\_t \_\_dim, const \_Dist &\_\_dist, const \_Acc &\_\_acc, const \_ValA &\_\_a, const \_ValB &\_\_b)
- template<typename \_ValA , typename \_ValB , typename \_Cmp , typename \_Acc >  
**bool \_S\_node\_compare** (const size\_t \_\_dim, const \_Cmp &\_\_cmp, const \_Acc &\_\_acc, const \_ValA &\_\_a, const \_ValB &\_\_b)
- template<typename \_Val , typename \_Cmp , typename \_Acc , typename NodeType >  
**NodeType \* \_S\_node\_descend** (const size\_t \_\_dim, const \_Cmp &\_\_cmp, const \_Acc &\_\_acc, const \_Val &\_\_val, const NodeType \*\_\_node)
- template<typename \_ValA , typename \_ValB , typename \_Dist , typename \_Acc >  
**\_Dist::distance\_type \_S\_node\_distance** (const size\_t \_\_dim, const \_Dist &\_\_dist, const \_Acc &\_\_acc, const \_ValA &\_\_a, const \_ValB &\_\_b)
- template<class SearchVal , typename NodeType , typename \_Cmp , typename \_Acc , typename \_Dist , typename \_Predicate >  
**std::pair< const NodeType \*, std::pair< size\_t, typename \_Dist::distance\_type > > \_S\_node\_nearest** (const size\_t \_\_k, size\_t \_\_dim, SearchVal const &\_\_val, const NodeType \*\_\_node, const \_Node\_base \*\_\_end, const NodeType \*\_\_best, typename \_Dist::distance\_type \_\_max, const \_Cmp &\_\_cmp, const \_Acc &\_\_acc, const \_Dist &\_\_dist, \_Predicate \_\_p)
- template<typename \_Val >  
**bool operator!=** (\_Iterator< \_Val, \_Val &, \_Val \* > const &, \_Iterator< \_Val, const \_Val &, const \_Val \* > const &)

- `template<typename _Val >`  
`bool operator!= (_Iterator< _Val, const _Val &, const _Val * > const &, _Iterator< _Val, _Val &, _Val * > const &)`
- `template<typename _Val, typename _Ref, typename _Ptr >`  
`bool operator!= (_Iterator< _Val, _Ref, _Ptr > const &, _Iterator< _Val, _Ref, _Ptr > const &)`
- `template<typename _Val >`  
`bool operator== (_Iterator< _Val, _Val &, _Val * > const &, _Iterator< _Val, const _Val &, const _Val * > const &)`
- `template<typename _Val >`  
`bool operator== (_Iterator< _Val, const _Val &, const _Val * > const &, _Iterator< _Val, _Val &, _Val * > const &)`
- `template<typename _Val, typename _Ref, typename _Ptr >`  
`bool operator== (_Iterator< _Val, _Ref, _Ptr > const &, _Iterator< _Val, _Ref, _Ptr > const &)`

### 7.34.1 Detailed Description

Defines the interface for the KDTree class. Author Martin F. Krafft <libkdtree@pobox.madduck.net> Paul Harris figured this stuff out (below) Notes: This is similar to a binary tree, but its not the same. There are a few important differences:

\* Each level is sorted by a different criteria (this is fundamental to the design).

\* It is possible to have children IDENTICAL to its parent in BOTH branches This is different to a binary tree, where identical children are always to the right So, KDTree has the relationships: \* The left branch is <= its parent (in binary tree, this relationship is a plain < ) \* The right branch is <= its parent (same as binary tree)

This is done for mostly for performance. Its a LOT easier to maintain a consistent tree if we use the <= relationship. Note that this relationship only makes a difference when searching for an exact item with find() or find\_exact, other search, erase and insert functions don't notice the difference.

In the case of binary trees, you can safely assume that the next identical item will be the child leaf, but in the case of KDTree, the next identical item might be a long way down a subtree, because of the various different sort criteria.

So erase()ing a node from a KDTree could require serious and complicated tree rebalancing to maintain consistency... IF we required binary-tree-like relationships.

This has no effect on insert()s, a < test is good enough to keep consistency.

It has an effect on find() searches: \* Instead of using compare(child,node) for a < relationship and following 1 branch, we must use !compare(node,child) for a <= relationship, and test BOTH branches, as we could potentially go down both branches.

It has no real effect on bounds-based searches (like find\_nearest, find\_within\_range) as it compares vs a boundary and would follow both branches if required.

This has no real effect on erase()s, a < test is good enough to keep consistency.

### 7.34.2 Define Documentation

**7.34.2.1 #define KDTREE\_LIB\_VERSION "0\_7\_1"**

**7.34.2.2 #define KDTREE\_VERSION 701**

## 7.35 keyframedecoder.h File Reference

Utility methods for using strings, often for XML purposes. `#include <dtUtil/log.h>`  
`#include <dtUtil/xercesutils.h>`  
`#include <dtCore/refptr.h>`  
`#include <xercesc/dom/DOMTreeWalker.hpp>`

### Classes

- class **KeyFrameDecoder**< **RecordableT**, **FrameDataT** >  
*A class that fills key frame data with DOM parsing.*

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*

#### 7.35.1 Detailed Description

Utility methods for using strings, often for XML purposes. Author John K. Grant

## 7.36 librarysharingmanager.cpp File Reference

```
#include <prefix/dtutilprefix-src.h>
#include <dtUtil/macros.h>
#include <dlfcn.h>
#include <sstream>
#include <osgDB/FileUtils>
#include <osgDB/FileNameUtils>
#include <dtUtil/fileutils.h>
#include <dtUtil/librarysharingmanager.h>
```

### Classes

- class **InternalLibraryHandle**

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

## 7.37 librarysharingmanager.h File Reference

```
#include <set>
#include <map>
#include <string>
#include <vector>
#include <osg/Referenced>
#include <osgDB/DynamicLibrary>
#include <dtUtil/export.h>
#include <dtUtil/exception.h>
#include <dtUtil/enumeration.h>
#include <dtCore/refptr.h>
```

### Classes

- class **ExceptionEnum**
- class **LibraryHandle**  
*pure virtual class abstracting a checked-out handle to a library.*
- class **LibrarySharingManager**  
*Singleton for controlling loading and unloading libraries shared by multiple bodies of code.*

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*

## 7.38 log.cpp File Reference

```
#include <prefix/dtutilprefix-src.h>
#include <dtUtil/log.h>
#include <dtUtil/bits.h>
#include <dtCore/refptr.h>
#include <OpenThreads/Mutex>
#include <OpenThreads/ScopedLock>
#include <iomanip>
#include <iostream>
#include <fstream>
#include <cstdarg>
#include <cstdio>
#include <ctime>
#include <map>
```

### Classes

- struct **LogImpl**
- class **LogManager**

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

### Functions

- static dtCore::RefPtr< LogManager > **manager** (NULL)
- static std::string **sTitle** ("Delta 3D Engine Log File")

### Variables

- static const char \* **sLogFileName** = "delta3d\_log.html"

## 7.39 log.h File Reference

```
#include <string>
#include <osg/Referenced>
#include <dtUtil/export.h>
```

### Classes

- class **Log**  
*Log* (p. 205) class which the engine uses for all of its logging needs.
- class **LogFile**

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*

### Defines

- #define **LOG\_ALWAYS**(msg) dtUtil::Log::GetInstance().LogMessage(\_\_FUNCTION\_\_, \_\_LINE\_\_, msg, dtUtil::Log::LOG\_ALWAYS);
- #define **LOG\_DEBUG**(msg) dtUtil::Log::GetInstance().LogMessage(\_\_FUNCTION\_\_, \_\_LINE\_\_, msg, dtUtil::Log::LOG\_DEBUG);  
*Helps making logging a little easier.*
- #define **LOG\_ERROR**(msg) dtUtil::Log::GetInstance().LogMessage(\_\_FUNCTION\_\_, \_\_LINE\_\_, msg, dtUtil::Log::LOG\_ERROR);
- #define **LOG\_INFO**(msg) dtUtil::Log::GetInstance().LogMessage(\_\_FUNCTION\_\_, \_\_LINE\_\_, msg, dtUtil::Log::LOG\_INFO);
- #define **LOG\_WARNING**(msg) dtUtil::Log::GetInstance().LogMessage(\_\_FUNCTION\_\_, \_\_LINE\_\_, msg, dtUtil::Log::LOG\_WARNING);
- #define **LOGN\_ALWAYS**(name, msg) dtUtil::Log::GetInstance(name).LogMessage(\_\_FUNCTION\_\_, \_\_LINE\_\_, msg, dtUtil::Log::LOG\_ALWAYS);
- #define **LOGN\_DEBUG**(name, msg) dtUtil::Log::GetInstance(name).LogMessage(\_\_FUNCTION\_\_, \_\_LINE\_\_, msg, dtUtil::Log::LOG\_DEBUG);
- #define **LOGN\_ERROR**(name, msg) dtUtil::Log::GetInstance(name).LogMessage(\_\_FUNCTION\_\_, \_\_LINE\_\_, msg, dtUtil::Log::LOG\_ERROR);
- #define **LOGN\_INFO**(name, msg) dtUtil::Log::GetInstance(name).LogMessage(\_\_FUNCTION\_\_, \_\_LINE\_\_, msg, dtUtil::Log::LOG\_INFO);
- #define **LOGN\_WARNING**(name, msg) dtUtil::Log::GetInstance(name).LogMessage(\_\_FUNCTION\_\_, \_\_LINE\_\_, msg, dtUtil::Log::LOG\_WARNING);

#### 7.39.1 Define Documentation

**7.39.1.1 #define LOG\_ALWAYS(msg) dtUtil::Log::GetInstance().LogMessage(\_\_FUNCTION\_\_, \_\_LINE\_\_, msg, dtUtil::Log::LOG\_ALWAYS);**

**7.39.1.2 #define LOG\_DEBUG(msg) dtUtil::Log::GetInstance().LogMessage(\_\_FUNCTION\_\_, \_\_LINE\_\_, msg, dtUtil::Log::LOG\_DEBUG);**

Helps making logging a little easier. However, printf style logging is desired, you cannot use this macro.

- 7.39.1.3 `#define LOG_ERROR(msg) dtUtil::Log::GetInstance().LogMessage(__FUNCTION__, __LINE__, msg, dtUtil::Log::LOG_ERROR);`
- 7.39.1.4 `#define LOG_INFO(msg) dtUtil::Log::GetInstance().LogMessage(__FUNCTION__, __LINE__, msg, dtUtil::Log::LOG_INFO);`
- 7.39.1.5 `#define LOG_WARNING(msg) dtUtil::Log::GetInstance().LogMessage(__FUNCTION__, __LINE__, msg, dtUtil::Log::LOG_WARNING);`
- 7.39.1.6 `#define LOGN_ALWAYS(name, msg) dtUtil::Log::GetInstance(name).LogMessage(__FUNCTION__, __LINE__, msg, dtUtil::Log::LOG_ALWAYS);`
- 7.39.1.7 `#define LOGN_DEBUG(name, msg) dtUtil::Log::GetInstance(name).LogMessage(__FUNCTION__, __LINE__, msg, dtUtil::Log::LOG_DEBUG);`
- 7.39.1.8 `#define LOGN_ERROR(name, msg) dtUtil::Log::GetInstance(name).LogMessage(__FUNCTION__, __LINE__, msg, dtUtil::Log::LOG_ERROR);`
- 7.39.1.9 `#define LOGN_INFO(name, msg) dtUtil::Log::GetInstance(name).LogMessage(__FUNCTION__, __LINE__, msg, dtUtil::Log::LOG_INFO);`
- 7.39.1.10 `#define LOGN_WARNING(name, msg) dtUtil::Log::GetInstance(name).LogMessage(__FUNCTION__, __LINE__, msg, dtUtil::Log::LOG_WARNING);`

## 7.40 macros.h File Reference

```
#include <string>
```

```
#include <vector>
```

### Defines

- #define **BIT**(a) (long(1L<<long(a)))
- #define **DECLARE\_MANAGEMENT\_LAYER**(T)
- #define **DTUNREFERENCED\_PARAMETER**(parameter) ((void)parameter)
- #define **IMPLEMENT\_MANAGEMENT\_LAYER**(T)
- #define **IS\_A**(P, T) (dynamic\_cast<T>(P)!=NULL)
- #define **UNSIGNED\_BIT**(a) ((unsigned long)(1L<<(unsigned long)(a)))
- #define **UNSIGNED\_INT\_BIT**(a) ((unsigned int)(1L<<(unsigned int)(a)))

### 7.40.1 Define Documentation

#### 7.40.1.1 #define BIT(a) (long(1L<<long(a)))

#### 7.40.1.2 #define DECLARE\_MANAGEMENT\_LAYER(T)

**Value:**

```
private:
    static std::vector<T*> instances;           \
    static void RegisterInstance(T* instance); \
    static void DeregisterInstance(T* instance); \
public:
    static int GetInstanceCount();             \
    static T* GetInstance(int index);          \
    static T* GetInstance(std::string name);
```

#### 7.40.1.3 #define DTUNREFERENCED\_PARAMETER(parameter) ((void)parameter)

#### 7.40.1.4 #define IMPLEMENT\_MANAGEMENT\_LAYER(T)

#### 7.40.1.5 #define IS\_A(P, T) (dynamic\_cast<T>(P)!=NULL)

#### 7.40.1.6 #define UNSIGNED\_BIT(a) ((unsigned long)(1L<<(unsigned long)(a)))

#### 7.40.1.7 #define UNSIGNED\_INT\_BIT(a) ((unsigned int)(1L<<(unsigned int)(a)))

## 7.41 mainpage.h File Reference

### 7.41.1 Detailed Description

This file contains Doxygen special commands and text for the **Main Page** (p. ??) and some other minor aspects of this documentation. It is not part of Delta3D.

## 7.42 mathdefines.h File Reference

```
#include <cstdlib>
```

```
#include <cmath>
```

```
#include <cmath>
```

```
#include <osg/Math>
```

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

### Defines

- #define **RAND\_MAX** 0x7fff

### Functions

- template<typename Real >  
Real **Abs** (Real x)
- template<typename T >  
T **CalculateNormal** (T sX, T sMin, T sMax)  
*Normalizes a value within a specified space range.*
- template<typename Real >  
void **Clamp** (Real &x, const Real low, const Real high)
- template<typename Real >  
void **ClampMax** (Real &x, const Real high)
- template<typename Real >  
void **ClampMin** (Real &x, const Real low)
- template<typename TVec >  
bool **Equivalent** (const TVec &lhs, const TVec &rhs)  
*Does an epsilon equals on an any osg::Vec#, but auto calculates the epsilon per comparison.*
- template<typename TVec , typename Real >  
bool **Equivalent** (const TVec &lhs, const TVec &rhs, Real epsilon)  
*Does an epsilon equals on an any osg::Vec#.*
- template<typename TVec , typename Real >  
bool **Equivalent** (const TVec &lhs, const TVec &rhs, size\_t size, Real epsilon)  
*Does an epsilon equals on an any osg::Vec# or array.*
- bool **Equivalent** (double double1, double double2, double baseEpsilon=DBL\_EPSILON)  
*This does a relative comparison of doubles.*
- bool **Equivalent** (float float1, float float2, float baseEpsilon=FLT\_EPSILON)  
*This does a relative comparison of floats.*
- template<typename T >  
bool **IsFinite** (const T value)
- template<typename VecType >  
bool **IsFiniteVec** (const VecType value)
- template<typename T >  
bool **IsNaN** (const T value)
- template<typename Real >  
Real **Lerp** (Real x, Real y, Real t)

*Apply a linear interpolation between the two supplied numbers using a third percentage value.*

- `template<typename T >`  
**T MapRangeValue** (T sX, T xMin, T xMax, T yMin, T yMax)  
*Calculates the corresponding value for a mirrored space.*
- `template<typename Real >`  
**Real Max** (Real a, Real b)
- `template<typename Real >`  
**Real Min** (Real a, Real b)
- **float RandFloat** (float min, float max)
- **float RandPercent** ()
- **int RandRange** (int from, int to)
- `template<typename Real >`  
**bool WithinRange** (Real lhs, Real rhs, Real epsilon)

## 7.42.1 Define Documentation

### 7.42.1.1 #define RAND\_MAX 0x7fff

## 7.43 matrixutil.cpp File Reference

```
#include <prefix/dtutilprefix-src.h>  
#include <dtUtil/matrixutil.h>  
#include <iostream>
```

## 7.44 matrixutil.h File Reference

```
#include <osg/Matrix>
#include <osg/Vec3>
#include <osg/Vec4>
#include <dtUtil/export.h>
```

### Classes

- class **MatrixUtil**

*MatrixUtil* (p. 212) is a utility class for operating on an *osg::Matrix*.

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

## 7.45 mswin.h File Reference

## 7.46 nodecollector.cpp File Reference

```
#include <prefix/dtutilprefix-src.h>
#include <dtUtil/nodecollector.h>
#include <dtUtil/collectorutil.h>
#include <dtUtil/log.h>
#include <dtUtil/bits.h>
#include <osg/Node>
#include <osg/MatrixTransform>
#include <osgSim/DOFTransform>
#include <osg/Switch>
#include <osgSim/MultiSwitch>
#include <osg/Drawable>
#include <osg/Geode>
#include <osg/LOD>
```

### Classes

- class **GroupVisitor**

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

## 7.47 nodecollector.h File Reference

```
#include <dtCore/refptr.h>
#include <dtUtil/export.h>
#include <osg/Referenced>
#include <map>
#include <string>
```

### Classes

- class **NodeCollector**  
*NodeCollector* (p. 218) is used to gather osg Group nodes, DOFTransform nodes, MatrixTransform nodes, Switch Nodes and Geode nodes (which have Drawable objects and Material objects).

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*

## 7.48 nodeprintout.cpp File Reference

```
#include <prefix/dtutilprefix-src.h>
#include <dtUtil/nodeprintout.h>
#include <dtUtil/fileutils.h>
#include <dtUtil/log.h>
#include <dtUtil/bits.h>
#include <sstream>
#include <fstream>
#include <osg/Geode>
#include <osg/Geometry>
#include <osg/Group>
#include <osg/Node>
#include <osg/PrimitiveSet>
#include <osgDB/WriteFile>
```

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

## 7.49 nodeprintout.h File Reference

```
#include <dtUtil/export.h>
#include <osg/Referenced>
#include <osg/Node>
#include <sstream>
```

### Classes

- class **NodePrintOut**  
*Utility class used to traverse a node and generate a formatted text output.*

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*

## 7.50 noise1.h File Reference

```
#include "dtUtil/mathdefines.h"
```

### Classes

- class **Noise1**< **Real**, **Vector** >  
*An implementation of 1D Gradient Noise.*

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*

## 7.51 noise2.h File Reference

```
#include "dtUtil/mathdefines.h"
```

### Classes

- class **Noise2**< **Real**, **Vector** >  
*An implementation of 2D Gradient Noise.*

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*

## 7.52 noise3.h File Reference

```
#include "dtUtil/mathdefines.h"
```

### Classes

- class **Noise3**< **Real**, **Vector** >  
*An implementation of 3D Gradient Noise.*

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*

## 7.53 noisetexture.cpp File Reference

```
#include <prefix/dtutilprefix-src.h>
#include <dtUtil/noisetexture.h>
#include <cmath>
#include <cstdio>
#include <cstring>
```

## 7.54 noisetexture.h File Reference

```
#include <osg/Image>
#include <dtUtil/noiseutility.h>
#include <dtUtil/export.h>
```

### Classes

- class **NoiseTexture**

*Noise Texture is a class that uses **SeamlessNoise** (p. 251) to generate an osg::Image.*

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

## 7.55 noiseutility.h File Reference

```
#include <cstdlib>
#include <osg/Vec2f>
#include <osg/Vec3f>
#include <osg/Vec2d>
#include <osg/Vec3d>
#include "noise1.h"
#include "noise2.h"
#include "noise3.h"
#include "seamlessnoise.h"
#include "fractal.h"
```

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

### Typedefs

- typedef Fractal< double, osg::Vec2d, Noise2d > **Fractal2d**
- typedef Fractal< float, osg::Vec2f, Noise2f > **Fractal2f**
- typedef Fractal< double, osg::Vec3d, Noise3d > **Fractal3d**
- typedef Fractal< float, osg::Vec3f, Noise3f > **Fractal3f**
- typedef Noise1< double, double > **Noise1d**
- typedef Noise1< float, float > **Noise1f**

***NoiseUtility.h** (p. 372) contains all necessary defines to use the Noise Library in **dtUtil** (p. 13).*

- typedef Noise2< double, osg::Vec2d > **Noise2d**
- typedef Noise2< float, osg::Vec2f > **Noise2f**
- typedef Noise3< double, osg::Vec3d > **Noise3d**
- typedef Noise3< float, osg::Vec3f > **Noise3f**
- typedef Fractal< float, osg::Vec3f, SeamlessNoise > **SeamlessFractal**

## 7.56 objectfactory.h File Reference

```
#include <map>
#include <vector>
#include <osg/Referenced>
```

### Classes

- class **ObjectFactory**< **UniqueIdTypeClass**, **BaseTypeClass**, **ItCmpClass** >  
*This class is a template object factory.*

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*

### Functions

- template<typename BaseType , typename DerivedType >  
**BaseType \* construct** ()  
*Templated function to provide a generic object construction utility.*

## 7.57 packager.cpp File Reference

```
#include <dtUtil/packager.h>
#include <dtUtil/fileutils.h>
#include <osgDB/FileNameUtils>
#include <stdio.h>
#include <string.h>
```

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

## 7.58 packager.h File Reference

```
#include <string>
#include <vector>
#include <dtUtil/export.h>
```

### Classes

- class **Packager**  
*The **Packager** (p. 241) is used to package multiple files into a single .dtpkg file.*
- struct **PackTreeData**

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*

## 7.59 polardecomp.cpp File Reference

```
#include <prefix/dtutilprefix-src.h>  
#include <dtUtil/polardecomp.h>  
#include <dtUtil/matrixutil.h>  
#include <cmath>
```

## 7.60 polardecomp.h File Reference

```
#include <osg/Matrix>
#include <osg/Vec3>
#include <dtUtil/export.h>
```

### Classes

- class **PolarDecomp**

***PolarDecomp** (p. 244) is a class that will take a 4x4 Matrix and break it up into the following components Rotation, Scale, and Translation.*

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

## 7.61 precomp.cpp File Reference

```
#include <prefix/dtutilprefix-src.h>
```

### Defines

- #define **DELTA\_PCH**

#### 7.61.1 Define Documentation

##### 7.61.1.1 #define DELTA\_PCH

## 7.62 refstring.cpp File Reference

```
#include "prefix/dtutilprefix-src.h"  
#include <dtUtil/refstring.h>  
#include <ostream>  
#include <set>  
#include <OpenThreads/Mutex>
```

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

### Defines

- #define **THREAD\_SAFETY** 1
- #define **USE\_TABLE** 1

### Functions

- std::ostream & **operator**<< (std::ostream &stream, const RefString &rs)

### Variables

- static OpenThreads::Mutex **gStringSetMutex**
- static size\_t **StringCount** = 0
- static std::set< std::string > **StringSet**

#### 7.62.1 Define Documentation

7.62.1.1 #define **THREAD\_SAFETY** 1

7.62.1.2 #define **USE\_TABLE** 1

## 7.63 refstring.h File Reference

```
#include <dtUtil/export.h>
#include <string>
#include <iosfwd>
```

### Classes

- class **RefString**

*A string wrapper that will "intern" all of the strings so that strings with the same value will point to the same memory.*

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

### Functions

- bool **operator!=** (const std::string &s1, const RefString &s2)
- std::string **operator+** (const std::string &s1, const RefString &s2)
- std::ostream & **operator<<** (std::ostream &stream, const RefString &rs)
- bool **operator==** (const std::string &s1, const RefString &s2)

## 7.64 resourceloader.h File Reference

```
#include <osg/Referenced>
```

### Classes

- class **ResourceLoader**< **ResourceDescriptor**, **Resource** >

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*

## 7.65 resourcemanager.h File Reference

```
#include <map>
#include <string>
#include <dtUtil/resourceloader.h>
#include <dtUtil/log.h>
#include <dtCore/refptr.h>
#include <osg/Referenced>
```

### Classes

- class **ResourceManager**< **ResourceKey**, **Resource** >

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*

## 7.66 seamlessnoise.cpp File Reference

```
#include <prefix/dtutilprefix-src.h>
#include <dtUtil/seamlessnoise.h>
#include <cstdlib>
```

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

### Defines

- #define **LERP**(t, a, b) ((a) + (t) \* ((b) - (a)))

### Variables

- int **p** [512]
- int **permutation** [256]

#### 7.66.1 Define Documentation

##### 7.66.1.1 #define LERP(t, a, b) ((a) + (t) \* ((b) - (a)))

## 7.67 seamlessnoise.h File Reference

```
#include <osg/Vec3f>
#include "mathdefines.h"
```

### Classes

- class **SeamlessNoise**  
*SeamlessNoise* (p. 251) is a noise class that creates tileable noise.

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*

## 7.68 serializer.cpp File Reference

```
#include <prefix/dtutilprefix-src.h>  
#include <dtUtil/serializer.h>  
#include <dtUtil/stringutils.h>  
#include <xercesc/dom/DOMDocument.hpp>  
#include <xercesc/dom/DOMELEMENT.hpp>  
#include <xercesc/util/XMLString.hpp>
```

## 7.69 serializer.h File Reference

Utility functions for serialization. `#include <string>`  
`#include <dtUtil/export.h>`  
`#include <xercesc/util/XercesDefs.hpp>`

### Classes

- struct **Serializer**  
*A place to implement functions for serialization.*

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*

#### 7.69.1 Detailed Description

Utility functions for serialization. John K. Grant

## 7.70 stateattributecollector.cpp File Reference

```
#include <prefix/dtutilprefix-src.h>
#include <dtUtil/stateattributecollector.h>
#include <dtUtil/collectorutil.h>
#include <dtUtil/log.h>
#include <dtUtil/bits.h>
#include <osg/Node>
#include <osg/Drawable>
#include <osg/Geode>
#include <osg/Material>
#include <osg/Texture>
```

### Classes

- class **StateVisitor**

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

## 7.71 stateattributecollector.h File Reference

```
#include <osg/Referenced>
#include <dtCore/refptr.h>
#include <dtUtil/export.h>
#include <map>
#include <string>
```

### Classes

- class **StateAttributeCollector**

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

## 7.72 stringutils.cpp File Reference

```
#include <prefix/dtutilprefix-src.h>
#include <dtUtil/stringutils.h>
#include <dtUtil/macros.h>
#include <cstdio>
#include <sstream>
#include <iomanip>
#include <cmath>
```

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

### Functions

- void **MakeIndexString** (unsigned index, std::string &toFill, unsigned paddedLength=4)
- bool **Match** (const char \*Wildcards, const char \*str)
- static bool **Scan** (const char \*&wildCards, const char \*&str)
- double **ToDouble** (const std::string &str)  
*converts a std::string to a 'double'*
- float **ToFloat** (const std::string &d)  
*Converts a string to a 'float'.*
- template<>  
bool **ToType**< bool > (const std::string &u)  
*Special exception for bool where things like "True", "TRUE", and "true" should be accepted.*
- unsigned int **ToUnsignedInt** (const std::string &u)  
*converts a std::string to an 'unsigned int'*
- const std::string & **Trim** (std::string &toTrim)  
*Trims whitespace off the front and end of a string.*
- static bool **WildMatch** (const char \*wildCards, const char \*str)

## 7.73 stringutils.h File Reference

Utility methods for using strings. #include <dtUtil/export.h>

#include <sstream>

#include <algorithm>

#include <locale>

#include <string>

#include <vector>

#include <functional>

#include <osg/io\_utils>

#include <dtUtil/deprecationmgr.h>

### Classes

- class **IsDelimiter**  
*Generic string delimiter check function class.*
- class **IsSlash**  
*Determines if the current character is a forward slash.*
- class **IsSpace**  
*A functor which tests if a character is whitespace.*
- class **StringTokenizer**< **Pred** >

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*

### Functions

- void **MakeIndexString** (unsigned index, std::string &toFill, unsigned paddedLength=4)
- bool **Match** (const char \*Wildcards, const char \*str)
- template<class VecType >  
bool **ParseVec** (const std::string &value, VecType &vec, unsigned size, unsigned numberPrecision=16)  
*A templated function for taking any of the osg vector types and reading the data from a string.*
- double **ToDouble** (const std::string &str)  
*converts a std::string to a 'double'*
- float **ToFloat** (const std::string &d)  
*Converts a string to a 'float'.*
- template<typename T >  
std::string **ToString** (const T &t, int precision=-1)  
*A utility function to convert a basic type into a string.*
- template<typename T >  
T **ToType** (const std::string &u)  
*Converts a string to a specified type.*
- template<>  
bool **ToType**< **bool** > (const std::string &u)

*Special exception for bool where things like "True", "TRUE", and "true" should be accepted.*

- unsigned int **ToUnsignedInt** (const std::string &u)  
*converts a std::string to an 'unsigned int'*
- DEPRECATE\_FUNC const std::string & **trim** (std::string &toTrim)  
*Call Trim instead.*
- const std::string & **Trim** (std::string &toTrim)  
*Trims whitespace off the front and end of a string.*

### 7.73.1 Detailed Description

Utility methods for using strings. David Guthrie John K. Grant William E. Johnson II Chris Osborn

## 7.74 tangentspacevisitor.cpp File Reference

```
#include <prefix/dtutilprefix-src.h>
#include <dtUtil/tangentspacevisitor.h>
#include <osg/Program>
#include <osg/Geode>
#include <osgUtil/TangentSpaceGenerator>
#include <dtCore/refptr.h>
```

### Namespaces

- namespace **dtUtil**

*Contains generic, reusable features which are useful for most applications.*

## 7.75 tangentspacevisitor.h File Reference

```
#include <dtUtil/export.h>
```

```
#include <osg/NodeVisitor>
```

### Classes

- class **TangentSpaceVisitor**  
*This visitor is used to generate tangents for your node.*

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*

## 7.76 templateutility.h File Reference

```
#include <utility>
#include <algorithm>
#include <dtUtil/typetraits.h>
```

### Classes

- class **array\_remove**< **\_Container** >
- struct **DoNothing**< **Ret, T** >  
*This odd functor is actually useful for supplying a default template parameter when a functors is expected.*
- struct **DoNothing0**< **Ret** >  
*This odd functor is actually useful for supplying a default template parameter when a functors is expected.*
- struct **EvaluateFunctor**< **FunctorType, ArgType, RetType** >
- struct **EvaluateFunctor**< **FunctorType \***, **ArgType, RetType** >
- struct **EvaluateInvoke**< **CompareHandler, InvokeHandler, EvaluateResult** >
- class **insert\_back**< **\_Container, \_InputType** >
- class **insert\_back\_no\_duplicates**< **\_Container** >

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*

## 7.77 transformation.h File Reference

### Classes

- class **EdgeStepFilter**

The **EdgeStepFilter** (p. 108) will return back 1.0, 0.0, or -1.0 depending if the supplied input value is greater than the high, in between high and low, or less than the low.

- class **Transformation**< T >

An interface class to use for "things that transform" i.e., things that return a value given an input value.

### Namespaces

- namespace **dtUtil**

Contains generic, reusable features which are useful for most applications.

## 7.78 tree.h File Reference

### Classes

- class `tree< T >`
- class `tree_iterator< T >`

### Namespaces

- namespace `dtUtil`  
*Contains generic, reusable features which are useful for most applications.*

### Defines

- `#define NULL 0`

#### 7.78.1 Define Documentation

##### 7.78.1.1 `#define NULL 0`

## 7.79 typetraits.h File Reference

### Classes

- struct **\_IsConst**< **U** >
- struct **\_IsConst**< **const U** >
- struct **\_IsPointer**< **U** >
- struct **\_IsPointer**< **U \*** >
- struct **\_IsReference**< **U** >
- struct **\_IsReference**< **U &** >
- struct **\_TypeTraits**< **U** >
- struct **\_TypeTraits**< **const U &** >
- struct **\_TypeTraits**< **const U \*** >
- struct **\_TypeTraits**< **const U \*const** >
- struct **\_TypeTraits**< **U &** >
- struct **\_TypeTraits**< **U \*** >
- struct **\_TypeTraits**< **U \*const** >
- struct **IsConst**< **\_Type** >
- struct **IsPointer**< **\_Type** >
- struct **IsReference**< **\_Type** >
- struct **TypeTraits**< **\_Type** >

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*

## 7.80 version.cpp File Reference

```
#include <prefix/dtutilprefix-src.h>
#include <dtUtil/version.h>
#include <stdio.h>
```

### Functions

- const char \* **Delta3DGetLibraryName** ()  
*The osgGetLibraryName() method returns the library name in human-friendly form.*
- const char \* **Delta3DGetVersion** ()  
*delta3DGetVersion() returns the library version number.*

### 7.80.1 Function Documentation

#### 7.80.1.1 const char\* Delta3DGetLibraryName ()

The osgGetLibraryName() method returns the library name in human-friendly form.

#### 7.80.1.2 const char\* Delta3DGetVersion ()

delta3DGetVersion() returns the library version number. Numbering convention : Delta3D 1.4.0 will return "1.4.0" from delta3DGetVersion.

This C function can be also used to check for the existence of the Delta3D library using autoconf and its m4 macro AC\_CHECK\_LIB.

Here is the code to add to your configure.in:

```
#
# Check for the Delta3D library
#
AC_CHECK_LIB(osg, delta3DGetVersion, ,
[AC_MSG_ERROR(Delta3D library not found. See http://www.delta3d.org)]])
```

## 7.81 version.h File Reference

```
#include <dtUtil/export.h>
```

### Defines

- #define **DELTA3D\_VERSION\_MAJOR** 2
- #define **DELTA3D\_VERSION\_MINOR** 4
- #define **DELTA3D\_VERSION\_PATCH** 0

### Functions

- DT\_UTIL\_EXPORT const char \* **Delta3DGetLibraryName** ()  
*The osgGetLibraryName() method returns the library name in human-friendly form.*
- DT\_UTIL\_EXPORT const char \* **Delta3DGetVersion** ()  
*delta3DGetVersion() returns the library version number.*

#### 7.81.1 Define Documentation

**7.81.1.1 #define DELTA3D\_VERSION\_MAJOR** 2

**7.81.1.2 #define DELTA3D\_VERSION\_MINOR** 4

**7.81.1.3 #define DELTA3D\_VERSION\_PATCH** 0

#### 7.81.2 Function Documentation

**7.81.2.1 DT\_UTIL\_EXPORT const char\* Delta3DGetLibraryName** ()

The osgGetLibraryName() method returns the library name in human-friendly form.

**7.81.2.2 DT\_UTIL\_EXPORT const char\* Delta3DGetVersion** ()

delta3DGetVersion() returns the library version number. Numbering convention : Delta3D 1.4.0 will return "1.4.0" from delta3DGetVersion.

This C function can be also used to check for the existence of the Delta3D library using autoconf and its m4 macro AC\_CHECK\_LIB.

Here is the code to add to your configure.in:

```
#
# Check for the Delta3D library
#
AC_CHECK_LIB(osg, delta3DGetVersion, ,
[AC_MSG_ERROR(Delta3D library not found. See http://www.delta3d.org)],)
```

## 7.82 xerceserrorhandler.cpp File Reference

```
#include <prefix/dtutilprefix-src.h>
#include <dtUtil/xerceserrorhandler.h>
#include <dtUtil/stringutils.h>
#include <dtUtil/log.h>
#include <string>
#include <xercesc/util/XMLString.hpp>
```

## 7.83 xerceserrorhandler.h File Reference

```
#include <dtUtil/export.h>
#include <xercesc/sax/ErrorHandler.hpp>
#include <xercesc/sax/SAXParseException.hpp>
```

### Classes

- class **XercesErrorHandler**  
*Logs Xerces parsing errors.*

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*

### 7.83.1 Detailed Description

Author John K. Grant

## 7.84 xercesparser.cpp File Reference

```
#include <prefix/dtutilprefix-src.h>
#include <dtUtil/xercesparser.h>
#include <dtUtil/xerceserrorhandler.h>
#include <dtUtil/log.h>
#include <osgDB/FileUtils>
#include <osgDB/FileNameUtils>
#include <xercesc/util/PlatformUtils.hpp>
#include <xercesc/util/XMLUni.hpp>
#include <xercesc/util/XMLString.hpp>
#include <xercesc/sax2/XMLReaderFactory.hpp>
#include <xercesc/validators/common/Grammar.hpp>
```

## 7.85 xercesparser.h File Reference

```
#include <string>
#include <xercesc/sax2/ContentHandler.hpp>
#include <dtUtil/export.h>
```

### Classes

- class **XercesParser**  
*A class to unify the usage of Xerces parsing.*

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*

#### 7.85.1 Detailed Description

Author John K. Grant

## 7.86 xercesutils.cpp File Reference

```
#include <prefix/dtutilprefix-src.h>
#include <dtUtil/xercesutils.h>
#include <dtUtil/log.h>
#include <algorithm>
#include <xercesc/dom/DOMDocument.hpp>
#include <xercesc/dom/DOMNodeFilter.hpp>
#include <xercesc/dom/DOMTreeWalker.hpp>
#include <xercesc/dom/DOMAttr.hpp>
#include <xercesc/util/XMLString.hpp>
```

## 7.87 xercesutils.h File Reference

```
#include <dtUtil/export.h>
#include <string>
#include <vector>
#include <map>
#include <xercesc/util/XercesDefs.hpp>
#include <xercesc/sax2/Attributes.hpp>
#include <xercesc/dom/DOM.hpp>
#include <xercesc/util/XMLString.hpp>
```

### Classes

- class **AttributeSearch**  
*Searches a Xerces XML Attribute list for names of interest.*
- class **StringToXMLConverter**  
*Utility methods for using strings, often for XML purposes.*
- class **XMLStringConverter**  
*Utility methods for using strings, often for XML purposes.*

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*

### Functions

- `std::string DT_UTIL_EXPORT FindAttributeValueFor` (const char \*attributeName, XERCES\_CPP\_-  
NAMESPACE\_QUALIFIER DOMNamedNodeMap \*attrs)  
*A utility that finds the string value for a specifically named attribute when a DOM Node is available.*

## 7.88 xerceswriter.cpp File Reference

```
#include <prefix/dtutilprefix-src.h>
#include <dtUtil/xerceswriter.h>
#include <dtUtil/log.h>
#include <xercesc/framework/LocalFileFormatTarget.hpp>
#include <xercesc/dom/DOMImplementationRegistry.hpp>
#include <xercesc/dom/DOMImplementation.hpp>
#include <xercesc/dom/DOMDocument.hpp>
#include <xercesc/dom/DOMWriter.hpp>
#include <xercesc/util/XMLUni.hpp>
#include <xercesc/util/XMLString.hpp>
#include <xercesc/util/OutOfMemoryException.hpp>
```

## 7.89 xerceswriter.h File Reference

```
#include <osg/Referenced>
#include <string>
#include <vector>
#include <xercesc/util/XercesDefs.hpp>
#include <dtUtil/export.h>
```

### Classes

- class **XercesWriter**  
*A class that manages one XML DOM document.*

### Namespaces

- namespace **dtUtil**  
*Contains generic, reusable features which are useful for most applications.*

### 7.89.1 Detailed Description

Author John K. Grant

# Index

---

## - Symbols -

- ~AttributeSearch
  - dtUtil::AttributeSearch, 52
- ~Command
  - dtUtil::Command, 70
- ~Command0
  - dtUtil::Command0, 71
- ~Command1
  - dtUtil::Command1, 72
- ~Coordinates
  - dtUtil::Coordinates, 80
- ~DataStream
  - dtUtil::DataStream, 92
- ~DateTime
  - dtUtil::DateTime, 98
- ~DeprecationMgr
  - DeprecationMgr, 104
- ~Enumeration
  - dtUtil::Enumeration, 111
- ~Exception
  - dtUtil::Exception, 117
- ~Functor
  - dtUtil::Functor, 131
- ~HotSpotFileHandler
  - dtUtil::HotSpotFileHandler, 170
- ~InternalLibraryHandle
  - dtUtil::InternalLibraryHandle, 181
- ~KDTree
  - dtUtil::KDTree, 194
- ~KeyFrameDecoder
  - dtUtil::KeyFrameDecoder, 197
- ~LibraryHandle
  - dtUtil::LibrarySharingManager::LibraryHandle, 200
- ~Log
  - dtUtil::Log, 206
- ~LogManager
  - dtUtil::LogManager, 211
- ~NoLeakAlloc
  - dtUtil::\_Alloc\_base::NoLeakAlloc, 236
- ~NodeCollector
  - dtUtil::NodeCollector, 221
- ~NodePrintOut
  - dtUtil::NodePrintOut, 228
- ~Noise1
  - dtUtil::Noise1, 230
- ~Noise2
  - dtUtil::Noise2, 231
- ~Noise3
  - dtUtil::Noise3, 232
- ~NoiseTexture
  - dtUtil::NoiseTexture, 234
- ~ObjectFactory
  - dtUtil::ObjectFactory, 240
- ~Packager
  - dtUtil::Packager, 241
- ~RefString
  - dtUtil::RefString, 247
- ~ResourceLoader
  - dtUtil::ResourceLoader, 249
- ~ResourceManager
  - dtUtil::ResourceManager, 250
- ~SeamlessNoise
  - dtUtil::SeamlessNoise, 251
- ~StateAttributeCollector
  - dtUtil::StateAttributeCollector, 260
- ~StringToXMLConverter
  - dtUtil::StringToXMLConverter, 266
- ~TimeFormat
  - dtUtil::DateTime::TimeFormat, 278
- ~TimeOrigin
  - dtUtil::DateTime::TimeOrigin, 279
- ~TimeType
  - dtUtil::DateTime::TimeType, 280
- ~XMLStringConverter
  - dtUtil::XMLStringConverter, 309
- ~XercesErrorHandler
  - dtUtil::XercesErrorHandler, 306
- ~XercesParser
  - dtUtil::XercesParser, 307
- ~XercesWriter
  - dtUtil::XercesWriter, 308
- ~tree
  - dtUtil::tree, 284
- ~tree\_iterator
  - dtUtil::tree\_iterator, 288
- \_Alloc\_base
  - dtUtil::\_Alloc\_base, 38
- \_Base
  - dtUtil::KDTree, 193
- \_Base\_const\_ptr
  - dtUtil::\_Base\_iterator, 39
  - dtUtil::\_Node\_base, 44
  - dtUtil::KDTree, 193
- \_Base\_iterator
  - dtUtil::\_Base\_iterator, 39
- \_Base\_ptr
  - dtUtil::\_Alloc\_base, 37
  - dtUtil::\_Node, 43
  - dtUtil::\_Node\_base, 44
  - dtUtil::KDTree, 193
- \_CenterPt
  - dtUtil::\_Region, 47
- \_Iterator
  - dtUtil::\_Iterator, 42
- \_Link\_const\_type
  - dtUtil::\_Iterator, 42
  - dtUtil::KDTree, 193
- \_Link\_type
  - dtUtil::\_Node, 43
  - dtUtil::KDTree, 193
- \_M\_acc
  - dtUtil::\_Region, 47
  - dtUtil::KDTree, 195
- \_M\_allocate\_node
  - dtUtil::\_Alloc\_base, 38
- \_M\_check\_children
  - dtUtil::KDTree, 194
- \_M\_check\_node
  - dtUtil::KDTree, 194
- \_M\_cmp

- dtUtil:: Region, 47
- dtUtil::KDTree, 195
- \_M\_construct\_node
  - dtUtil::\_Alloc\_base, 38
- \_M\_count
  - dtUtil::KDTree, 195
- \_M\_count\_within\_range
  - dtUtil::KDTree, 194
- \_M\_deallocate\_node
  - dtUtil::\_Alloc\_base, 38
- \_M\_decrement
  - dtUtil::\_Base\_iterator, 39
- \_M\_delete\_node
  - dtUtil::KDTree, 194
- \_M\_destroy\_node
  - dtUtil::\_Alloc\_base, 38
- \_M\_dist
  - dtUtil::KDTree, 195
- \_M\_empty\_initialise
  - dtUtil::KDTree, 194
- \_M\_erase
  - dtUtil::KDTree, 194
- \_M\_erase\_subtree
  - dtUtil::KDTree, 194
- \_M\_find
  - dtUtil::KDTree, 194
- \_M\_find\_exact
  - dtUtil::KDTree, 194
- \_M\_find\_within\_range
  - dtUtil::KDTree, 194
- \_M\_get\_erase\_replacement
  - dtUtil::KDTree, 194
- \_M\_get\_j\_max
  - dtUtil::KDTree, 194
- \_M\_get\_j\_min
  - dtUtil::KDTree, 194
- \_M\_get\_leftmost
  - dtUtil::KDTree, 194
- \_M\_get\_rightmost
  - dtUtil::KDTree, 194
- \_M\_get\_root
  - dtUtil::KDTree, 194
- \_M\_header
  - dtUtil::KDTree, 195
- \_M\_high\_bounds
  - dtUtil::\_Region, 47
- \_M\_increment
  - dtUtil::\_Base\_iterator, 39
- \_M\_insert
  - dtUtil::KDTree, 194
- \_M\_insert\_left
  - dtUtil::KDTree, 194
- \_M\_insert\_right
  - dtUtil::KDTree, 194
- \_M\_left
  - dtUtil::\_Node\_base, 44
- \_M\_low\_bounds
  - dtUtil::\_Region, 47
- \_M\_matches\_node
  - dtUtil::KDTree, 194
- \_M\_matches\_node\_in\_d
  - dtUtil::KDTree, 194
- \_M\_matches\_node\_in\_other\_ds
  - dtUtil::KDTree, 194
- \_M\_new\_node
  - dtUtil::KDTree, 194
- \_M\_node
  - dtUtil::\_Base\_iterator, 39
- \_M\_node\_allocator
  - dtUtil::\_Alloc\_base, 38
- \_M\_optimise
  - dtUtil::KDTree, 194
- \_M\_parent
  - dtUtil::\_Node\_base, 44
- \_M\_right
  - dtUtil::\_Node\_base, 44
- \_M\_root
  - dtUtil::KDTree, 195
- \_M\_set\_leftmost
  - dtUtil::KDTree, 194
- \_M\_set\_rightmost
  - dtUtil::KDTree, 194
- \_M\_set\_root
  - dtUtil::KDTree, 194
- \_M\_value
  - dtUtil::\_Node, 43
- \_M\_visit\_within\_range
  - dtUtil::KDTree, 194
- \_Node
  - dtUtil::\_Node, 43
- \_Node\_
  - dtUtil::\_Alloc\_base, 37
- \_Node\_base
  - dtUtil::\_Node\_base, 44
- \_Node\_compare
  - dtUtil::\_Node\_compare, 45
- \_Node\_compare\_
  - dtUtil::KDTree, 193
- \_Region
  - dtUtil::\_Region, 47
- \_Region\_
  - dtUtil::KDTree, 193
- \_S\_accumulate\_node\_distance
  - dtUtil, 23
- \_S\_is\_leaf
  - dtUtil::KDTree, 194
- \_S\_left
  - dtUtil::KDTree, 194
- \_S\_maximum
  - dtUtil::\_Node\_base, 44
  - dtUtil::KDTree, 194
- \_S\_minimum
  - dtUtil::\_Node\_base, 44
  - dtUtil::KDTree, 194
- \_S\_node\_compare
  - dtUtil, 23
- \_S\_node\_descend
  - dtUtil, 23
- \_S\_node\_distance
  - dtUtil, 23
- \_S\_node\_nearest
  - dtUtil, 23
- \_S\_parent
  - dtUtil::KDTree, 194
- \_S\_right
  - dtUtil::KDTree, 194
- \_S\_set\_left
  - dtUtil::KDTree, 194
- \_S\_set\_parent
  - dtUtil::KDTree, 194
- \_S\_set\_right
  - dtUtil::KDTree, 194

- \_S\_value
  - dtUtil::KdTree, 194
- \_Self
  - dtUtil::\_Iterator, 42
- A -**
- Abs
  - dtUtil, 24
- AD\_C
  - dtUtil, 30
- Add
  - dtUtil::Bits, 32
- AddDeprecatedFunction
  - DeprecationMgr, 104
- AddDOFTransform
  - dtUtil::NodeCollector, 222
- AddFile
  - dtUtil::Packager, 242
- AddGeode
  - dtUtil::NodeCollector, 222
- AddGroup
  - dtUtil::NodeCollector, 222
- AddInstance
  - dtUtil::LogManager, 211
- AddLOD
  - dtUtil::NodeCollector, 222
- AddMaterial
  - dtUtil::StateAttributeCollector, 261
- AddMatrixTransform
  - dtUtil::NodeCollector, 222
- AddMultiSwitch
  - dtUtil::NodeCollector, 222
- AddNode
  - dtUtil::CollectorUtil, 33
- AddProgram
  - dtUtil::StateAttributeCollector, 261
- AddResource
  - dtUtil::ResourceManager, 250
- AddSwitch
  - dtUtil::NodeCollector, 222
- AddTexture
  - dtUtil::StateAttributeCollector, 261
- AddToSearchPath
  - dtUtil::LibrarySharingManager, 202
- AdjustTimeZone
  - dtUtil::DateTime, 98
- AllAttributes
  - dtUtil::StateAttributeCollector, 263
- AllNodeTypes
  - dtUtil::NodeCollector, 226
- allocator\_type
  - dtUtil::\_Alloc\_base, 37
  - dtUtil::KdTree, 193
- Analyze
  - dtUtil::NodePrintOut, 228
- AnalyzeGeode
  - dtUtil::NodePrintOut, 228
- AnalyzePrimSet
  - dtUtil::NodePrintOut, 228
- AppendDataStream
  - dtUtil::DataStream, 92
- apply
  - dtUtil::BoundingBoxVisitor, 59
  - dtUtil::GeometryCollector, 164
  - dtUtil::GroupVisitor, 166, 167
  - dtUtil::StateVisitor, 264
  - dtUtil::TangentSpaceVisitor, 269
- array\_remove
  - dtUtil::array\_remove, 51
- AttributeSearch
  - dtUtil::AttributeSearch, 52
- B -**
- barycentric.h, 311
- BarycentricSpace
  - dtUtil::BarycentricSpace, 53
- BaseExceptionType
  - dtUtil::BaseExceptionType, 54
- baseName
  - dtUtil::FileInfo, 120
- BaseType
  - dtUtil::ObjectFactory, 240
- begin
  - dtUtil::KdTree, 194
  - dtUtil::tree, 284
  - dtUtil::tree\_iterator, 288
- Bind
  - dtUtil, 25
- Binder
  - dtUtil::Binder, 56
- BIT
  - macros.h, 356
- bits.h, 312
- BoundingBoxVisitor
  - dtUtil::BoundingBoxVisitor, 59
- boundingshapeutils.h, 313
- BoundParamsTL
  - dtUtil::Binder, 56
- BoundPTL
  - dtUtil::Binder, 56
- BoundTL
  - dtUtil::BoundHelper, 58
- BREAK\_OVERRIDE
  - breakoverride.h, 314
- breakoverride.h, 314
  - BREAK\_OVERRIDE, 314
- BUFFER\_INVALID
  - dtUtil::DataStreamException, 95
- BUFFER\_INVALID\_POS
  - dtUtil::DataStreamException, 95
- BUFFER\_READ\_ERROR
  - dtUtil::DataStreamException, 95
- BUFFER\_WRITE\_ERROR
  - dtUtil::DataStreamException, 95
- C -**
- c\_str
  - dtUtil::RefString, 247
  - dtUtil::XMLStringConverter, 309
- CalcTransverseMercatorParameters
  - dtUtil::UTMParameters, 304
- CalculateBarycentricCenter
  - dtUtil, 25
- CalculateConvergencParamForFlatEarth
  - dtUtil::Coordinates, 80
- CalculateMagneticNorthOffset
  - dtUtil::Coordinates, 80
- CalculateNormal
  - dtUtil, 25
- CalculateUTMZone
  - dtUtil::Coordinates, 80

- CALENDAR\_DATE\_AND\_TIME\_FORMAT
  - dtUtil::DateTime::TimeFormat, 278
- CALENDAR\_DATE\_FORMAT
  - dtUtil::DateTime::TimeFormat, 278
- Call
  - dtUtil::FunctorCall< CallType, R, TYPELIST\_0()>, 132
  - dtUtil::FunctorCall< CallType, R, TYPELIST\_1(P1)>, 133
  - dtUtil::FunctorCall< CallType, R, TYPELIST\_2(P1, P2)>, 134
  - dtUtil::FunctorCall< CallType, R, TYPELIST\_3(P1, P2, P3)>, 135
  - dtUtil::FunctorCall< CallType, R, TYPELIST\_4(P1, P2, P3, P4)>, 136
  - dtUtil::FunctorCall< CallType, R, TYPELIST\_5(P1, P2, P3, P4, P5)>, 137
  - dtUtil::FunctorCall< CallType, R, TYPELIST\_6(P1, P2, P3, P4, P5, P6)>, 138
  - dtUtil::FunctorCall< CallType, R, TYPELIST\_7(P1, P2, P3, P4, P5, P6, P7)>, 139
- call\_
  - dtUtil::Functor::FunImplBase::VTable, 305
- CalledFrom
  - DeprecatedFunction, 103
- CARTESIAN
  - dtUtil::LocalCoordinateType, 204
- CARTESIAN\_FLAT\_EARTH
  - dtUtil::LocalCoordinateType, 204
- CARTESIAN\_UTM
  - dtUtil::LocalCoordinateType, 204
- CentralMeridianScale
  - dtUtil, 30
- ChangeDirectory
  - dtUtil::FileUtils, 122
- characters
  - dtUtil::HotSpotFileHandler, 170
- check\_tree
  - dtUtil::KDTree, 194
- Clamp
  - dtUtil, 26
- ClampMax
  - dtUtil, 26
- ClampMin
  - dtUtil, 26
- ClampUnity
  - dtUtil::MatrixUtil, 212
- clear
  - dtUtil::KDTree, 194
  - dtUtil::tree, 284
- clear\_children
  - dtUtil::tree\_iterator, 288
- clear\_tree
  - dtUtil::tree\_iterator, 288
- ClearAll
  - dtUtil::NodeCollector, 222
  - dtUtil::StateAttributeCollector, 261
- ClearBuffer
  - dtUtil::DataStream, 92
- ClearSearchPath
  - dtUtil::LibrarySharingManager, 202
- CLOCK\_TIME
  - dtUtil::DateTime::TimeType, 280
- CLOCK\_TIME\_12\_HOUR\_FORMAT
  - dtUtil::DateTime::TimeFormat, 278
- CLOCK\_TIME\_24\_HOUR\_FORMAT
  - dtUtil::DateTime::TimeFormat, 278
- clone\_
  - dtUtil::Functor::FunImplBase::VTable, 305
- ClosePackage
  - dtUtil::Packager, 242
- CollectNodeData
  - dtUtil::NodePrintOut, 229
- CollectNodes
  - dtUtil::NodeCollector, 222
- collectorutil.h, 315
- CollectStateAttributes
  - dtUtil::StateAttributeCollector, 261
- Command
  - dtUtil::Command, 70
- command.h, 316
- Command0
  - dtUtil::Command0, 71
- Command1
  - dtUtil::Command1, 72
- Command2
  - dtUtil::Command2, 75
- configproperties.h, 317
- const\_iterator
  - dtUtil::\_Iterator, 42
  - dtUtil::KDTree, 193
  - dtUtil::tree, 284
  - dtUtil::tree\_iterator, 288
- const\_param\_type
  - dtUtil::TypeTraits, 299
- const\_pointer
  - dtUtil::KDTree, 194
- const\_reference
  - dtUtil::KDTree, 194
  - dtUtil::TypeTraits, 299
- const\_return\_type
  - dtUtil::TypeTraits, 299
- const\_reverse\_iterator
  - dtUtil::KDTree, 194
- ConstRef
  - dtUtil::details::TypeTraits, 298
- construct
  - dtUtil, 26
- container
  - dtUtil::array\_remove, 51
  - dtUtil::insert\_back, 174
  - dtUtil::insert\_back\_no\_duplicates, 175
- container\_type
  - dtUtil::array\_remove, 51
  - dtUtil::insert\_back, 174
  - dtUtil::insert\_back\_no\_duplicates, 175
- ConvertFlatEarthToLatLon
  - dtUtil::Coordinates, 80
- ConvertGeocentricToGeodetic
  - dtUtil::Coordinates, 81
- ConvertGeodeticToTransverseMercator
  - dtUtil::Coordinates, 81
- ConvertGeodeticToUTM
  - dtUtil::Coordinates, 81
- ConvertLatLonToFlatEarth
  - dtUtil::Coordinates, 81
- ConvertMGRSToUTM
  - dtUtil::Coordinates, 82
- ConvertMGRSToXYZ
  - dtUtil::Coordinates, 82
- ConvertToLocalRotation
  - dtUtil::Coordinates, 82
- ConvertToLocalTranslation

- dtUtil::Coordinates, 82
- ConvertToRemoteRotation
  - dtUtil::Coordinates, 82
- ConvertToRemoteTranslation
  - dtUtil::Coordinates, 82
- ConvertTransverseMercatorToGeodetic
  - dtUtil::Coordinates, 82
- ConvertUTMToGeodetic
  - dtUtil::Coordinates, 82
- ConvertUTMToMGRS
  - dtUtil::Coordinates, 82
- Coordinates
  - dtUtil::Coordinates, 80
- coordinates.cpp, 318
- coordinates.h, 319
- copy\_tree
  - dtUtil::tree, 284
- COS\_67P5
  - dtUtil, 30
- count
  - dtUtil::squared\_difference\_counted, 258
- count\_within\_range
  - dtUtil::KDTree, 194
- CreateDocument
  - dtUtil::XercesWriter, 308
- CreateObject
  - dtUtil::ObjectFactory, 240
- createObjectFunc
  - dtUtil::ObjectFactory, 240
- CURRENT
  - dtUtil::DataStream::SeekTypeEnum, 252
- CurrentDirectory
  - dtUtil::FileUtils, 122
- D -**
- data
  - dtUtil::tree, 284
  - dtUtil::tree\_iterator, 288
- DataStream
  - dtUtil::DataStream, 92
- datastream.cpp, 321
- datastream.h, 322
- DateTime
  - dtUtil::DateTime, 98
- datetime.cpp, 323
- datetime.h, 324
- DECLARE\_ENUM
  - enumeration.h, 329
- DECLARE\_MANAGEMENT\_LAYER
  - macros.h, 356
- DecodeFrameStamp
  - dtUtil::KeyFrameDecoder, 197
- DecodeSourceData
  - dtUtil::KeyFrameDecoder, 197
- Decompose
  - dtUtil::PolarDecomp, 244
- DEFAULT\_LOCALE\_NAME
  - dtUtil::IsSpace, 189
- DEFAULT\_VALUE
  - dtUtil::HotSpotFileHandler, 170
- DegreesToMils
  - dtUtil::Coordinates, 83
- DELTA3D\_VERSION\_MAJOR
  - version.h, 399
- DELTA3D\_VERSION\_MINOR
  - version.h, 399
- DELTA3D\_VERSION\_PATCH
  - version.h, 399
- Delta3DGetLibraryName
  - version.cpp, 398
  - version.h, 399
- Delta3DGetVersion
  - version.cpp, 398
  - version.h, 399
- DELTA\_PCH
  - precomp.cpp, 378
- DENOM
  - dtUtil::UTMParameters, 304
- DEPRECATE
  - deprecationmgr.h, 326
- DeprecatedFunction, 103
  - CalledFrom, 103
  - NewFunctionName, 103
  - OldFunctionName, 103
- DeprecationMgr, 104
  - ~DeprecationMgr, 104
  - AddDeprecatedFunction, 104
  - GetInstance, 104
- deprecationmgr.cpp, 325
- deprecationmgr.h, 326
  - DEPRECATE, 326
- DeprecationMgrImpl, 105
  - mFunctions, 105
- destroy\_
  - dtUtil::Functor::FunImplBase::VTable, 305
- difference\_type
  - dtUtil::\_Iterator, 42
  - dtUtil::KDTree, 194
- DirCopy
  - dtUtil::FileUtils, 122
- DirDelete
  - dtUtil::FileUtils, 122
- DIRECTORY
  - dtUtil, 23
- DirectoryContents
  - dtUtil, 23
- DirExists
  - dtUtil::FileUtils, 123
- DirGetFiles
  - dtUtil::FileUtils, 123
- DirGetSubs
  - dtUtil::FileUtils, 123
- disconnect
  - dtUtil::\_Alloc\_base::NoLeakAlloc, 236
- distance\_type
  - dtUtil::KDTree, 194
  - dtUtil::squared\_difference, 257
  - dtUtil::squared\_difference\_counted, 258
- DoCall
  - functor.h, 340
- DOFTransformFlag
  - dtUtil::NodeCollector, 226
- DT\_UTIL\_EXPORT
  - export.h, 333
- DTUNREFERENCED\_PARAMETER
  - macros.h, 356
- dtUtil, 13
  - \_S\_accumulate\_node\_distance, 23
  - \_S\_node\_compare, 23
  - \_S\_node\_descend, 23
  - \_S\_node\_distance, 23
  - \_S\_node\_nearest, 23

Abs, 24  
 AD\_C, 30  
 Bind, 25  
 CalculateBarycentricCenter, 25  
 CalculateNormal, 25  
 CentralMeridianScale, 30  
 Clamp, 26  
 ClampMax, 26  
 ClampMin, 26  
 construct, 26  
 COS\_67P5, 30  
 DIRECTORY, 23  
 DirectoryContents, 23  
 Equivalent, 26  
 FILE\_NOT\_FOUND, 23  
 FileExtensionList, 23  
 FileType, 23  
 FindAttributeValueFor, 27  
 flatteningReciprocal, 30  
 Fractal2d, 23  
 Fractal2f, 23  
 Fractal3d, 23  
 Fractal3f, 23  
 Geocent\_a, 30  
 Geocent\_e2, 31  
 Geocent\_e2\_2, 31  
 Geocent\_e2\_3, 31  
 Geocent\_ef, 31  
 Geocent\_ef\_3, 31  
 Geocent\_ef\_4, 31  
 Geocent\_ep2, 31  
 Geocent\_f, 31  
 getFileExtensionIncludingDot, 27  
 GetH, 27  
 gStringSetMutex, 31  
 iMakeDirectory, 27  
 IMPLEMENT\_ENUM, 27  
 IsFinite, 27  
 IsFiniteVec, 27  
 IsNAN, 27  
 Lerp, 27  
 LOGNAME, 31  
 MagneticNorthLatitude, 31  
 MagneticNorthLongitude, 31  
 MakeFunctor, 27, 28  
 MakeIndexString, 28  
 manager, 28  
 MapRangeValue, 28  
 Match, 28  
 Max, 28  
 MAX\_DELTA\_LONG, 31  
 MAX\_EASTING, 31  
 MAX\_LAT, 31  
 MAX\_NORTHING, 31  
 MAX\_SCALE\_FACTOR, 31  
 MergeParms, 28  
 METERS\_PER\_DEGREE, 31  
 Min, 28  
 MIN\_EASTING, 31  
 MIN\_LAT, 31  
 MIN\_NORTHING, 31  
 MIN\_SCALE\_FACTOR, 31  
 Noise1d, 23  
 Noise1f, 23  
 Noise2d, 23  
 Noise2f, 23  
 Noise3d, 23  
 Noise3f, 23  
 operator<<, 28  
 operator+, 28  
 operator==, 28, 29  
 p, 31  
 ParseVec, 29  
 permutation, 31  
 RandFloat, 29  
 RandPercent, 29  
 RandRange, 29  
 REGULAR\_FILE, 23  
 safeASIN, 29  
 Scan, 29  
 SeamlessFractal, 23  
 semiMajorAxis, 31  
 sLogFileName, 31  
 sTitle, 29  
 StringCount, 31  
 StringSet, 31  
 ToDouble, 29  
 ToFloat, 29  
 ToString, 29  
 ToType, 29  
 ToType< bool >, 30  
 ToUnsignedInt, 30  
 Trim, 30  
 trim, 30  
 WildMatch, 30  
 WithinRange, 30  
 dtutil.h, 327  
 dtUtil:: Alloc\_base, 37  
   \_Alloc\_base, 38  
   \_Base\_ptr, 37  
   \_M\_allocate\_node, 38  
   \_M\_construct\_node, 38  
   \_M\_deallocate\_node, 38  
   \_M\_destroy\_node, 38  
   \_M\_node\_allocator, 38  
   \_Node\_, 37  
   allocator\_type, 37  
   get\_allocator, 38  
 dtUtil:: Alloc\_base::NoLeakAlloc, 236  
   ~NoLeakAlloc, 236  
   disconnect, 236  
   get, 236  
   NoLeakAlloc, 236  
 dtUtil:: Base\_iterator, 39  
   \_Base\_const\_ptr, 39  
   \_Base\_iterator, 39  
   \_M\_decrement, 39  
   \_M\_increment, 39  
   \_M\_node, 39  
   KDTree, 39  
 dtUtil:: Bracket\_accessor, 40  
   operator(), 40  
   result\_type, 40  
 dtUtil:: Iterator, 41  
   \_iterator, 42  
   \_Link\_const\_type, 42  
   \_Self, 42  
   const\_iterator, 42  
   difference\_type, 42  
   get\_raw\_node, 42  
   iterator, 42  
   iterator\_category, 42

- operator\*, 42
- operator++, 42
- operator->, 42
- operator--, 42
- operator==, 42
- pointer, 42
- reference, 42
- value\_type, 42
- dtUtil::\_Node, 43
  - \_Base\_ptr, 43
  - \_Link\_type, 43
  - \_M\_value, 43
  - \_Node, 43
- dtUtil::\_Node\_base, 44
  - \_Base\_const\_ptr, 44
  - \_Base\_ptr, 44
  - \_M\_left, 44
  - \_M\_parent, 44
  - \_M\_right, 44
  - \_Node\_base, 44
  - \_S\_maximum, 44
  - \_S\_minimum, 44
- dtUtil::\_Node\_compare, 45
  - \_Node\_compare, 45
  - operator(), 45
- dtUtil::\_Region, 46
  - \_CenterPt, 47
  - \_M\_acc, 47
  - \_M\_cmp, 47
  - \_M\_high\_bounds, 47
  - \_M\_low\_bounds, 47
  - \_Region, 47
  - encloses, 47
  - intersects\_with, 47
  - set\_high\_bound, 47
  - set\_low\_bound, 47
  - subvalue\_type, 47
  - value\_type, 47
- dtUtil::always\_true, 48
  - operator(), 48
- dtUtil::AppendTL, 49
  - Type, 49
- dtUtil::AppendTL< NullType, T >, 50
  - Type, 50
- dtUtil::array\_remove, 51
  - array\_remove, 51
  - container, 51
  - container\_type, 51
  - operator(), 51
  - reference, 51
- dtUtil::AttributeSearch, 52
  - ~AttributeSearch, 52
  - AttributeSearch, 52
  - operator(), 52
  - ResultMap, 52
- dtUtil::BarycentricSpace, 53
  - BarycentricSpace, 53
  - Transform, 53
- dtUtil::BaseExceptionType, 54
  - BaseExceptionType, 54
  - GENERAL\_EXCEPTION, 54
- dtUtil::Binder, 55
  - Binder, 56
  - BoundParamsTL, 56
  - BoundPTL, 56
  - operator(), 56
  - Outgoing, 56
  - Parm1, 56
  - Parm2, 56
  - Parm3, 56
  - Parm4, 56
  - Parm5, 56
  - ResultType, 56
  - TypeListType, 56
  - UnboundParamsTL, 56
  - UnboundPTL, 56
- dtUtil::Bits, 32
  - Add, 32
  - Has, 32
  - Remove, 32
  - Toggle, 32
- dtUtil::BoundHelper, 58
  - BoundTL, 58
  - IncomingTL, 58
  - Parm1, 58
  - Parm2, 58
  - Parm3, 58
  - Parm4, 58
  - Parm5, 58
- dtUtil::BoundingBoxVisitor, 59
  - apply, 59
  - BoundingBoxVisitor, 59
  - mBoundingBox, 59
- dtUtil::BoundTL2, 60
  - Result, 60
- dtUtil::BreakOverride, 61
- dtUtil::CallParms< TYPELIST\_0()>, 62
  - Make, 62
  - ParmsListType, 62
- dtUtil::CallParms< TYPELIST\_1(P1)>, 63
  - Make, 63
  - ParmsListType, 63
- dtUtil::CallParms< TYPELIST\_2(P1, P2)>, 64
  - Make, 64
  - ParmsListType, 64
- dtUtil::CallParms< TYPELIST\_3(P1, P2, P3)>, 65
  - Make, 65
  - ParmsListType, 65
- dtUtil::CallParms< TYPELIST\_4(P1, P2, P3, P4)>, 66
  - Make, 66
  - ParmsListType, 66
- dtUtil::CallParms< TYPELIST\_5(P1, P2, P3, P4, P5)>, 67
  - Make, 67
  - ParmsListType, 67
- dtUtil::CallParms< TYPELIST\_6(P1, P2, P3, P4, P5, P6)>, 68
  - Make, 68
  - ParmsListType, 68
- dtUtil::CallParms< TYPELIST\_7(P1, P2, P3, P4, P5, P6, P7)>, 69
  - Make, 69
  - ParmsListType, 69
- dtUtil::CollectorUtil, 33
  - AddNode, 33
  - FindNodePointer, 33
  - RemoveNode, 34
- dtUtil::CollectorUtil::GetElementType, 165
  - value\_type, 165
- dtUtil::Command, 70
  - ~Command, 70
  - Command, 70
  - operator(), 70

- Return Type, 70
- dtUtil::Command0, 71
  - ~Command0, 71
  - Command0, 71
  - FunctorType, 71
  - operator(), 71
- dtUtil::Command1, 72
  - ~Command1, 72
  - Command1, 72
  - FunctorType, 72
  - MemberType, 72
  - operator(), 72
  - Param0, 72
  - Params, 72
  - TYPELIST\_1, 72
- dtUtil::Command2, 74
  - Command2, 75
  - FunctorType, 74
  - MemberType1, 74
  - MemberType2, 74
  - operator(), 75
  - Param0, 74
  - Param1, 74
  - Params, 75
  - TYPELIST\_2, 75
- dtUtil::ConfigProperties, 76
  - GetConfigPropertyValue, 76
  - RemoveConfigPropertyValue, 76
  - SetConfigPropertyValue, 76
- dtUtil::CoordinateConversionExceptionEnum, 77
  - INVALID\_INPUT, 77
- dtUtil::Coordinates, 78
  - ~Coordinates, 80
  - CalculateConvergenceParamForFlatEarth, 80
  - CalculateMagneticNorthOffset, 80
  - CalculateUTMZone, 80
  - ConvertFlatEarthToLatLon, 80
  - ConvertGeocentricToGeodetic, 81
  - ConvertGeodeticToTransverseMercator, 81
  - ConvertGeodeticToUTM, 81
  - ConvertLatLonToFlatEarth, 81
  - ConvertMGRSToUTM, 82
  - ConvertMGRSToXYZ, 82
  - ConvertToLocalRotation, 82
  - ConvertToLocalTranslation, 82
  - ConvertToRemoteRotation, 82
  - ConvertToRemoteTranslation, 82
  - ConvertTransverseMercatorToGeodetic, 82
  - ConvertUTMToGeodetic, 82
  - ConvertUTMToMGRS, 82
  - Coordinates, 80
  - DegreesToMils, 83
  - EulersToMatrix, 83
  - GeocentricToGeodetic, 83
  - GeodeticToGeocentric, 83
  - GetApplyRotationConversionMatrix, 83
  - GetFlatEarthOrigin, 84
  - GetGlobeRadius, 84
  - GetIncomingCoordinateType, 84
  - GetLocalCoordinateType, 84
  - GetLocalOffset, 84
  - GetMagneticNorthOffset, 84
  - GetOriginRotationMatrix, 84
  - GetOriginRotationMatrixInverse, 84
  - GetUTMHemisphere, 84
  - GetUTMZone, 84
  - MatrixToEulers, 84
  - MilsToDegrees, 84
  - operator=, 84
  - operator==, 84
  - ReconfigureRotationMatrix, 84
  - SetApplyRotationConversionMatrix, 84
  - SetFlatEarthOrigin, 85
  - SetGlobeRadius, 85
  - SetIncomingCoordinateType, 85
  - SetLocalCoordinateType, 85
  - SetLocalOffset, 85
  - SetMagneticNorthOffset, 85
  - SetUTMHemisphere, 85
  - SetUTMLocalOffsetAsLatLon, 85
  - SetUTMZone, 85
  - XYZToMGRS, 85
  - ZFlop, 85
- dtUtil::CreateIdsTL, 86
  - Type, 86
- dtUtil::CreateIdsTL<-1,-1,-1,-1,-1,-1,-1,-1 >, 87
  - Type, 87
- dtUtil::CreateTL, 88
  - Type, 88
- dtUtil::CreateTL< NullType, NullType, NullType, NullType, NullType, NullType, NullType, NullType >, 89
  - Type, 89
- dtUtil::DataStream, 90
  - ~DataStream, 92
  - AppendDataStream, 92
  - ClearBuffer, 92
  - DataStream, 92
  - GetBuffer, 92
  - GetBufferCapacity, 92
  - GetBufferSize, 92
  - GetRemainingReadSize, 92
  - IncreaseBufferSize, 92
  - IsLittleEndian, 92
  - operator<<, 92, 93
  - operator>>, 93
  - operator=, 93
  - Read, 93
  - ReadBinary, 93
  - Rewind, 93
  - Seek, 93
  - Seekp, 93
  - SetBufferSize, 93
  - SetForceLittleEndian, 93
  - Write, 94
  - WriteBinary, 94
  - WriteBytes, 94
- dtUtil::DataStream::SeekTypeEnum, 252
  - CURRENT, 252
  - END, 252
  - SET, 252
- dtUtil::DataStreamException, 95
  - BUFFER\_INVALID, 95
  - BUFFER\_INVALID\_POS, 95
  - BUFFER\_READ\_ERROR, 95
  - BUFFER\_WRITE\_ERROR, 95
- dtUtil::DateTime, 96
  - ~DateTime, 98
  - AdjustTimeZone, 98
  - DateTime, 98
  - GetDay, 98
  - GetGMTOffset, 99
  - GetGMTTime, 99

- GetHour, 99
- GetLocalGMTOffset, 99
- GetMinute, 99
- GetMonth, 99
- GetSecond, 99
- GetTime, 99, 100
- GetTimeFormat, 100
- GetTimeInSeconds, 100
- GetTimeOrigin, 100
- GetTimeScale, 100
- GetTimeType, 100
- GetYear, 100
- IncrementClock, 100
- operator std::string, 100
- operator time\_t, 101
- operator tm, 101
- operator=, 101
- SetDay, 101
- SetGMTOffset, 101
- SetHour, 101
- SetMinute, 101
- SetMonth, 101
- SetSecond, 101
- SetTime, 101
- SetTimeFormat, 102
- SetTimeOrigin, 102
- SetTimeScale, 102
- SetTimeType, 102
- SetToGMTTime, 102
- SetToLocalTime, 102
- SetYear, 102
- ToString, 102
- dtUtil::DateTime::TimeFormat, 278
  - ~TimeFormat, 278
  - CALENDAR\_DATE\_AND\_TIME\_FORMAT, 278
  - CALENDAR\_DATE\_FORMAT, 278
  - CLOCK\_TIME\_12\_HOUR\_FORMAT, 278
  - CLOCK\_TIME\_24\_HOUR\_FORMAT, 278
  - LEXICAL\_DATE\_FORMAT, 278
  - LOCAL\_DATE\_AND\_TIME\_FORMAT, 278
  - LOCAL\_DATE\_FORMAT, 278
  - ORDINAL\_DATE\_FORMAT, 278
  - WEEK\_DATE\_FORMAT, 278
- dtUtil::DateTime::TimeOrigin, 279
  - ~TimeOrigin, 279
  - GMT\_TIME, 279
  - LOCAL\_TIME, 279
- dtUtil::DateTime::TimeType, 280
  - ~TimeType, 280
  - CLOCK\_TIME, 280
  - SCENARIO\_TIME, 280
  - SIMULATION\_TIME, 280
  - TIME\_STAMP, 280
  - TIME\_TYPE\_OTHER, 280
  - TRIP\_TIME, 280
- dtUtil::details, 35
- dtUtil::details::TypeTraits, 298
  - ConstRef, 298
  - NonConstNoRef, 298
  - NonConstRef, 298
- dtUtil::DoNothing, 106
  - operator(), 106
- dtUtil::DoNothing0, 107
  - operator(), 107
- dtUtil::EdgeStepFilter, 108
  - EdgeStepFilter, 108
- operator(), 108
- dtUtil::EmptyType, 109
- dtUtil::Enumeration, 110
  - ~Enumeration, 111
  - Enumeration, 111
  - GetName, 111
  - operator<, 111
  - operator>, 112
  - operator==, 111
- dtUtil::EvaluateFunctor, 113
  - operator(), 113
- dtUtil::EvaluateFunctor< FunctorType \*, ArgType, RetType >, 114
  - operator(), 114
- dtUtil::EvaluateInvoke, 115
  - operator(), 115
- dtUtil::Exception, 116
  - ~Exception, 117
  - Exception, 116
  - File, 117
  - Line, 117
  - LogException, 117
  - mFileName, 117
  - mLineNum, 117
  - mMessage, 117
  - mType, 117
  - Print, 117
  - ToString, 117
  - TypeEnum, 117
  - What, 117
- dtUtil::FileExceptionEnum, 119
  - FileExceptionEnum, 119
  - FileNotFound, 119
  - IOException, 119
- dtUtil::FileInfo, 120
  - baseName, 120
  - FileInfo, 120
  - fileName, 120
  - fileType, 120
  - lastModified, 120
  - path, 120
  - size, 120
- dtUtil::FileUtils, 121
  - ChangeDirectory, 122
  - CurrentDirectory, 122
  - DirCopy, 122
  - DirDelete, 122
  - DirExists, 123
  - DirGetFiles, 123
  - DirGetSubs, 123
  - FileCopy, 123
  - FileDelete, 123
  - FileExists, 124
  - FileMove, 124
  - GetAbsolutePath, 124
  - GetFileInfo, 124
  - GetInstance, 124
  - IsSameFile, 124
  - MakeDirectory, 125
  - PATH\_SEPARATOR, 126
  - PopDirectory, 125
  - PushDirectory, 125
  - RelativePath, 125
- dtUtil::Filter2, 127
  - Result, 127
- dtUtil::Filter2::Temp, 270

- Result, 270
- dtUtil::Filter2::Temp< dtUtil::NullType, i, dtUtil::NullType, Predicate >, 271
  - Result, 271
- dtUtil::Filter2::Temp< dtUtil::NullType, i, IdsTL2, Predicate >, 272
  - Result, 272
- dtUtil::Filter2::Temp< TL2, 0, dtUtil::NullType, dtUtil::IsIntType >, 273
  - Result, 273
- dtUtil::Filter2::Temp< TL2, 0, dtUtil::NullType, dtUtil::NotIntType >, 274
  - Result, 274
- dtUtil::Filter2::Temp< TL2, 0, typename dtUtil::IdsFromTL< TL2 >::Type, dtUtil::IsIntType >, 275
  - Result, 275
- dtUtil::Filter2::Temp< TL2, 0, typename dtUtil::IdsFromTL< TL2 >::Type, dtUtil::NotIntType >, 276
  - Result, 276
- dtUtil::Filter2::Temp< TL2, i, dtUtil::NullType, Predicate >, 277
  - Result, 277
- dtUtil::Fractal, 128
  - FBM, 128
  - HeteroFractal, 128
  - IslandFractal, 128
  - Marble, 128
  - RigidMultiFractal, 129
  - Turbulence, 129
- dtUtil::Functor, 130
  - ~Functor, 131
  - Functor, 131
  - operator(), 131
  - operator=, 131
  - Parm1, 131
  - Parm2, 131
  - Parm3, 131
  - Parm4, 131
  - Parm5, 131
  - Parm6, 131
  - Parm7, 131
  - ParmsListType, 131
  - ResultType, 131
  - TypeListType, 131
  - valid, 131
- dtUtil::Functor::FunImplBase::VTable, 305
  - call\_, 305
  - clone\_, 305
  - destroy\_, 305
- dtUtil::FunctorCall< CallType, R, TYPELIST\_0()>, 132
  - Call, 132
  - ParmsListType, 132
- dtUtil::FunctorCall< CallType, R, TYPELIST\_1(P1)>, 133
  - Call, 133
  - ParmsListType, 133
- dtUtil::FunctorCall< CallType, R, TYPELIST\_2(P1, P2)>, 134
  - Call, 134
  - ParmsListType, 134
- dtUtil::FunctorCall< CallType, R, TYPELIST\_3(P1, P2, P3)>, 135
  - Call, 135
  - ParmsListType, 135
- dtUtil::FunctorCall< CallType, R, TYPELIST\_4(P1, P2, P3, P4)>, 136
  - Call, 136
  - ParmsListType, 136
- dtUtil::FunctorCall< CallType, R, TYPELIST\_5(P1, P2, P3, P4, P5)>, 137
  - Call, 137
  - ParmsListType, 137
- dtUtil::FunctorCall< CallType, R, TYPELIST\_6(P1, P2, P3, P4, P5, P6)>, 138
  - Call, 138
  - ParmsListType, 138
- dtUtil::FunctorCall< CallType, R, TYPELIST\_7(P1, P2, P3, P4, P5, P6, P7)>, 139
  - Call, 139
  - ParmsListType, 139
- dtUtil::FunTraits< R(\*)>, 140
  - ObjType, 140
  - ResultType, 140
  - TypeListType, 140
- dtUtil::FunTraits< R(\*)>(P1)>, 141
  - ObjType, 141
  - Parm1, 141
  - ResultType, 141
  - TYPELIST\_1, 141
- dtUtil::FunTraits< R(\*)>(P1, P2)>, 142
  - ObjType, 142
  - Parm1, 142
  - Parm2, 142
  - ResultType, 142
  - TYPELIST\_2, 142
- dtUtil::FunTraits< R(\*)>(P1, P2, P3)>, 143
  - ObjType, 143
  - Parm1, 143
  - Parm2, 143
  - Parm3, 143
  - ResultType, 143
  - TYPELIST\_3, 143
- dtUtil::FunTraits< R(\*)>(P1, P2, P3, P4)>, 144
  - ObjType, 144
  - Parm1, 144
  - Parm2, 144
  - Parm3, 144
  - Parm4, 144
  - ResultType, 144
  - TYPELIST\_4, 144
- dtUtil::FunTraits< R(\*)>(P1, P2, P3, P4, P5)>, 145
  - ObjType, 145
  - Parm1, 145
  - Parm2, 145
  - Parm3, 145
  - Parm4, 145
  - Parm5, 145
  - ResultType, 145
  - TYPELIST\_5, 145
- dtUtil::FunTraits< R(\*)>(P1, P2, P3, P4, P5, P6)>, 146
  - ObjType, 146
  - Parm1, 146
  - Parm2, 146
  - Parm3, 146
  - Parm4, 146
  - Parm5, 146
  - Parm6, 146
  - ResultType, 146
  - TYPELIST\_6, 146
- dtUtil::FunTraits< R(\*)>(P1, P2, P3, P4, P5, P6, P7)>, 147
  - ObjType, 147
  - Parm1, 147
  - Parm2, 147
  - Parm3, 147

- Parm4, 147
- Parm5, 147
- Parm6, 147
- Parm7, 147
- ResultType, 147
- TYPELIST\_7, 147
- dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5) const >, 148
  - ObjType, 148
  - ResultType, 148
  - TypeListType, 148
- dtUtil::FunTraits< R(O::\*)(P1) >, 149
  - ObjType, 149
  - ResultType, 149
  - TypeListType, 149
- dtUtil::FunTraits< R(O::\*)(P1) const >, 150
  - ObjType, 150
  - Parm1, 150
  - ResultType, 150
  - TYPELIST\_1, 150
- dtUtil::FunTraits< R(O::\*)(P1) >, 151
  - ObjType, 151
  - Parm1, 151
  - ResultType, 151
  - TYPELIST\_1, 151
- dtUtil::FunTraits< R(O::\*)(P1, P2) const >, 152
  - ObjType, 152
  - Parm1, 152
  - Parm2, 152
  - ResultType, 152
  - TYPELIST\_2, 152
- dtUtil::FunTraits< R(O::\*)(P1, P2) >, 153
  - ObjType, 153
  - Parm1, 153
  - Parm2, 153
  - ResultType, 153
  - TYPELIST\_2, 153
- dtUtil::FunTraits< R(O::\*)(P1, P2, P3) const >, 154
  - ObjType, 154
  - Parm1, 154
  - Parm2, 154
  - Parm3, 154
  - ResultType, 154
  - TYPELIST\_3, 154
- dtUtil::FunTraits< R(O::\*)(P1, P2, P3) >, 155
  - ObjType, 155
  - Parm1, 155
  - Parm2, 155
  - Parm3, 155
  - ResultType, 155
  - TYPELIST\_3, 155
- dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4) const >, 156
  - ObjType, 156
  - Parm1, 156
  - Parm2, 156
  - Parm3, 156
  - Parm4, 156
  - ResultType, 156
  - TYPELIST\_4, 156
- dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4) >, 157
  - ObjType, 157
  - Parm1, 157
  - Parm2, 157
  - Parm3, 157
  - Parm4, 157
  - ResultType, 157
  - TYPELIST\_4, 157
- dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5) const >, 158
  - Parm1, 158
  - Parm2, 158
  - Parm3, 158
  - Parm4, 158
  - Parm5, 158
  - ResultType, 158
  - TYPELIST\_5, 158
- dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5) >, 159
  - Parm1, 159
  - Parm2, 159
  - Parm3, 159
  - Parm4, 159
  - Parm5, 159
  - ResultType, 159
  - TYPELIST\_5, 159
- dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5, P6) const >, 160
  - Parm1, 160
  - Parm2, 160
  - Parm3, 160
  - Parm4, 160
  - Parm5, 160
  - Parm6, 160
  - ResultType, 160
  - TYPELIST\_6, 160
- dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5, P6) >, 161
  - Parm1, 161
  - Parm2, 161
  - Parm3, 161
  - Parm4, 161
  - Parm5, 161
  - Parm6, 161
  - ResultType, 161
  - TYPELIST\_6, 161
- dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5, P6, P7) const >, 162
  - Parm1, 162
  - Parm2, 162
  - Parm3, 162
  - Parm4, 162
  - Parm5, 162
  - Parm6, 162
  - Parm7, 162
  - ResultType, 162
  - TYPELIST\_7, 162
- dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5, P6, P7) >, 163
  - Parm1, 163
  - Parm2, 163
  - Parm3, 163
  - Parm4, 163
  - Parm5, 163
  - Parm6, 163
  - Parm7, 163
  - ResultType, 163
  - TYPELIST\_7, 163
- dtUtil::GeometryCollector, 164
  - apply, 164
  - GeometryCollector, 164
  - mGeomList, 164
- dtUtil::GroupVisitor, 166
  - apply, 166, 167
  - GroupVisitor, 166
- dtUtil::HotSpotDefinition, 168
  - mLocalRotation, 168
  - mLocalTranslation, 168

- mName, 168
- mParentName, 168
- dtUtil::HotSpotFileHandler, 169
  - ~HotSpotFileHandler, 170
  - characters, 170
  - DEFAULT\_VALUE, 170
  - endDocument, 170
  - endElement, 170
  - endPrefixMapping, 170
  - GetData, 170
  - HEADING\_VEC, 170
  - HOT\_SPOT\_NODE\_NAME, 170
  - HOT\_SPOT\_PARENT\_NODE\_NAME, 170
  - HotSpotDefinitionVector, 170
  - HotSpotFileHandler, 170
  - ignorableWhitespace, 170
  - LOCAL\_ROTATION\_NODE\_NAME, 170
  - LOCAL\_TRANSLATION\_NODE\_NAME, 170
  - NAME\_ATTRIBUTE\_NAME, 170
  - PITCH\_VEC, 170
  - processingInstruction, 170
  - ROLL\_VEC, 170
  - setDocumentLocator, 170
  - skippedEntity, 170
  - startDocument, 170
  - startElement, 170
  - startPrefixMapping, 170
- dtUtil::ldsFromTL, 171
  - Type, 171
- dtUtil::ldsFromTL< NullType, i >, 172
  - Type, 172
- dtUtil::IncomingCoordinateType, 173
  - GEOCENTRIC, 173
  - GEODETTIC, 173
  - UTM, 173
- dtUtil::insert\_back, 174
  - container, 174
  - container\_type, 174
  - input\_type, 174
  - insert\_back, 174
  - operator(), 174
- dtUtil::insert\_back\_no\_duplicates, 175
  - container, 175
  - container\_type, 175
  - insert\_back\_no\_duplicates, 175
  - operator(), 175
  - reference, 175
- dtUtil::InstantiateH< NullType, Holder, i >, 176
  - InstantiateH, 176
- dtUtil::InstantiateH< TypeList< T, U >, Holder, i >, 177
  - InstantiateH, 177
  - LeftBase, 177
  - ordern, 177
  - RightBase, 177
- dtUtil::InstantiateHAccessor< O, InstantiateH< TypeList< T, U >, Holder, i >, i >, 178
  - Get, 178
  - Instance, 178
  - TargetHolder, 178
- dtUtil::InstantiateHAccessor< j, InstantiateH< TypeList< T, U >, Holder, i >, i >, 179
  - Get, 179
  - Instance, 179
  - RightBase, 179
  - TargetHolder, 179
- dtUtil::Int2Type, 180
  - value, 180
- dtUtil::InternalLibraryHandle, 181
  - ~InternalLibraryHandle, 181
  - FindSymbol, 181
  - GetHandle, 181
  - GetLibName, 181
  - InternalLibraryHandle, 181
  - LoadSharedLibrary, 181
- dtUtil::IsConst, 182
  - IS\_CONST, 182
- dtUtil::IsDelimiter, 183
  - IsDelimiter, 183
  - operator(), 183
- dtUtil::IsIntType, 184
  - value, 184
- dtUtil::IsIntType< Int2Type< i >, i >, 185
  - value, 185
- dtUtil::IsPointer, 186
  - IS\_POINTER, 186
- dtUtil::IsReference, 187
  - IS\_REFERENCE, 187
- dtUtil::IsSlash, 188
  - operator(), 188
- dtUtil::IsSpace, 189
  - DEFAULT\_LOCALE\_NAME, 189
  - GetLocale, 189
  - IsSpace, 189
  - operator(), 189
- dtUtil::KDTree, 190
  - ~KDTree, 194
  - \_Base, 193
  - \_Base\_const\_ptr, 193
  - \_Base\_ptr, 193
  - \_Link\_const\_type, 193
  - \_Link\_type, 193
  - \_M\_acc, 195
  - \_M\_check\_children, 194
  - \_M\_check\_node, 194
  - \_M\_cmp, 195
  - \_M\_count, 195
  - \_M\_count\_within\_range, 194
  - \_M\_delete\_node, 194
  - \_M\_dist, 195
  - \_M\_empty\_initialise, 194
  - \_M\_erase, 194
  - \_M\_erase\_subtree, 194
  - \_M\_find, 194
  - \_M\_find\_exact, 194
  - \_M\_find\_within\_range, 194
  - \_M\_get\_erase\_replacement, 194
  - \_M\_get\_j\_max, 194
  - \_M\_get\_j\_min, 194
  - \_M\_get\_leftmost, 194
  - \_M\_get\_rightmost, 194
  - \_M\_get\_root, 194
  - \_M\_header, 195
  - \_M\_insert, 194
  - \_M\_insert\_left, 194
  - \_M\_insert\_right, 194
  - \_M\_matches\_node, 194
  - \_M\_matches\_node\_in\_d, 194
  - \_M\_matches\_node\_in\_other\_ds, 194
  - \_M\_new\_node, 194
  - \_M\_optimise, 194
  - \_M\_root, 195
  - \_M\_set\_leftmost, 194

- \_M\_set\_rightmost, 194
- \_M\_set\_root, 194
- \_M\_visit\_within\_range, 194
- \_Node\_compare\_, 193
- \_Region\_, 193
- \_S\_is\_leaf, 194
- \_S\_left, 194
- \_S\_maximum, 194
- \_S\_minimum, 194
- \_S\_parent, 194
- \_S\_right, 194
- \_S\_set\_left, 194
- \_S\_set\_parent, 194
- \_S\_set\_right, 194
- \_S\_value, 194
- allocator\_type, 193
- begin, 194
- check\_tree, 194
- clear, 194
- const\_iterator, 193
- const\_pointer, 194
- const\_reference, 194
- const\_reverse\_iterator, 194
- count\_within\_range, 194
- difference\_type, 194
- distance\_type, 194
- efficient\_replace\_and\_optimize, 194
- empty, 194
- end, 194
- erase, 194
- erase\_exact, 194
- find, 194
- find\_exact, 194
- find\_nearest, 194
- find\_nearest\_if, 194
- find\_within\_range, 194
- find\_within\_range\_iterative, 194
- get\_allocator, 194
- insert, 194, 195
- iterator, 194
- KDTree, 194
- max\_size, 195
- operator=, 195
- optimize, 195
- pointer, 194
- rbegin, 195
- reference, 194
- rend, 195
- reverse\_iterator, 194
- size, 195
- size\_type, 194
- subvalue\_type, 194
- value\_acc, 195
- value\_comp, 195
- value\_distance, 195
- value\_type, 194
- visit\_within\_range, 195
- dtUtil::KeyFrameDecoder, 196
  - ~KeyFrameDecoder, 197
  - DecodeFrameStamp, 197
  - DecodeSourceData, 197
  - FrameDataPtrContainer, 196
  - FrameDataType, 196
  - KeyFrame, 197
  - KeyFrameContainer, 197
  - KeyFrameDecoder, 197
  - RecordablePtrContainer, 197
  - RecordableType, 197
  - Walk, 197
- dtUtil::Length< NullType >, 198
  - value, 198
- dtUtil::Length< TypeList< T, U > >, 199
  - value, 199
- dtUtil::LibrarySharingManager, 202
  - AddToSearchPath, 202
  - ClearSearchPath, 202
  - FindLibraryInSearchPath, 203
  - GetInstance, 203
  - GetPlatformIndependentLibraryName, 203
  - GetPlatformSpecificLibraryName, 203
  - GetSearchPath, 203
  - LoadSharedLibrary, 203
  - RemoveFromSearchPath, 203
- dtUtil::LibrarySharingManager::ExceptionEnum, 118
  - ExceptionEnum, 118
  - LibraryLoadingError, 118
- dtUtil::LibrarySharingManager::LibraryHandle, 200
  - ~LibraryHandle, 200
  - FindSymbol, 200
  - GetHandle, 200
  - GetLibName, 200
  - HANDLE, 200
  - IsShuttingDown, 200
  - LibraryHandle, 200
  - release, 201
  - SYMBOL\_ADDRESS, 200
- dtUtil::LocalCoordinateType, 204
  - CARTESIAN, 204
  - CARTESIAN\_FLAT\_EARTH, 204
  - CARTESIAN\_UTM, 204
  - GLOBE, 204
- dtUtil::Log, 205
  - ~Log, 206
  - GetInstance, 206
  - GetLogLevel, 206
  - GetLogLevelForString, 206
  - GetLogLevelString, 207
  - GetName, 207
  - GetOutputStreamBit, 207
  - IsLevelEnabled, 207
  - Log, 206
  - LOG\_ALWAYS, 206
  - LOG\_DEBUG, 206
  - LOG\_ERROR, 206
  - LOG\_INFO, 206
  - LOG\_WARNING, 206
  - LogHorizRule, 207
  - LogMessage, 207, 208
  - LogMessageType, 206
  - NO\_OUTPUT, 206
  - OutputStreamOptions, 206
  - SetLogLevel, 208
  - SetOutputStreamBit, 208
  - STANDARD, 206
  - TO\_CONSOLE, 206
  - TO\_FILE, 206
- dtUtil::LogFile, 209
  - GetFileName, 209
  - GetTitle, 209
  - SetFileName, 209
  - SetTitle, 209
- dtUtil::LogImpl, 210

- LogImpl, 210
- mDefaultName, 210
- mName, 210
- mOutputStreamBit, 210
- dtUtil::LogManager, 211
  - ~LogManager, 211
  - AddInstance, 211
  - EndFile, 211
  - GetInstance, 211
  - logFile, 211
  - LogManager, 211
  - mMutex, 211
  - OpenFile, 211
  - TimeTag, 211
- dtUtil::MatrixUtil, 212
  - ClampUnity, 212
  - GetRow3, 212
  - GetRow4, 213
  - HprToMatrix, 213
  - MatrixToHpr, 213
  - MatrixToHprAndPosition, 213
  - PositionAndHprToMatrix, 213
  - Print, 213
  - SetRow, 213, 214
  - TransformVec3, 214
  - Transpose, 214
- dtUtil::MergeParmsH, 215
  - MergeParms, 215
  - ResultType, 215
- dtUtil::MergeParmsH< 0, BoundPTL, UnboundPTL, dtUtil::NullType, TL >, 216
  - MergeParms, 216
- dtUtil::MergeParmsH< i, BoundPTL, UnboundPTL, IdsTL, dtUtil::NullType >, 217
  - MergeParms, 217
- dtUtil::MergeParmsH::Bound, 57
  - MergeParms, 57
- dtUtil::MergeParmsH::Predicate, 245
  - value, 245
- dtUtil::MergeParmsH::Predicate< j, dtUtil::NullType >, 246
  - value, 246
- dtUtil::MergeParmsH::Unbound, 300
  - MergeParms, 300
- dtUtil::NodeCollector, 218
  - ~NodeCollector, 221
  - AddDOFTransform, 222
  - AddGeode, 222
  - AddGroup, 222
  - AddLOD, 222
  - AddMatrixTransform, 222
  - AddMultiSwitch, 222
  - AddSwitch, 222
  - AllNodeTypes, 226
  - ClearAll, 222
  - CollectNodes, 222
  - DOFTransformFlag, 226
  - GeodeFlag, 226
  - GeodeNodeMap, 221
  - GetDOFTransform, 223
  - GetGeode, 223
  - GetGeodeNodeMap, 223
  - GetGroup, 223
  - GetGroupNodeMap, 224
  - GetLOD, 224
  - GetLODNodeMap, 224
  - GetMatrixTransform, 224
  - GetMatrixTransformNodeMap, 224
  - GetMultiSwitch, 224, 225
  - GetMultiSwitchNodeMap, 225
  - GetSwitch, 225
  - GetSwitchNodeMap, 225
  - GetTransformNodeMap, 225
  - GroupFlag, 226
  - GroupNodeMap, 221
  - LODFlag, 226
  - LODNodeMap, 221
  - MatrixTransformFlag, 227
  - MatrixTransformNodeMap, 221
  - MultiSwitchFlag, 227
  - MultiSwitchNodeMap, 221
  - NodeCollector, 221
  - NodeFlag, 221
  - RemoveDOFTransform, 225
  - RemoveGeode, 225
  - RemoveGroup, 226
  - RemoveLOD, 226
  - RemoveMatrixTransform, 226
  - RemoveMultiSwitch, 226
  - RemoveSwitch, 226
  - SwitchFlag, 227
  - SwitchNodeMap, 221
  - TransformNodeMap, 221
- dtUtil::NodePrintOut, 228
  - ~NodePrintOut, 228
  - Analyze, 228
  - AnalyzeGeode, 228
  - AnalyzePrimSet, 228
  - CollectNodeData, 229
  - GetFileOutput, 229
  - NodePrintOut, 228
  - PrintNodeToOSGFile, 229
- dtUtil::Noise1, 230
  - ~Noise1, 230
  - GetNoise, 230
  - Noise1, 230
  - Reseed, 230
- dtUtil::Noise2, 231
  - ~Noise2, 231
  - GetNoise, 231
  - Noise2, 231
  - Reseed, 231
- dtUtil::Noise3, 232
  - ~Noise3, 232
  - GetNoise, 232
  - Noise3, 232
  - Reseed, 232
- dtUtil::NoiseTexture, 233
  - ~NoiseTexture, 234
  - GetNoiseTexture, 234
  - MakeNoiseTexture, 234
  - NoiseTexture, 233
  - SetAmplitude, 234
  - SetFrequency, 234
  - SetHeight, 234
  - SetOctaves, 234
  - SetPersistence, 234
  - SetSlices, 234
  - SetWidth, 234
- dtUtil::NotIntType, 237
  - value, 237
- dtUtil::NullType, 238
- dtUtil::ObjectFactory, 239

- ~ObjectFactory, 240
- BaseType, 240
- CreateObject, 240
- createObjectFunc, 240
- GetMap, 240
- GetSupportedTypes, 240
- IsTypeSupported, 240
- ItCmp, 240
- ObjectFactory, 240
- ObjectMap, 240
- ObjTypeltor, 240
- ObjTypeltorConst, 240
- RegisterType, 240
- RemoveType, 240
- UniqueIdType, 240
- dtUtil::Packager, 241
  - ~Packager, 241
  - AddFile, 242
  - ClosePackage, 242
  - FindPackDataForPath, 242
  - GetPackTree, 242
  - OpenPackage, 242
  - Packager, 241
  - PackPackage, 242
  - RemoveFile, 242
  - UnpackPackage, 242
- dtUtil::Packager::PackTreeData, 243
  - files, 243
  - folders, 243
  - isFromPack, 243
  - name, 243
  - parent, 243
  - seekPos, 243
  - source, 243
- dtUtil::PolarDecomp, 244
  - Decompose, 244
- dtUtil::RefString, 247
  - ~RefString, 247
  - c\_str, 247
  - Get, 247
  - GetSharedStringCount, 247
  - operator const std::string &, 247
  - operator<, 248
  - operator+, 248
  - operator->, 248
  - operator=, 248
  - operator==, 248
  - operator[], 248
  - RefString, 247
- dtUtil::ResourceLoader, 249
  - ~ResourceLoader, 249
  - FreeResource, 249
  - LoadResource, 249
- dtUtil::ResourceManager, 250
  - ~ResourceManager, 250
  - AddResource, 250
  - FreeAll, 250
  - FreeResource, 250
  - GetResource, 250
  - LoadResource, 250
  - ResourceConstIterator, 250
  - ResourceHandle, 250
  - ResourceIterator, 250
  - ResourceManager, 250
  - ResourceMap, 250
  - SetResourceLoader, 250
- dtUtil::SeamlessNoise, 251
  - ~SeamlessNoise, 251
  - GetNoise, 251
  - Reseed, 251
  - SeamlessNoise, 251
  - SetRepeat, 251
- dtUtil::Select, 253
  - Result, 253
- dtUtil::Select< false, T, U >, 254
  - Result, 254
- dtUtil::Serializer, 255
  - ToBool, 255
  - ToDouble, 255
  - ToFloat, 255
  - ToInt, 256
  - ToString, 256
- dtUtil::squared\_difference, 257
  - distance\_type, 257
  - operator(), 257
- dtUtil::squared\_difference\_counted, 258
  - count, 258
  - distance\_type, 258
  - operator(), 258
  - reset, 258
  - squared\_difference\_counted, 258
- dtUtil::StateAttributeCollector, 259
  - ~StateAttributeCollector, 260
  - AddMaterial, 261
  - AddProgram, 261
  - AddTexture, 261
  - AllAttributes, 263
  - ClearAll, 261
  - CollectStateAttributes, 261
  - GetMaterial, 261
  - GetMaterialMap, 261
  - GetProgram, 262
  - GetProgramMap, 262
  - GetTexture, 262
  - GetTextureMap, 262
  - MaterialFlag, 263
  - MaterialNodeMap, 260
  - ProgramFlag, 263
  - ProgramNodeMap, 260
  - StateAttribFlag, 260
  - StateAttributeCollector, 260
  - TextureFlag, 263
  - TextureNodeMap, 260
- dtUtil::StateVisitor, 264
  - apply, 264
  - stateSetParser, 264
  - StateVisitor, 264
- dtUtil::StringTokenizer, 265
  - tokenize, 265
- dtUtil::StringToXMLConverter, 266
  - ~StringToXMLConverter, 266
  - StringToXMLConverter, 266
  - ToXmlString, 266
- dtUtil::TailAt< InstantiateH< TypeList< T, U >, Holder, i >, 0, i >, 267
  - Result, 267
- dtUtil::TailAt< InstantiateH< TypeList< T, U >, Holder, i >, j, i >, 268
  - Result, 268
- dtUtil::TangentSpaceVisitor, 269
  - apply, 269
  - TangentSpaceVisitor, 269

- dtUtil::ToLowerClass, 281
  - operator(), 281
  - ToLowerClass, 281
- dtUtil::Transformation, 282
  - operator(), 282
- dtUtil::tree, 283
  - ~tree, 284
  - begin, 284
  - clear, 284
  - const\_iterator, 284
  - copy\_tree, 284
  - data, 284
  - end, 284
  - erase, 285
  - find, 285
  - get\_tree\_iterator, 285
  - in, 285
  - insert, 285
  - iterator, 284
  - level, 285
  - next, 285
  - operator\*, 285
  - operator=, 285
  - operator==, 285
  - operator[], 285
  - out, 285, 286
  - prev, 286
  - push\_back, 286
  - push\_front, 286
  - reinsert, 286
  - remove, 286
  - size, 286
  - tree, 284
  - tree\_find\_breadth, 286
  - tree\_find\_depth, 286
- dtUtil::tree\_iterator, 287
  - ~tree\_iterator, 288
  - begin, 288
  - clear\_children, 288
  - clear\_tree, 288
  - const\_iterator, 288
  - data, 288
  - end, 288
  - end\_iterator, 289
  - find, 289
  - in, 289
  - insert, 289
  - iterator, 288
  - level, 289
  - next, 289
  - operator\*, 289
  - operator++, 289
  - operator->, 289
  - operator--, 289
  - operator=, 289
  - operator==, 289
  - operator[], 289
  - out, 290
  - push\_back, 290
  - push\_front, 290
  - reinsert, 290
  - remove, 290
  - size, 290
  - tree\_find\_breadth, 290
  - tree\_find\_depth, 290
  - tree\_iterator, 288
  - tree\_ptr, 290
  - tree\_ref, 290
- dtUtil::TupleHolder, 291
  - operator=, 291
  - StoredType, 291
  - TupleHolder, 291
  - Type, 291
  - value, 291
- dtUtil::TypeAt< TypeList< T, U >, 0 >, 292
  - Result, 292
- dtUtil::TypeAt< TypeList< T, U >, i >, 293
  - Result, 293
- dtUtil::TypeAtNonStrict, 294
  - Result, 294
- dtUtil::TypeAtNonStrict< TypeList< T, U >, 0, DefType >, 295
  - Result, 295
- dtUtil::TypeAtNonStrict< TypeList< T, U >, i, DefType >, 296
  - Result, 296
- dtUtil::TypeList, 297
  - Head, 297
  - Tail, 297
- dtUtil::TypeTraits, 299
  - const\_param\_type, 299
  - const\_reference, 299
  - const\_return\_type, 299
  - param\_type, 299
  - pointer\_type, 299
  - reference, 299
  - return\_type, 299
  - value\_type, 299
- dtUtil::UnboundHelper, 301
  - IncomingTL, 301
  - Parm1, 301
  - Parm2, 301
  - Parm3, 301
  - Parm4, 301
  - Parm5, 301
  - UnboundTL, 301
- dtUtil::UnboundTL2, 302
  - Result, 302
- dtUtil::UTMParameters, 303
  - CalcTransverseMercatorParameters, 304
  - DENOM, 304
  - SPHSN, 304
  - SPHSR, 304
  - SPHTMD, 304
  - TranMerc\_a, 304
  - TranMerc\_ap, 304
  - TranMerc\_bp, 304
  - TranMerc\_cp, 304
  - TranMerc\_Delta\_Easting, 304
  - TranMerc\_Delta\_Northing, 304
  - TranMerc\_dp, 304
  - TranMerc\_ebs, 304
  - TranMerc\_ep, 304
  - TranMerc\_es, 304
  - TranMerc\_f, 304
  - TranMerc\_False\_Easting, 304
  - TranMerc\_False\_Northing, 304
  - TranMerc\_Origin\_Lat, 304
  - TranMerc\_Origin\_Long, 304
  - TranMerc\_Scale\_Factor, 304
  - UTMParameters, 304
- dtUtil::XercesErrorHandler, 306
  - ~XercesErrorHandler, 306
  - error, 306

- fatalError, 306
- resetErrors, 306
- warning, 306
- XercesErrorHandler, 306
- dtUtil::XercesParser, 307
  - ~XercesParser, 307
  - Parse, 307
  - XercesParser, 307
- dtUtil::XercesWriter, 308
  - ~XercesWriter, 308
  - CreateDocument, 308
  - GetDocument, 308
  - WriteFile, 308
  - XercesWriter, 308
- dtUtil::XMLStringConverter, 309
  - ~XMLStringConverter, 309
  - c\_str, 309
  - ToString, 309
  - XMLStringConverter, 309
- E -**
- EdgeStepFilter
  - dtUtil::EdgeStepFilter, 108
- efficient\_replace\_and\_optimize
  - dtUtil::KDTree, 194
- empty
  - dtUtil::KDTree, 194
- encloses
  - dtUtil::\_Region, 47
- END
  - dtUtil::DataStream::SeekTypeEnum, 252
- end
  - dtUtil::KDTree, 194
  - dtUtil::tree, 284
  - dtUtil::tree\_iterator, 288
- end\_iterator
  - dtUtil::tree\_iterator, 289
- endDocument
  - dtUtil::HotSpotFileHandler, 170
- endElement
  - dtUtil::HotSpotFileHandler, 170
- EndFile
  - dtUtil::LogManager, 211
- endPrefixMapping
  - dtUtil::HotSpotFileHandler, 170
- Enumeration
  - dtUtil::Enumeration, 111
- enumeration.cpp, 328
- enumeration.h, 329
  - DECLARE\_ENUM, 329
  - IMPLEMENT\_ENUM, 330
- Equivalent
  - dtUtil, 26
- erase
  - dtUtil::KDTree, 194
  - dtUtil::tree, 285
- erase\_exact
  - dtUtil::KDTree, 194
- error
  - dtUtil::XercesErrorHandler, 306
- EulersToMatrix
  - dtUtil::Coordinates, 83
- Exception
  - dtUtil::Exception, 116
- exception.cpp, 331
- exception.h, 332
- ExceptionEnum
  - dtUtil::LibrarySharingManager::ExceptionEnum, 118
- export.h, 333
  - DT\_UTIL\_EXPORT, 333
- F -**
- fatalError
  - dtUtil::XercesErrorHandler, 306
- FBM
  - dtUtil::Fractal, 128
- File
  - dtUtil::Exception, 117
- FILE\_NOT\_FOUND
  - dtUtil, 23
- FileCopy
  - dtUtil::FileUtils, 123
- FileDelete
  - dtUtil::FileUtils, 123
- FileExceptionEnum
  - dtUtil::FileExceptionEnum, 119
- FileExists
  - dtUtil::FileUtils, 124
- FileExtensionList
  - dtUtil, 23
- FileInfo
  - dtUtil::FileInfo, 120
- FileMove
  - dtUtil::FileUtils, 124
- fileName
  - dtUtil::FileInfo, 120
- FileNotFound
  - dtUtil::FileExceptionEnum, 119
- files
  - dtUtil::Packager::PackTreeData, 243
- FileType
  - dtUtil, 23
- fileType
  - dtUtil::FileInfo, 120
- fileutils.cpp, 334
  - MAX\_PATH, 334
  - S\_ISDIR, 334
  - S\_ISREG, 334
  - stat64, 334
- fileutils.h, 335
- find
  - dtUtil::KDTree, 194
  - dtUtil::tree, 285
  - dtUtil::tree\_iterator, 289
- find\_exact
  - dtUtil::KDTree, 194
- find\_nearest
  - dtUtil::KDTree, 194
- find\_nearest\_if
  - dtUtil::KDTree, 194
- find\_within\_range
  - dtUtil::KDTree, 194
- find\_within\_range\_iterative
  - dtUtil::KDTree, 194
- FindAttributeValueFor
  - dtUtil, 27
- FindLibraryInSearchPath
  - dtUtil::LibrarySharingManager, 203
- FindNodePointer
  - dtUtil::CollectorUtil, 33
- FindPackDataForPath
  - dtUtil::Packager, 242

- FindSymbol
  - dtUtil::InternalLibraryHandle, 181
  - dtUtil::LibrarySharingManager::LibraryHandle, 200
- flatteningReciprocal
  - dtUtil, 30
- folders
  - dtUtil::Packager::PackTreeData, 243
- fractal.h, 336
- Fractal2d
  - dtUtil, 23
- Fractal2f
  - dtUtil, 23
- Fractal3d
  - dtUtil, 23
- Fractal3f
  - dtUtil, 23
- FrameDataPtrContainer
  - dtUtil::KeyFrameDecoder, 196
- FrameDataType
  - dtUtil::KeyFrameDecoder, 196
- FreeAll
  - dtUtil::ResourceManager, 250
- FreeResource
  - dtUtil::ResourceLoader, 249
  - dtUtil::ResourceManager, 250
- funbind.h, 337
- funcall.h, 339
- Functor
  - dtUtil::Functor, 131
- functor.h, 340
  - DoCall, 340
- FunctorType
  - dtUtil::Command0, 71
  - dtUtil::Command1, 72
  - dtUtil::Command2, 74
- funtraits.h, 341
- G -**
- GENERAL\_EXCEPTION
  - dtUtil::BaseExceptionType, 54
- generic.h, 342
  - TYPelist\_0, 343
  - TYPelist\_1, 343
  - TYPelist\_2, 343
  - TYPelist\_3, 343
  - TYPelist\_4, 343
  - TYPelist\_5, 343
  - TYPelist\_6, 343
  - TYPelist\_7, 343
  - TYPelist\_8, 343
- Geocent\_a
  - dtUtil, 30
- Geocent\_e2
  - dtUtil, 31
- Geocent\_e2\_2
  - dtUtil, 31
- Geocent\_e2\_3
  - dtUtil, 31
- Geocent\_ef
  - dtUtil, 31
- Geocent\_ef\_3
  - dtUtil, 31
- Geocent\_ef\_4
  - dtUtil, 31
- Geocent\_ep2
  - dtUtil, 31
- Geocent\_f
  - dtUtil, 31
- GEOCENTRIC
  - dtUtil::IncomingCoordinateType, 173
- GeocentricToGeodetic
  - dtUtil::Coordinates, 83
- GeodeFlag
  - dtUtil::NodeCollector, 226
- GeodeNodeMap
  - dtUtil::NodeCollector, 221
- GEODETTIC
  - dtUtil::IncomingCoordinateType, 173
- GeodeticToGeocentric
  - dtUtil::Coordinates, 83
- GeometryCollector
  - dtUtil::GeometryCollector, 164
- geometrycollector.h, 344
- Get
  - dtUtil::InstantiateHAccessor< 0, InstantiateH< TypeList< T, U >, Holder, i >, i >, 178
  - dtUtil::InstantiateHAccessor< j, InstantiateH< TypeList< T, U >, Holder, i >, i >, 179
  - dtUtil::RefString, 247
- get
  - dtUtil::\_Alloc\_base::NoLeakAlloc, 236
- get\_allocator
  - dtUtil::\_Alloc\_base, 38
  - dtUtil::KDTTree, 194
- get\_raw\_node
  - dtUtil::\_Iterator, 42
- get\_tree\_iterator
  - dtUtil::tree, 285
- GetAbsolutePath
  - dtUtil::FileUtils, 124
- GetApplyRotationConversionMatrix
  - dtUtil::Coordinates, 83
- GetBuffer
  - dtUtil::DataStream, 92
- GetBufferCapacity
  - dtUtil::DataStream, 92
- GetBufferSize
  - dtUtil::DataStream, 92
- GetConfigPropertyValue
  - dtUtil::ConfigProperties, 76
- GetData
  - dtUtil::HotSpotFileHandler, 170
- GetDay
  - dtUtil::DateTime, 98
- GetDocument
  - dtUtil::XercesWriter, 308
- GetDOFTransform
  - dtUtil::NodeCollector, 223
- getFileExtensionIncludingDot
  - dtUtil, 27
- GetFileInfo
  - dtUtil::FileUtils, 124
- GetFileName
  - dtUtil::LogFile, 209
- GetFileOutput
  - dtUtil::NodePrintOut, 229
- GetFlatEarthOrigin
  - dtUtil::Coordinates, 84
- GetGeode
  - dtUtil::NodeCollector, 223
- GetGeodeNodeMap
  - dtUtil::NodeCollector, 223

- GetGlobeRadius
  - dtUtil::Coordinates, 84
- GetGMTOffset
  - dtUtil::DateTime, 99
- GetGMTTime
  - dtUtil::DateTime, 99
- GetGroup
  - dtUtil::NodeCollector, 223
- GetGroupNodeMap
  - dtUtil::NodeCollector, 224
- GetH
  - dtUtil, 27
- GetHandle
  - dtUtil::InternalLibraryHandle, 181
  - dtUtil::LibrarySharingManager::LibraryHandle, 200
- GetHour
  - dtUtil::DateTime, 99
- GetIncomingCoordinateType
  - dtUtil::Coordinates, 84
- GetInstance
  - DeprecationMgr, 104
  - dtUtil::FileUtils, 124
  - dtUtil::LibrarySharingManager, 203
  - dtUtil::Log, 206
  - dtUtil::LogManager, 211
- GetLibName
  - dtUtil::InternalLibraryHandle, 181
  - dtUtil::LibrarySharingManager::LibraryHandle, 200
- GetLocalCoordinateType
  - dtUtil::Coordinates, 84
- GetLocale
  - dtUtil::IsSpace, 189
- GetLocalGMTOffset
  - dtUtil::DateTime, 99
- GetLocalOffset
  - dtUtil::Coordinates, 84
- GetLOD
  - dtUtil::NodeCollector, 224
- GetLODNodeMap
  - dtUtil::NodeCollector, 224
- GetLogLevel
  - dtUtil::Log, 206
- GetLogLevelForString
  - dtUtil::Log, 206
- GetLogLevelString
  - dtUtil::Log, 207
- GetMagneticNorthOffset
  - dtUtil::Coordinates, 84
- GetMap
  - dtUtil::ObjectFactory, 240
- GetMaterial
  - dtUtil::StateAttributeCollector, 261
- GetMaterialMap
  - dtUtil::StateAttributeCollector, 261
- GetMatrixTransform
  - dtUtil::NodeCollector, 224
- GetMatrixTransformNodeMap
  - dtUtil::NodeCollector, 224
- GetMinute
  - dtUtil::DateTime, 99
- GetMonth
  - dtUtil::DateTime, 99
- GetMultiSwitch
  - dtUtil::NodeCollector, 224, 225
- GetMultiSwitchNodeMap
  - dtUtil::NodeCollector, 225
- GetName
  - dtUtil::Enumeration, 111
  - dtUtil::Log, 207
- GetNoise
  - dtUtil::Noise1, 230
  - dtUtil::Noise2, 231
  - dtUtil::Noise3, 232
  - dtUtil::SeamlessNoise, 251
- GetNoiseTexture
  - dtUtil::NoiseTexture, 234
- GetOriginRotationMatrix
  - dtUtil::Coordinates, 84
- GetOriginRotationMatrixInverse
  - dtUtil::Coordinates, 84
- GetOutputStreamBit
  - dtUtil::Log, 207
- GetPackTree
  - dtUtil::Packager, 242
- GetPlatformIndependentLibraryName
  - dtUtil::LibrarySharingManager, 203
- GetPlatformSpecificLibraryName
  - dtUtil::LibrarySharingManager, 203
- GetProgram
  - dtUtil::StateAttributeCollector, 262
- GetProgramMap
  - dtUtil::StateAttributeCollector, 262
- GetRemainingReadSize
  - dtUtil::DataStream, 92
- GetResource
  - dtUtil::ResourceManager, 250
- GetRow3
  - dtUtil::MatrixUtil, 212
- GetRow4
  - dtUtil::MatrixUtil, 213
- GetSearchPath
  - dtUtil::LibrarySharingManager, 203
- GetSecond
  - dtUtil::DateTime, 99
- GetSharedStringCount
  - dtUtil::RefString, 247
- GetSupportedTypes
  - dtUtil::ObjectFactory, 240
- GetSwitch
  - dtUtil::NodeCollector, 225
- GetSwitchNodeMap
  - dtUtil::NodeCollector, 225
- GetTexture
  - dtUtil::StateAttributeCollector, 262
- GetTextureMap
  - dtUtil::StateAttributeCollector, 262
- GetTime
  - dtUtil::DateTime, 99, 100
- GetTimeFormat
  - dtUtil::DateTime, 100
- GetTimeInSeconds
  - dtUtil::DateTime, 100
- GetTimeOrigin
  - dtUtil::DateTime, 100
- GetTimeScale
  - dtUtil::DateTime, 100
- GetTimeType
  - dtUtil::DateTime, 100
- GetTitle
  - dtUtil::LogFile, 209
- GetTransformNodeMap
  - dtUtil::NodeCollector, 225

- GetUTMHemisphere
  - dtUtil::Coordinates, 84
- GetUTMZone
  - dtUtil::Coordinates, 84
- GetYear
  - dtUtil::DateTime, 100
- GLOBE
  - dtUtil::LocalCoordinateType, 204
- GMT\_TIME
  - dtUtil::DateTime::TimeOrigin, 279
- GroupFlag
  - dtUtil::NodeCollector, 226
- GroupNodeMap
  - dtUtil::NodeCollector, 221
- GroupVisitor
  - dtUtil::GroupVisitor, 166
- gStringSetMutex
  - dtUtil, 31
- H -**
- HANDLE
  - dtUtil::LibrarySharingManager::LibraryHandle, 200
- Has
  - dtUtil::Bits, 32
- Head
  - dtUtil::TypeList, 297
- HEADING\_VEC
  - dtUtil::HotSpotFileHandler, 170
- HeteroFractal
  - dtUtil::Fractal, 128
- HOT\_SPOT\_NODE\_NAME
  - dtUtil::HotSpotFileHandler, 170
- HOT\_SPOT\_PARENT\_NODE\_NAME
  - dtUtil::HotSpotFileHandler, 170
- hotspotdefinition.h, 345
- HotSpotDefinitionVector
  - dtUtil::HotSpotFileHandler, 170
- HotSpotFileHandler
  - dtUtil::HotSpotFileHandler, 170
- hotspotxml.cpp, 346
- hotspotxml.h, 347
- HprToMatrix
  - dtUtil::MatrixUtil, 213
- I -**
- ignorableWhitespace
  - dtUtil::HotSpotFileHandler, 170
- iMakeDirectory
  - dtUtil, 27
- IMPLEMENT\_ENUM
  - dtUtil, 27
  - enumeration.h, 330
- IMPLEMENT\_MANAGEMENT\_LAYER
  - macros.h, 356
- in
  - dtUtil::tree, 285
  - dtUtil::tree\_iterator, 289
- inc/ Directory Reference, 10
- inc/dtUtil/ Directory Reference, 8
- IncomingTL
  - dtUtil::BoundHelper, 58
  - dtUtil::UnboundHelper, 301
- IncreaseBufferSize
  - dtUtil::DataStream, 92
- IncrementClock
  - dtUtil::DateTime, 100
- input\_type
  - dtUtil::insert\_back, 174
- insert
  - dtUtil::KDTree, 194, 195
  - dtUtil::tree, 285
  - dtUtil::tree\_iterator, 289
- insert\_back
  - dtUtil::insert\_back, 174
- insert\_back\_no\_duplicates
  - dtUtil::insert\_back\_no\_duplicates, 175
- Instance
  - dtUtil::InstantiateHAccessor< 0, InstantiateH< TypeList< T, U >, Holder, i >, i >, 178
  - dtUtil::InstantiateHAccessor< j, InstantiateH< TypeList< T, U >, Holder, i >, i >, 179
- InstantiateH
  - dtUtil::InstantiateH< NullType, Holder, i >, 176
  - dtUtil::InstantiateH< TypeList< T, U >, Holder, i >, 177
- InternalLibraryHandle
  - dtUtil::InternalLibraryHandle, 181
- intersects\_with
  - dtUtil::\_Region, 47
- INVALID\_INPUT
  - dtUtil::CoordinateConversionExceptionEnum, 77
- IOException
  - dtUtil::FileExceptionEnum, 119
- IS\_A
  - macros.h, 356
- IS\_CONST
  - dtUtil::IsConst, 182
- IS\_POINTER
  - dtUtil::IsPointer, 186
- IS\_REFERENCE
  - dtUtil::IsReference, 187
- IsDelimiter
  - dtUtil::IsDelimiter, 183
- IsFinite
  - dtUtil, 27
- IsFiniteVec
  - dtUtil, 27
- isFromPack
  - dtUtil::Packager::PackTreeData, 243
- IslandFractal
  - dtUtil::Fractal, 128
- IsLevelEnabled
  - dtUtil::Log, 207
- IsLittleEndian
  - dtUtil::DataStream, 92
- IsNaN
  - dtUtil, 27
- IsSameFile
  - dtUtil::FileUtils, 124
- IsShuttingDown
  - dtUtil::LibrarySharingManager::LibraryHandle, 200
- IsSpace
  - dtUtil::IsSpace, 189
- IsTypeSupported
  - dtUtil::ObjectFactory, 240
- iterator
  - dtUtil::\_Iterator, 42
  - dtUtil::KDTree, 194
  - dtUtil::tree, 284
  - dtUtil::tree\_iterator, 288
- iterator\_category

- dtUtil::\_Iterator, 42
- K -**
- KDTree
  - dtUtil::\_Base\_iterator, 39
  - dtUtil::KDTree, 194
- kdtree.h, 348
  - KDTREE\_LIB\_VERSION, 349
  - KDTREE\_VERSION, 349
- KDTREE\_LIB\_VERSION
  - kdtree.h, 349
- KDTREE\_VERSION
  - kdtree.h, 349
- KeyFrame
  - dtUtil::KeyFrameDecoder, 197
- KeyFrameContainer
  - dtUtil::KeyFrameDecoder, 197
- KeyFrameDecoder
  - dtUtil::KeyFrameDecoder, 197
- keyframedecoder.h, 350
- L -**
- lastModified
  - dtUtil::FileInfo, 120
- LeftBase
  - dtUtil::InstantiateH< TypeList< T, U >, Holder, i >, 177
- LERP
  - seamlessnoise.cpp, 383
- Lerp
  - dtUtil, 27
- level
  - dtUtil::tree, 285
  - dtUtil::tree\_iterator, 289
- LEXICAL\_DATE\_FORMAT
  - dtUtil::DateTime::TimeFormat, 278
- LibraryHandle
  - dtUtil::LibrarySharingManager::LibraryHandle, 200
- LibraryLoadingError
  - dtUtil::LibrarySharingManager::ExceptionEnum, 118
- librarysharingmanager.cpp, 351
- librarysharingmanager.h, 352
- Line
  - dtUtil::Exception, 117
- LoadResource
  - dtUtil::ResourceLoader, 249
  - dtUtil::ResourceManager, 250
- LoadSharedLibrary
  - dtUtil::InternalLibraryHandle, 181
  - dtUtil::LibrarySharingManager, 203
- LOCAL\_DATE\_AND\_TIME\_FORMAT
  - dtUtil::DateTime::TimeFormat, 278
- LOCAL\_DATE\_FORMAT
  - dtUtil::DateTime::TimeFormat, 278
- LOCAL\_ROTATION\_NODE\_NAME
  - dtUtil::HotSpotFileHandler, 170
- LOCAL\_TIME
  - dtUtil::DateTime::TimeOrigin, 279
- LOCAL\_TRANSLATION\_NODE\_NAME
  - dtUtil::HotSpotFileHandler, 170
- LODFlag
  - dtUtil::NodeCollector, 226
- LODNodeMap
  - dtUtil::NodeCollector, 221
- Log
  - dtUtil::Log, 206
- log.cpp, 353
- log.h, 354
  - LOG\_ALWAYS, 354
  - LOG\_DEBUG, 354
  - LOG\_ERROR, 354
  - LOG\_INFO, 355
  - LOG\_WARNING, 355
  - LOGN\_ALWAYS, 355
  - LOGN\_DEBUG, 355
  - LOGN\_ERROR, 355
  - LOGN\_INFO, 355
  - LOGN\_WARNING, 355
- LOG\_ALWAYS
  - dtUtil::Log, 206
  - log.h, 354
- LOG\_DEBUG
  - dtUtil::Log, 206
  - log.h, 354
- LOG\_ERROR
  - dtUtil::Log, 206
  - log.h, 354
- LOG\_INFO
  - dtUtil::Log, 206
  - log.h, 355
- LOG\_WARNING
  - dtUtil::Log, 206
  - log.h, 355
- LogException
  - dtUtil::Exception, 117
- logFile
  - dtUtil::LogManager, 211
- LogHorizRule
  - dtUtil::Log, 207
- LogImpl
  - dtUtil::LogImpl, 210
- LogManager
  - dtUtil::LogManager, 211
- LogMessage
  - dtUtil::Log, 207, 208
- LogMessageType
  - dtUtil::Log, 206
- LOGN\_ALWAYS
  - log.h, 355
- LOGN\_DEBUG
  - log.h, 355
- LOGN\_ERROR
  - log.h, 355
- LOGN\_INFO
  - log.h, 355
- LOGN\_WARNING
  - log.h, 355
- LOGNAME
  - dtUtil, 31
- ltCmp
  - dtUtil::ObjectFactory, 240
- M -**
- macros.h, 356
  - BIT, 356
  - DECLARE\_MANAGEMENT\_LAYER, 356
  - DTUNREFERENCED\_PARAMETER, 356
  - IMPLEMENT\_MANAGEMENT\_LAYER, 356
  - IS\_A, 356
  - UNSIGNED\_BIT, 356
  - UNSIGNED\_INT\_BIT, 356
- MagneticNorthLatitude

- dtUtil, 31
  - MagneticNorthLongitude
    - dtUtil, 31
  - mainpage.h, 357
  - Make
    - dtUtil::CallParms< TYPELIST\_0()>, 62
    - dtUtil::CallParms< TYPELIST\_1(P1)>, 63
    - dtUtil::CallParms< TYPELIST\_2(P1, P2)>, 64
    - dtUtil::CallParms< TYPELIST\_3(P1, P2, P3)>, 65
    - dtUtil::CallParms< TYPELIST\_4(P1, P2, P3, P4)>, 66
    - dtUtil::CallParms< TYPELIST\_5(P1, P2, P3, P4, P5)>, 67
    - dtUtil::CallParms< TYPELIST\_6(P1, P2, P3, P4, P5, P6)>, 68
    - dtUtil::CallParms< TYPELIST\_7(P1, P2, P3, P4, P5, P6, P7)>, 69
  - MakeDirectory
    - dtUtil::FileUtils, 125
  - MakeFunctor
    - dtUtil, 27, 28
  - MakeIndexString
    - dtUtil, 28
  - MakeNoiseTexture
    - dtUtil::NoiseTexture, 234
  - manager
    - dtUtil, 28
  - MapRangeValue
    - dtUtil, 28
  - Marble
    - dtUtil::Fractal, 128
  - Match
    - dtUtil, 28
  - MaterialFlag
    - dtUtil::StateAttributeCollector, 263
  - MaterialNodeMap
    - dtUtil::StateAttributeCollector, 260
  - mathdefines.h, 358
    - RAND\_MAX, 359
  - MatrixToEulers
    - dtUtil::Coordinates, 84
  - MatrixToHpr
    - dtUtil::MatrixUtil, 213
  - MatrixToHprAndPosition
    - dtUtil::MatrixUtil, 213
  - MatrixTransformFlag
    - dtUtil::NodeCollector, 227
  - MatrixTransformNodeMap
    - dtUtil::NodeCollector, 221
  - matrixutil.cpp, 360
  - matrixutil.h, 361
  - Max
    - dtUtil, 28
  - MAX\_DELTA\_LONG
    - dtUtil, 31
  - MAX\_EASTING
    - dtUtil, 31
  - MAX\_LAT
    - dtUtil, 31
  - MAX\_NORTHING
    - dtUtil, 31
  - MAX\_PATH
    - fileutils.cpp, 334
  - MAX\_SCALE\_FACTOR
    - dtUtil, 31
  - max\_size
    - dtUtil::KDTree, 195
  - mBoundingBox
    - dtUtil::BoundingBoxVisitor, 59
  - mDefaultName
    - dtUtil::LogImpl, 210
  - MemberType
    - dtUtil::Command1, 72
  - MemberType1
    - dtUtil::Command2, 74
  - MemberType2
    - dtUtil::Command2, 74
  - MergeParms
    - dtUtil, 28
    - dtUtil::MergeParmsH, 215
    - dtUtil::MergeParmsH::Bound, 57
    - dtUtil::MergeParmsH::Unbound, 300
    - dtUtil::MergeParmsH< 0, BoundPTL, UnboundPTL, dtUtil::NullType, TL >, 216
    - dtUtil::MergeParmsH< i, BoundPTL, UnboundPTL, IdsTL, dtUtil::NullType >, 217
  - METERS\_PER\_DEGREE
    - dtUtil, 31
  - mFileName
    - dtUtil::Exception, 117
  - mFunctions
    - DeprecationMgrImpl, 105
  - mGeomList
    - dtUtil::GeometryCollector, 164
  - MilsToDegrees
    - dtUtil::Coordinates, 84
  - Min
    - dtUtil, 28
  - MIN\_EASTING
    - dtUtil, 31
  - MIN\_LAT
    - dtUtil, 31
  - MIN\_NORTHING
    - dtUtil, 31
  - MIN\_SCALE\_FACTOR
    - dtUtil, 31
  - mLineNum
    - dtUtil::Exception, 117
  - mLocalRotation
    - dtUtil::HotSpotDefinition, 168
  - mLocalTranslation
    - dtUtil::HotSpotDefinition, 168
  - mMessage
    - dtUtil::Exception, 117
  - mMutex
    - dtUtil::LogManager, 211
  - mName
    - dtUtil::HotSpotDefinition, 168
    - dtUtil::LogImpl, 210
  - mOutputStreamBit
    - dtUtil::LogImpl, 210
  - mParentName
    - dtUtil::HotSpotDefinition, 168
  - mwin.h, 362
  - mType
    - dtUtil::Exception, 117
  - MultiSwitchFlag
    - dtUtil::NodeCollector, 227
  - MultiSwitchNodeMap
    - dtUtil::NodeCollector, 221
- N -**
- name

- dtUtil::Packager::PackTreeData, 243
- NAME\_ATTRIBUTE\_NAME
  - dtUtil::HotSpotFileHandler, 170
- NewFunctionName
  - DeprecatedFunction, 103
- next
  - dtUtil::tree, 285
  - dtUtil::tree\_iterator, 289
- NO\_OUTPUT
  - dtUtil::Log, 206
- NodeCollector
  - dtUtil::NodeCollector, 221
- nodecollector.cpp, 363
- nodecollector.h, 364
- NodeFlag
  - dtUtil::NodeCollector, 221
- NodePrintOut
  - dtUtil::NodePrintOut, 228
- nodeprintout.cpp, 365
- nodeprintout.h, 366
- Noise1
  - dtUtil::Noise1, 230
- noise1.h, 367
- Noise1d
  - dtUtil, 23
- Noise1f
  - dtUtil, 23
- Noise2
  - dtUtil::Noise2, 231
- noise2.h, 368
- Noise2d
  - dtUtil, 23
- Noise2f
  - dtUtil, 23
- Noise3
  - dtUtil::Noise3, 232
- noise3.h, 369
- Noise3d
  - dtUtil, 23
- Noise3f
  - dtUtil, 23
- NoiseTexture
  - dtUtil::NoiseTexture, 233
- noisetexture.cpp, 370
- noisetexture.h, 371
- noiseutility.h, 372
- NoLeakAlloc
  - dtUtil::\_Alloc\_base::NoLeakAlloc, 236
- NonConstNoRef
  - dtUtil::details::TypeTraits, 298
- NonConstRef
  - dtUtil::details::TypeTraits, 298
- NULL
  - tree.h, 396
- O -
- ObjectFactory
  - dtUtil::ObjectFactory, 240
- objectfactory.h, 373
- ObjectMap
  - dtUtil::ObjectFactory, 240
- ObjType
  - dtUtil::FunTraits< R(\*)>, 140
  - dtUtil::FunTraits< R(\*)>(P1), 141
  - dtUtil::FunTraits< R(\*)>(P1, P2), 142
  - dtUtil::FunTraits< R(\*)>(P1, P2, P3), 143
  - dtUtil::FunTraits< R(\*)>(P1, P2, P3, P4), 144
  - dtUtil::FunTraits< R(\*)>(P1, P2, P3, P4, P5), 145
  - dtUtil::FunTraits< R(\*)>(P1, P2, P3, P4, P5, P6), 146
  - dtUtil::FunTraits< R(\*)>(P1, P2, P3, P4, P5, P6, P7), 147
  - dtUtil::FunTraits< R(O::\*)() const >, 148
  - dtUtil::FunTraits< R(O::\*)()>, 149
  - dtUtil::FunTraits< R(O::\*)(P1) const >, 150
  - dtUtil::FunTraits< R(O::\*)(P1)>, 151
  - dtUtil::FunTraits< R(O::\*)(P1, P2) const >, 152
  - dtUtil::FunTraits< R(O::\*)(P1, P2)>, 153
  - dtUtil::FunTraits< R(O::\*)(P1, P2, P3) const >, 154
  - dtUtil::FunTraits< R(O::\*)(P1, P2, P3)>, 155
  - dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4) const >, 156
  - dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4)>, 157
- ObjTypeIrtor
  - dtUtil::ObjectFactory, 240
- ObjTypeIrtorConst
  - dtUtil::ObjectFactory, 240
- OldFunctionName
  - DeprecatedFunction, 103
- OpenFile
  - dtUtil::LogManager, 211
- OpenPackage
  - dtUtil::Packager, 242
- operator const std::string &
  - dtUtil::RefString, 247
- operator std::string
  - dtUtil::DateTime, 100
- operator time\_t
  - dtUtil::DateTime, 101
- operator tm
  - dtUtil::DateTime, 101
- operator<
  - dtUtil::Enumeration, 111
  - dtUtil::RefString, 248
- operator<<
  - dtUtil, 28
  - dtUtil::DataStream, 92, 93
- operator>
  - dtUtil::Enumeration, 112
- operator>>
  - dtUtil::DataStream, 93
- operator\*
  - dtUtil::\_Iterator, 42
  - dtUtil::tree, 285
  - dtUtil::tree\_iterator, 289
- operator()
  - dtUtil::\_Bracket\_accessor, 40
  - dtUtil::\_Node\_compare, 45
  - dtUtil::always\_true, 48
  - dtUtil::array\_remove, 51
  - dtUtil::AttributeSearch, 52
  - dtUtil::Binder, 56
  - dtUtil::Command, 70
  - dtUtil::Command0, 71
  - dtUtil::Command1, 72
  - dtUtil::Command2, 75
  - dtUtil::DoNothing, 106
  - dtUtil::DoNothing0, 107
  - dtUtil::EdgeStepFilter, 108
  - dtUtil::EvaluateFuncor, 113
  - dtUtil::EvaluateFuncor< FuncorType \*, ArgType, RetType >, 114
  - dtUtil::EvaluateInvoke, 115
  - dtUtil::Funcor, 131

- dtUtil::insert\_back, 174
- dtUtil::insert\_back\_no\_duplicates, 175
- dtUtil::lsDelimiter, 183
- dtUtil::lsSlash, 188
- dtUtil::lsSpace, 189
- dtUtil::squared\_difference, 257
- dtUtil::squared\_difference\_counted, 258
- dtUtil::ToLowerClass, 281
- dtUtil::Transformation, 282
- operator+
  - dtUtil, 28
  - dtUtil::RefString, 248
- operator++
  - dtUtil::\_Iterator, 42
  - dtUtil::tree\_iterator, 289
- operator->
  - dtUtil::\_Iterator, 42
  - dtUtil::RefString, 248
  - dtUtil::tree\_iterator, 289
- operator--
  - dtUtil::\_Iterator, 42
  - dtUtil::tree\_iterator, 289
- operator=
  - dtUtil::Coordinates, 84
  - dtUtil::DataStream, 93
  - dtUtil::DateTime, 101
  - dtUtil::Functor, 131
  - dtUtil::KDTree, 195
  - dtUtil::RefString, 248
  - dtUtil::tree, 285
  - dtUtil::tree\_iterator, 289
  - dtUtil::TupleHolder, 291
- operator==
  - dtUtil, 28, 29
  - dtUtil::\_Iterator, 42
  - dtUtil::Coordinates, 84
  - dtUtil::Enumeration, 111
  - dtUtil::RefString, 248
  - dtUtil::tree, 285
  - dtUtil::tree\_iterator, 289
- operator[]
  - dtUtil::RefString, 248
  - dtUtil::tree, 285
  - dtUtil::tree\_iterator, 289
- optimize
  - dtUtil::KDTree, 195
- ordern
  - dtUtil::InstantiateH< TypeList< T, U >, Holder, i >, 177
- ORDINAL\_DATE\_FORMAT
  - dtUtil::DateTime::TimeFormat, 278
- out
  - dtUtil::tree, 285, 286
  - dtUtil::tree\_iterator, 290
- Outgoing
  - dtUtil::Binder, 56
- OutputStreamOptions
  - dtUtil::Log, 206
- P -**
- p
  - dtUtil, 31
- Packager
  - dtUtil::Packager, 241
- packager.cpp, 374
- packager.h, 375
- PackPackage
  - dtUtil::Packager, 242
- Param0
  - dtUtil::Command1, 72
  - dtUtil::Command2, 74
- Param1
  - dtUtil::Command2, 74
- param\_type
  - dtUtil::TypeTraits, 299
- Params
  - dtUtil::Command1, 72
  - dtUtil::Command2, 75
- parent
  - dtUtil::Packager::PackTreeData, 243
- Parm1
  - dtUtil::Binder, 56
  - dtUtil::BoundHelper, 58
  - dtUtil::Functor, 131
  - dtUtil::FunTraits< R(\*) (P1)>, 141
  - dtUtil::FunTraits< R(\*) (P1, P2)>, 142
  - dtUtil::FunTraits< R(\*) (P1, P2, P3)>, 143
  - dtUtil::FunTraits< R(\*) (P1, P2, P3, P4)>, 144
  - dtUtil::FunTraits< R(\*) (P1, P2, P3, P4, P5)>, 145
  - dtUtil::FunTraits< R(\*) (P1, P2, P3, P4, P5, P6)>, 146
  - dtUtil::FunTraits< R(\*) (P1, P2, P3, P4, P5, P6, P7)>, 147
  - dtUtil::FunTraits< R(O::\*) (P1) const >, 150
  - dtUtil::FunTraits< R(O::\*) (P1)>, 151
  - dtUtil::FunTraits< R(O::\*) (P1, P2) const >, 152
  - dtUtil::FunTraits< R(O::\*) (P1, P2)>, 153
  - dtUtil::FunTraits< R(O::\*) (P1, P2, P3) const >, 154
  - dtUtil::FunTraits< R(O::\*) (P1, P2, P3)>, 155
  - dtUtil::FunTraits< R(O::\*) (P1, P2, P3, P4) const >, 156
  - dtUtil::FunTraits< R(O::\*) (P1, P2, P3, P4)>, 157
  - dtUtil::FunTraits< R(O::\*) (P1, P2, P3, P4, P5) const >, 158
  - dtUtil::FunTraits< R(O::\*) (P1, P2, P3, P4, P5)>, 159
  - dtUtil::FunTraits< R(O::\*) (P1, P2, P3, P4, P5, P6) const >, 160
  - dtUtil::FunTraits< R(O::\*) (P1, P2, P3, P4, P5, P6)>, 161
  - dtUtil::FunTraits< R(O::\*) (P1, P2, P3, P4, P5, P6, P7) const >, 162
  - dtUtil::FunTraits< R(O::\*) (P1, P2, P3, P4, P5, P6, P7)>, 163
  - dtUtil::UnboundHelper, 301
- Parm2
  - dtUtil::Binder, 56
  - dtUtil::BoundHelper, 58
  - dtUtil::Functor, 131
  - dtUtil::FunTraits< R(\*) (P1, P2)>, 142
  - dtUtil::FunTraits< R(\*) (P1, P2, P3)>, 143
  - dtUtil::FunTraits< R(\*) (P1, P2, P3, P4)>, 144
  - dtUtil::FunTraits< R(\*) (P1, P2, P3, P4, P5)>, 145
  - dtUtil::FunTraits< R(\*) (P1, P2, P3, P4, P5, P6)>, 146
  - dtUtil::FunTraits< R(\*) (P1, P2, P3, P4, P5, P6, P7)>, 147
  - dtUtil::FunTraits< R(O::\*) (P1, P2) const >, 152
  - dtUtil::FunTraits< R(O::\*) (P1, P2)>, 153
  - dtUtil::FunTraits< R(O::\*) (P1, P2, P3) const >, 154
  - dtUtil::FunTraits< R(O::\*) (P1, P2, P3)>, 155
  - dtUtil::FunTraits< R(O::\*) (P1, P2, P3, P4) const >, 156
  - dtUtil::FunTraits< R(O::\*) (P1, P2, P3, P4)>, 157
  - dtUtil::FunTraits< R(O::\*) (P1, P2, P3, P4, P5) const >, 158
  - dtUtil::FunTraits< R(O::\*) (P1, P2, P3, P4, P5)>, 159
  - dtUtil::FunTraits< R(O::\*) (P1, P2, P3, P4, P5, P6) const >, 160

- dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5, P6)>, 161
- dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5, P6, P7) const >, 162
- dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5, P6, P7)>, 163
- dtUtil::UnboundHelper, 301
- Parm3
  - dtUtil::Binder, 56
  - dtUtil::BoundHelper, 58
  - dtUtil::Functor, 131
  - dtUtil::FunTraits< R(\*) (P1, P2, P3)>, 143
  - dtUtil::FunTraits< R(\*) (P1, P2, P3, P4)>, 144
  - dtUtil::FunTraits< R(\*) (P1, P2, P3, P4, P5)>, 145
  - dtUtil::FunTraits< R(\*) (P1, P2, P3, P4, P5, P6)>, 146
  - dtUtil::FunTraits< R(\*) (P1, P2, P3, P4, P5, P6, P7)>, 147
  - dtUtil::FunTraits< R(O::\*)(P1, P2, P3) const >, 154
  - dtUtil::FunTraits< R(O::\*)(P1, P2, P3)>, 155
  - dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4) const >, 156
  - dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4)>, 157
  - dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5) const >, 158
  - dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5)>, 159
  - dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5, P6) const >, 160
  - dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5, P6)>, 161
  - dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5, P6, P7) const >, 162
  - dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5, P6, P7)>, 163
  - dtUtil::UnboundHelper, 301
- Parm4
  - dtUtil::Binder, 56
  - dtUtil::BoundHelper, 58
  - dtUtil::Functor, 131
  - dtUtil::FunTraits< R(\*) (P1, P2, P3, P4)>, 144
  - dtUtil::FunTraits< R(\*) (P1, P2, P3, P4, P5)>, 145
  - dtUtil::FunTraits< R(\*) (P1, P2, P3, P4, P5, P6)>, 146
  - dtUtil::FunTraits< R(\*) (P1, P2, P3, P4, P5, P6, P7)>, 147
  - dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4) const >, 156
  - dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4)>, 157
  - dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5) const >, 158
  - dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5)>, 159
  - dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5, P6) const >, 160
  - dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5, P6)>, 161
  - dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5, P6, P7) const >, 162
  - dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5, P6, P7)>, 163
  - dtUtil::UnboundHelper, 301
- Parm5
  - dtUtil::Binder, 56
  - dtUtil::BoundHelper, 58
  - dtUtil::Functor, 131
  - dtUtil::FunTraits< R(\*) (P1, P2, P3, P4, P5)>, 145
  - dtUtil::FunTraits< R(\*) (P1, P2, P3, P4, P5, P6)>, 146
  - dtUtil::FunTraits< R(\*) (P1, P2, P3, P4, P5, P6, P7)>, 147
  - dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5) const >, 158
  - dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5)>, 159
  - dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5, P6) const >, 160
- dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5, P6)>, 161
- dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5, P6, P7) const >, 162
- dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5, P6, P7)>, 163
- dtUtil::UnboundHelper, 301
- Parm6
  - dtUtil::Functor, 131
  - dtUtil::FunTraits< R(\*) (P1, P2, P3, P4, P5, P6)>, 146
  - dtUtil::FunTraits< R(\*) (P1, P2, P3, P4, P5, P6, P7)>, 147
  - dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5, P6) const >, 160
  - dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5, P6)>, 161
  - dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5, P6, P7) const >, 162
  - dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5, P6, P7)>, 163
- Parm7
  - dtUtil::Functor, 131
  - dtUtil::FunTraits< R(\*) (P1, P2, P3, P4, P5, P6, P7)>, 147
  - dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5, P6, P7) const >, 162
  - dtUtil::FunTraits< R(O::\*)(P1, P2, P3, P4, P5, P6, P7)>, 163
- ParmsListType
  - dtUtil::CallParams< TYPELIST\_0()>, 62
  - dtUtil::CallParams< TYPELIST\_1(P1)>, 63
  - dtUtil::CallParams< TYPELIST\_2(P1, P2)>, 64
  - dtUtil::CallParams< TYPELIST\_3(P1, P2, P3)>, 65
  - dtUtil::CallParams< TYPELIST\_4(P1, P2, P3, P4)>, 66
  - dtUtil::CallParams< TYPELIST\_5(P1, P2, P3, P4, P5)>, 67
  - dtUtil::CallParams< TYPELIST\_6(P1, P2, P3, P4, P5, P6)>, 68
  - dtUtil::CallParams< TYPELIST\_7(P1, P2, P3, P4, P5, P6, P7)>, 69
  - dtUtil::Functor, 131
  - dtUtil::FunctorCall< CallType, R, TYPELIST\_0()>, 132
  - dtUtil::FunctorCall< CallType, R, TYPELIST\_1(P1)>, 133
  - dtUtil::FunctorCall< CallType, R, TYPELIST\_2(P1, P2)>, 134
  - dtUtil::FunctorCall< CallType, R, TYPELIST\_3(P1, P2, P3)>, 135
  - dtUtil::FunctorCall< CallType, R, TYPELIST\_4(P1, P2, P3, P4)>, 136
  - dtUtil::FunctorCall< CallType, R, TYPELIST\_5(P1, P2, P3, P4, P5)>, 137
  - dtUtil::FunctorCall< CallType, R, TYPELIST\_6(P1, P2, P3, P4, P5, P6)>, 138
  - dtUtil::FunctorCall< CallType, R, TYPELIST\_7(P1, P2, P3, P4, P5, P6, P7)>, 139
- Parse
  - dtUtil::XercesParser, 307
- ParseVec
  - dtUtil, 29
- path
  - dtUtil::FileInfo, 120
- PATH\_SEPARATOR
  - dtUtil::FileUtils, 126
- permutation
  - dtUtil, 31
- PITCH\_VEC
  - dtUtil::HotSpotFileHandler, 170

- pointer
  - dtUtil::\_Iterator, 42
  - dtUtil::KDTree, 194
- pointer\_type
  - dtUtil::TypeTraits, 299
- polardecomp.cpp, 376
- polardecomp.h, 377
- PopDirectory
  - dtUtil::FileUtils, 125
- PositionAndHprToMatrix
  - dtUtil::MatrixUtil, 213
- precomp.cpp, 378
  - DELTA\_PCH, 378
- prev
  - dtUtil::tree, 286
- Print
  - dtUtil::Exception, 117
  - dtUtil::MatrixUtil, 213
- PrintNodeToOSGFile
  - dtUtil::NodePrintOut, 229
- processingInstruction
  - dtUtil::HotSpotFileHandler, 170
- ProgramFlag
  - dtUtil::StateAttributeCollector, 263
- ProgramNodeMap
  - dtUtil::StateAttributeCollector, 260
- push\_back
  - dtUtil::tree, 286
  - dtUtil::tree\_iterator, 290
- push\_front
  - dtUtil::tree, 286
  - dtUtil::tree\_iterator, 290
- PushDirectory
  - dtUtil::FileUtils, 125
- R -**
- RAND\_MAX
  - mathdefines.h, 359
- RandFloat
  - dtUtil, 29
- RandPercent
  - dtUtil, 29
- RandRange
  - dtUtil, 29
- rbegin
  - dtUtil::KDTree, 195
- Read
  - dtUtil::DataStream, 93
- ReadBinary
  - dtUtil::DataStream, 93
- ReconfigureRotationMatrix
  - dtUtil::Coordinates, 84
- RecordablePtrContainer
  - dtUtil::KeyFrameDecoder, 197
- RecordableType
  - dtUtil::KeyFrameDecoder, 197
- reference
  - dtUtil::\_Iterator, 42
  - dtUtil::array\_remove, 51
  - dtUtil::insert\_back\_no\_duplicates, 175
  - dtUtil::KDTree, 194
  - dtUtil::TypeTraits, 299
- RefString
  - dtUtil::RefString, 247
- refstring.cpp, 379
  - THREAD\_SAFETY, 379
- USE\_TABLE, 379
- refstring.h, 380
- RegisterType
  - dtUtil::ObjectFactory, 240
- REGULAR\_FILE
  - dtUtil, 23
- reinsert
  - dtUtil::tree, 286
  - dtUtil::tree\_iterator, 290
- RelativePath
  - dtUtil::FileUtils, 125
- release
  - dtUtil::LibrarySharingManager::LibraryHandle, 201
- Remove
  - dtUtil::Bits, 32
- remove
  - dtUtil::tree, 286
  - dtUtil::tree\_iterator, 290
- RemoveConfigPropertyValue
  - dtUtil::ConfigProperties, 76
- RemoveDOFTransform
  - dtUtil::NodeCollector, 225
- RemoveFile
  - dtUtil::Packager, 242
- RemoveFromSearchPath
  - dtUtil::LibrarySharingManager, 203
- RemoveGeode
  - dtUtil::NodeCollector, 225
- RemoveGroup
  - dtUtil::NodeCollector, 226
- RemoveLOD
  - dtUtil::NodeCollector, 226
- RemoveMatrixTransform
  - dtUtil::NodeCollector, 226
- RemoveMultiSwitch
  - dtUtil::NodeCollector, 226
- RemoveNode
  - dtUtil::CollectorUtil, 34
- RemoveSwitch
  - dtUtil::NodeCollector, 226
- RemoveType
  - dtUtil::ObjectFactory, 240
- rend
  - dtUtil::KDTree, 195
- Reseed
  - dtUtil::Noise1, 230
  - dtUtil::Noise2, 231
  - dtUtil::Noise3, 232
  - dtUtil::SeamlessNoise, 251
- reset
  - dtUtil::squared\_difference\_counted, 258
- resetErrors
  - dtUtil::XercesErrorHandler, 306
- ResourceConstIterator
  - dtUtil::ResourceManager, 250
- ResourceHandle
  - dtUtil::ResourceManager, 250
- ResourceIterator
  - dtUtil::ResourceManager, 250
- resourceloader.h, 381
- ResourceManager
  - dtUtil::ResourceManager, 250
- resourcemanager.h, 382
- ResourceMap
  - dtUtil::ResourceManager, 250
- Result

- dtUtil::BoundTL2, 60
- dtUtil::Filter2, 127
- dtUtil::Filter2::Temp, 270
- dtUtil::Filter2::Temp< dtUtil::NullType, i, dtUtil::NullType, Predicate >, 271
- dtUtil::Filter2::Temp< dtUtil::NullType, i, IdsTL2, Predicate >, 272
- dtUtil::Filter2::Temp< TL2, 0, dtUtil::NullType, dtUtil::IsIntType >, 273
- dtUtil::Filter2::Temp< TL2, 0, dtUtil::NullType, dtUtil::NotIntType >, 274
- dtUtil::Filter2::Temp< TL2, 0, typename dtUtil::IdsFromTL< TL2 >::Type, dtUtil::IsIntType >, 275
- dtUtil::Filter2::Temp< TL2, 0, typename dtUtil::IdsFromTL< TL2 >::Type, dtUtil::NotIntType >, 276
- dtUtil::Filter2::Temp< TL2, i, dtUtil::NullType, Predicate >, 277
- dtUtil::Select, 253
- dtUtil::Select< false, T, U >, 254
- dtUtil::TailAt< InstantiateH< TypeList< T, U >, Holder, i >, 0, i >, 267
- dtUtil::TailAt< InstantiateH< TypeList< T, U >, Holder, i >, j, i >, 268
- dtUtil::TypeAt< TypeList< T, U >, 0 >, 292
- dtUtil::TypeAt< TypeList< T, U >, i >, 293
- dtUtil::TypeAtNonStrict, 294
- dtUtil::TypeAtNonStrict< TypeList< T, U >, 0, DefType >, 295
- dtUtil::TypeAtNonStrict< TypeList< T, U >, i, DefType >, 296
- dtUtil::UnboundTL2, 302
- result\_type
  - dtUtil::\_Bracket\_accessor, 40
- ResultMap
  - dtUtil::AttributeSearch, 52
- ResultType
  - dtUtil::Binder, 56
  - dtUtil::Functor, 131
  - dtUtil::FunTraits< R(\*)()>, 140
  - dtUtil::FunTraits< R(\*)<P1>>, 141
  - dtUtil::FunTraits< R(\*)<P1, P2>>, 142
  - dtUtil::FunTraits< R(\*)<P1, P2, P3>>, 143
  - dtUtil::FunTraits< R(\*)<P1, P2, P3, P4>>, 144
  - dtUtil::FunTraits< R(\*)<P1, P2, P3, P4, P5>>, 145
  - dtUtil::FunTraits< R(\*)<P1, P2, P3, P4, P5, P6>>, 146
  - dtUtil::FunTraits< R(\*)<P1, P2, P3, P4, P5, P6, P7>>, 147
  - dtUtil::FunTraits< R(O::\*)() const >, 148
  - dtUtil::FunTraits< R(O::\*)()>, 149
  - dtUtil::FunTraits< R(O::\*)<P1> const >, 150
  - dtUtil::FunTraits< R(O::\*)<P1>>, 151
  - dtUtil::FunTraits< R(O::\*)<P1, P2> const >, 152
  - dtUtil::FunTraits< R(O::\*)<P1, P2>>, 153
  - dtUtil::FunTraits< R(O::\*)<P1, P2, P3> const >, 154
  - dtUtil::FunTraits< R(O::\*)<P1, P2, P3>>, 155
  - dtUtil::FunTraits< R(O::\*)<P1, P2, P3, P4> const >, 156
  - dtUtil::FunTraits< R(O::\*)<P1, P2, P3, P4>>, 157
  - dtUtil::FunTraits< R(O::\*)<P1, P2, P3, P4, P5> const >, 158
  - dtUtil::FunTraits< R(O::\*)<P1, P2, P3, P4, P5>>, 159
  - dtUtil::FunTraits< R(O::\*)<P1, P2, P3, P4, P5, P6> const >, 160
  - dtUtil::FunTraits< R(O::\*)<P1, P2, P3, P4, P5, P6>>, 161
  - dtUtil::FunTraits< R(O::\*)<P1, P2, P3, P4, P5, P6, P7> const >, 162
  - dtUtil::FunTraits< R(O::\*)<P1, P2, P3, P4, P5, P6, P7>>, 163
  - dtUtil::MergeParmsH, 215
  - return\_type
    - dtUtil::TypeTraits, 299
  - ReturnType
    - dtUtil::Command, 70
  - reverse\_iterator
    - dtUtil::KDTree, 194
  - Rewind
    - dtUtil::DataStream, 93
  - RightBase
    - dtUtil::InstantiateH< TypeList< T, U >, Holder, i >, 177
    - dtUtil::InstantiateHAccessor< j, InstantiateH< TypeList< T, U >, Holder, i >, i >, 179
  - RigidMultiFractal
    - dtUtil::Fractal, 129
  - ROLL\_VEC
    - dtUtil::HotSpotFileHandler, 170
  - S -
  - S\_ISDIR
    - fileutils.cpp, 334
  - S\_ISREG
    - fileutils.cpp, 334
  - safeASIN
    - dtUtil, 29
  - Scan
    - dtUtil, 29
  - SCENARIO\_TIME
    - dtUtil::DateTime::TimeType, 280
  - SeamlessFractal
    - dtUtil, 23
  - SeamlessNoise
    - dtUtil::SeamlessNoise, 251
  - seamlessnoise.cpp, 383
    - LERP, 383
  - seamlessnoise.h, 384
  - Seekg
    - dtUtil::DataStream, 93
  - Seekp
    - dtUtil::DataStream, 93
  - seekPos
    - dtUtil::Packager::PackTreeData, 243
  - semiMajorAxis
    - dtUtil, 31
  - serializer.cpp, 385
  - serializer.h, 386
  - SET
    - dtUtil::DataStream::SeekTypeEnum, 252
  - set\_high\_bound
    - dtUtil::\_Region, 47
  - set\_low\_bound
    - dtUtil::\_Region, 47
  - SetAmplitude
    - dtUtil::NoiseTexture, 234
  - SetApplyRotationConversionMatrix
    - dtUtil::Coordinates, 84
  - SetBufferSize
    - dtUtil::DataStream, 93
  - SetConfigPropertyValue
    - dtUtil::ConfigProperties, 76
  - SetDay
    - dtUtil::DateTime, 101

- setDocumentLocator
  - dtUtil::HotSpotFileHandler, 170
- SetFileName
  - dtUtil::LogFile, 209
- SetFlatEarthOrigin
  - dtUtil::Coordinates, 85
- SetForceLittleEndian
  - dtUtil::DataStream, 93
- SetFrequency
  - dtUtil::NoiseTexture, 234
- SetGlobeRadius
  - dtUtil::Coordinates, 85
- SetGMTOffset
  - dtUtil::DateTime, 101
- SetHeight
  - dtUtil::NoiseTexture, 234
- SetHour
  - dtUtil::DateTime, 101
- SetIncomingCoordinateType
  - dtUtil::Coordinates, 85
- SetLocalCoordinateType
  - dtUtil::Coordinates, 85
- SetLocalOffset
  - dtUtil::Coordinates, 85
- SetLogLevel
  - dtUtil::Log, 208
- SetMagneticNorthOffset
  - dtUtil::Coordinates, 85
- SetMinute
  - dtUtil::DateTime, 101
- SetMonth
  - dtUtil::DateTime, 101
- SetOctaves
  - dtUtil::NoiseTexture, 234
- SetOutputStreamBit
  - dtUtil::Log, 208
- SetPersistence
  - dtUtil::NoiseTexture, 234
- SetRepeat
  - dtUtil::SeamlessNoise, 251
- SetResourceLoader
  - dtUtil::ResourceManager, 250
- SetRow
  - dtUtil::MatrixUtil, 213, 214
- SetSecond
  - dtUtil::DateTime, 101
- SetSlices
  - dtUtil::NoiseTexture, 234
- SetTime
  - dtUtil::DateTime, 101
- SetTimeFormat
  - dtUtil::DateTime, 102
- SetTimeOrigin
  - dtUtil::DateTime, 102
- SetTimeScale
  - dtUtil::DateTime, 102
- SetTimeType
  - dtUtil::DateTime, 102
- SetTitle
  - dtUtil::LogFile, 209
- SetToGMTTime
  - dtUtil::DateTime, 102
- SetToLocalTime
  - dtUtil::DateTime, 102
- SetUTMHemisphere
  - dtUtil::Coordinates, 85
- SetUTMLocalOffsetAsLatLon
  - dtUtil::Coordinates, 85
- SetUTMZone
  - dtUtil::Coordinates, 85
- SetWidth
  - dtUtil::NoiseTexture, 234
- SetYear
  - dtUtil::DateTime, 102
- SIMULATION\_TIME
  - dtUtil::DateTime::TimeType, 280
- size
  - dtUtil::FileInfo, 120
  - dtUtil::KDTree, 195
  - dtUtil::tree, 286
  - dtUtil::tree\_iterator, 290
- size\_type
  - dtUtil::KDTree, 194
- skippedEntity
  - dtUtil::HotSpotFileHandler, 170
- sLogFileName
  - dtUtil, 31
- source
  - dtUtil::Packager::PackTreeData, 243
- SPHSN
  - dtUtil::UTMPParameters, 304
- SPHSR
  - dtUtil::UTMPParameters, 304
- SPHTMD
  - dtUtil::UTMPParameters, 304
- squared\_difference\_counted
  - dtUtil::squared\_difference\_counted, 258
- src/ Directory Reference, 11
- src/dtUtil/ Directory Reference, 7
- STANDARD
  - dtUtil::Log, 206
- startDocument
  - dtUtil::HotSpotFileHandler, 170
- startElement
  - dtUtil::HotSpotFileHandler, 170
- startPrefixMapping
  - dtUtil::HotSpotFileHandler, 170
- stat64
  - fileutils.cpp, 334
- StateAttribFlag
  - dtUtil::StateAttributeCollector, 260
- StateAttributeCollector
  - dtUtil::StateAttributeCollector, 260
- stateattributecollector.cpp, 387
- stateattributecollector.h, 388
- stateSetParser
  - dtUtil::StateVisitor, 264
- StateVisitor
  - dtUtil::StateVisitor, 264
- sTitle
  - dtUtil, 29
- StoredType
  - dtUtil::TupleHolder, 291
- StringCount
  - dtUtil, 31
- StringSet
  - dtUtil, 31
- StringToXMLConverter
  - dtUtil::StringToXMLConverter, 266
- stringutils.cpp, 389
- stringutils.h, 390
- subvalue\_type

- dtUtil::\_Region, 47
- dtUtil::KDTree, 194
- SwitchFlag
  - dtUtil::NodeCollector, 227
- SwitchNodeMap
  - dtUtil::NodeCollector, 221
- SYMBOL\_ADDRESS
  - dtUtil::LibrarySharingManager::LibraryHandle, 200
- T -**
- Tail
  - dtUtil::TypeList, 297
- TangentSpaceVisitor
  - dtUtil::TangentSpaceVisitor, 269
- tangentspacevisitor.cpp, 392
- tangentspacevisitor.h, 393
- TargetHolder
  - dtUtil::InstantiateHAccessor< 0, InstantiateH< TypeList< T, U >, Holder, i >, i >, 178
  - dtUtil::InstantiateHAccessor< j, InstantiateH< TypeList< T, U >, Holder, i >, i >, 179
- templateutility.h, 394
- TextureFlag
  - dtUtil::StateAttributeCollector, 263
- TextureNodeMap
  - dtUtil::StateAttributeCollector, 260
- THREAD\_SAFETY
  - refstring.cpp, 379
- TIME\_STAMP
  - dtUtil::DateTime::TimeType, 280
- TIME\_TYPE\_OTHER
  - dtUtil::DateTime::TimeType, 280
- TimeTag
  - dtUtil::LogManager, 211
- TO\_CONSOLE
  - dtUtil::Log, 206
- TO\_FILE
  - dtUtil::Log, 206
- ToBool
  - dtUtil::Serializer, 255
- ToDouble
  - dtUtil, 29
  - dtUtil::Serializer, 255
- ToFloat
  - dtUtil, 29
  - dtUtil::Serializer, 255
- Toggle
  - dtUtil::Bits, 32
- ToInt
  - dtUtil::Serializer, 256
- tokenize
  - dtUtil::StringTokenizer, 265
- ToLowerClass
  - dtUtil::ToLowerClass, 281
- ToString
  - dtUtil, 29
  - dtUtil::DateTime, 102
  - dtUtil::Exception, 117
  - dtUtil::Serializer, 256
  - dtUtil::XMLStringConverter, 309
- ToType
  - dtUtil, 29
- ToType< bool >
  - dtUtil, 30
- ToUnsignedInt
  - dtUtil, 30
- ToXmlString
  - dtUtil::StringToXMLConverter, 266
- TranMerc\_a
  - dtUtil::UTMPParameters, 304
- TranMerc\_ap
  - dtUtil::UTMPParameters, 304
- TranMerc\_bp
  - dtUtil::UTMPParameters, 304
- TranMerc\_cp
  - dtUtil::UTMPParameters, 304
- TranMerc\_Delta\_Easting
  - dtUtil::UTMPParameters, 304
- TranMerc\_Delta\_Northing
  - dtUtil::UTMPParameters, 304
- TranMerc\_dp
  - dtUtil::UTMPParameters, 304
- TranMerc\_ebs
  - dtUtil::UTMPParameters, 304
- TranMerc\_ep
  - dtUtil::UTMPParameters, 304
- TranMerc\_es
  - dtUtil::UTMPParameters, 304
- TranMerc\_f
  - dtUtil::UTMPParameters, 304
- TranMerc\_False\_Easting
  - dtUtil::UTMPParameters, 304
- TranMerc\_False\_Northing
  - dtUtil::UTMPParameters, 304
- TranMerc\_Origin\_Lat
  - dtUtil::UTMPParameters, 304
- TranMerc\_Origin\_Long
  - dtUtil::UTMPParameters, 304
- TranMerc\_Scale\_Factor
  - dtUtil::UTMPParameters, 304
- Transform
  - dtUtil::BarycentricSpace, 53
- transformation.h, 395
- TransformNodeMap
  - dtUtil::NodeCollector, 221
- TransformVec3
  - dtUtil::MatrixUtil, 214
- Transpose
  - dtUtil::MatrixUtil, 214
- tree
  - dtUtil::tree, 284
- tree.h, 396
  - NULL, 396
- tree\_find\_breadth
  - dtUtil::tree, 286
  - dtUtil::tree\_iterator, 290
- tree\_find\_depth
  - dtUtil::tree, 286
  - dtUtil::tree\_iterator, 290
- tree\_iterator
  - dtUtil::tree\_iterator, 288
- tree\_ptr
  - dtUtil::tree\_iterator, 290
- tree\_ref
  - dtUtil::tree\_iterator, 290
- Trim
  - dtUtil, 30
- trim
  - dtUtil, 30
- TRIP\_TIME
  - dtUtil::DateTime::TimeType, 280
- TupleHolder

- dtUtil::TupleHolder, 291
- Turbulence
  - dtUtil::Fractal, 129
- Type
  - dtUtil::AppendTL, 49
  - dtUtil::AppendTL< NullType, T >, 50
  - dtUtil::CreateldsTL, 86
  - dtUtil::CreateldsTL<-1,-1,-1,-1,-1,-1,-1,-1 >, 87
  - dtUtil::CreateTL, 88
  - dtUtil::CreateTL< NullType, NullType, NullType, NullType, NullType, NullType, NullType, NullType >, 89
  - dtUtil::IdsFromTL, 171
  - dtUtil::IdsFromTL< NullType, i >, 172
  - dtUtil::TupleHolder, 291
- TypeEnum
  - dtUtil::Exception, 117
- TYPELIST\_0
  - generic.h, 343
- TYPELIST\_1
  - dtUtil::Command1, 72
  - dtUtil::FunTraits< R(\*) (P1)>, 141
  - dtUtil::FunTraits< R(O::\*) (P1) const >, 150
  - dtUtil::FunTraits< R(O::\*) (P1)>, 151
  - generic.h, 343
- TYPELIST\_2
  - dtUtil::Command2, 75
  - dtUtil::FunTraits< R(\*) (P1, P2)>, 142
  - dtUtil::FunTraits< R(O::\*) (P1, P2) const >, 152
  - dtUtil::FunTraits< R(O::\*) (P1, P2)>, 153
  - generic.h, 343
- TYPELIST\_3
  - dtUtil::FunTraits< R(\*) (P1, P2, P3)>, 143
  - dtUtil::FunTraits< R(O::\*) (P1, P2, P3) const >, 154
  - dtUtil::FunTraits< R(O::\*) (P1, P2, P3)>, 155
  - generic.h, 343
- TYPELIST\_4
  - dtUtil::FunTraits< R(\*) (P1, P2, P3, P4)>, 144
  - dtUtil::FunTraits< R(O::\*) (P1, P2, P3, P4) const >, 156
  - dtUtil::FunTraits< R(O::\*) (P1, P2, P3, P4)>, 157
  - generic.h, 343
- TYPELIST\_5
  - dtUtil::FunTraits< R(\*) (P1, P2, P3, P4, P5)>, 145
  - dtUtil::FunTraits< R(O::\*) (P1, P2, P3, P4, P5) const >, 158
  - dtUtil::FunTraits< R(O::\*) (P1, P2, P3, P4, P5)>, 159
  - generic.h, 343
- TYPELIST\_6
  - dtUtil::FunTraits< R(\*) (P1, P2, P3, P4, P5, P6)>, 146
  - dtUtil::FunTraits< R(O::\*) (P1, P2, P3, P4, P5, P6) const >, 160
  - dtUtil::FunTraits< R(O::\*) (P1, P2, P3, P4, P5, P6)>, 161
  - generic.h, 343
- TYPELIST\_7
  - dtUtil::FunTraits< R(\*) (P1, P2, P3, P4, P5, P6, P7)>, 147
  - dtUtil::FunTraits< R(O::\*) (P1, P2, P3, P4, P5, P6, P7) const >, 162
  - dtUtil::FunTraits< R(O::\*) (P1, P2, P3, P4, P5, P6, P7)>, 163
  - generic.h, 343
- TYPELIST\_8
  - generic.h, 343
- TypeListType
  - dtUtil::Binder, 56
  - dtUtil::Functor, 131
  - dtUtil::FunTraits< R(\*) ()>, 140
  - dtUtil::FunTraits< R(O::\*) () const >, 148
  - dtUtil::FunTraits< R(O::\*) ()>, 149
- typetraits.h, 397
- U -**
- UnboundParamsTL
  - dtUtil::Binder, 56
- UnboundPTL
  - dtUtil::Binder, 56
- UnboundTL
  - dtUtil::UnboundHelper, 301
- UniqueldType
  - dtUtil::ObjectFactory, 240
- UnpackPackage
  - dtUtil::Packager, 242
- UNSIGNED\_BIT
  - macros.h, 356
- UNSIGNED\_INT\_BIT
  - macros.h, 356
- USE\_TABLE
  - refstring.cpp, 379
- UTM
  - dtUtil::IncomingCoordinateType, 173
- UTMPParameters
  - dtUtil::UTMPParameters, 304
- V -**
- valid
  - dtUtil::Functor, 131
- value
  - dtUtil::Int2Type, 180
  - dtUtil::IsIntType, 184
  - dtUtil::IsIntType< Int2Type< i >, i >, 185
  - dtUtil::Length< NullType >, 198
  - dtUtil::Length< TypeList< T, U > >, 199
  - dtUtil::MergeParmsH::Predicate, 245
  - dtUtil::MergeParmsH::Predicate< j, dtUtil::NullType >, 246
  - dtUtil::NotIntType, 237
  - dtUtil::TupleHolder, 291
- value\_acc
  - dtUtil::KDTree, 195
- value\_comp
  - dtUtil::KDTree, 195
- value\_distance
  - dtUtil::KDTree, 195
- value\_type
  - dtUtil::\_Iterator, 42
  - dtUtil::\_Region, 47
  - dtUtil::CollectorUtil::GetElementType, 165
  - dtUtil::KDTree, 194
  - dtUtil::TypeTraits, 299
- version.cpp, 398
  - Delta3DGetLibraryName, 398
  - Delta3DGetVersion, 398
- version.h, 399
  - DELTA3D\_VERSION\_MAJOR, 399
  - DELTA3D\_VERSION\_MINOR, 399
  - DELTA3D\_VERSION\_PATCH, 399
  - Delta3DGetLibraryName, 399
  - Delta3DGetVersion, 399
- visit\_within\_range

dtUtil::KdTree, 195

## - W -

Walk

dtUtil::KeyFrameDecoder, 197

warning

dtUtil::XercesErrorHandler, 306

WEEK\_DATE\_FORMAT

dtUtil::DateTime::TimeFormat, 278

What

dtUtil::Exception, 117

WildMatch

dtUtil, 30

WithinRange

dtUtil, 30

Write

dtUtil::DataStream, 94

WriteBinary

dtUtil::DataStream, 94

WriteBytes

dtUtil::DataStream, 94

WriteFile

dtUtil::XercesWriter, 308

## - X -

XercesErrorHandler

dtUtil::XercesErrorHandler, 306

xerceserrorhandler.cpp, 400

xerceserrorhandler.h, 401

XercesParser

dtUtil::XercesParser, 307

xercesparser.cpp, 402

xercesparser.h, 403

xercesutils.cpp, 404

xercesutils.h, 405

XercesWriter

dtUtil::XercesWriter, 308

xerceswriter.cpp, 406

xerceswriter.h, 407

XMLStringConverter

dtUtil::XMLStringConverter, 309

XYZToMGRS

dtUtil::Coordinates, 85

## - Z -

ZFlop

dtUtil::Coordinates, 85